World Ocean Simulation System (WOSS) library

Generated by Doxygen 1.9.4

1 World Ocean Simulation System (WOSS) Library

Author

Federico Guerra - federico@guerra-tlc.com

Version

1.12.6

This document provides a short techical description of the *World Ocean Simulation System* (WOSS) library and of its integration into *Multi InteRfAce Cross Layer Extension* (NS-Miracle).

1.1 Library overview

The WOSS library aims to model a channel power delay or frequency attenuation profiles according to a more accurate representation of propagation phenomena as described by the laws of acoustical physics [3]. WOSS is a **multi-threaded** framework that permits the integration of any existing underwater channel simulator that expects environmental data as input and provides as output a channel realization represented using the channel profiles just mentioned.

From version 1.3.5 the simplified acoustic model based on empirical equations for the calculation of delay, attenuation and noise power **has been moved into NS-Miracle**. These equations can be found in [1], [2].

Currently WOSS integrates the Bellhop ray-tracing program [4], while the NS-Miracle integration library retains the previously mentioned formula for the noise power.

From version 1.3.0 WOSS supports time evolution, for a better representation of the channel variability. Bellhop calculations are performed across the whole bandwidth with custom resolution, while noise calculations are performed at the geometric average of the highest and the lowest frequencies of the band being used, as pointed in [5]. To calculate the solution of the propagation equations between a transmitter and a receiver, Bellhop requires the knowledge of the Sound Speed Profile (SSP) (The *Sound Speed Profile* is the propagation speed of sound

considered as a function of water depth. Different profiles lead to potentially very different propagation effects, including surface sound channels, deep sound channels, convergence zones, shadow zones, and so on, see [3]), the Bathymetric Profile (BP) and the type of Bottom Sediments (BS, required to model acoustic power losses due to bottom reflections). In this respect, WOSS offers a technology independent database API and a number of classes to manipulate SSP, BP and BS data. For the SSP, it employs a custom implementation of the World Ocean Database [7-11], a collection of SSPs calculated with the TEOS-10 [6] exact formula based on the depth, temperature and salinity data measured during a number of experiments around the world; the measurements are divided by location and day or season of the year when the measurement was performed (recall that sound propagation is affected by water temperature, which in turn undergoes seasonal changes, especially in the superficial layer). The bathymetry data have been taken from the General Bathymetric Chart of the Oceans [12], a public database offering samples of the depth of the sea bottom with an angular spacing of 30 seconds of arc. Finally, the type of bottom sediments is provided by a geo-acoustic analysis of the National Geophysical Data Center's Deck41 data-base [13].

Thanks to this automation the user only has to specify the location in the world and the time where the simulation should take place. This is done by setting the simulated date and the wanted latitude and longitude of every node involved. The simulator automatically handles the rest. In more detail, the simulator picks the location (i.e., latitude, longitude and depth) of the transmitter and the receiver and queries the database manager for samples of bottom sediments, measured SSPs (for simplicity, the SSP can also be assumed to be constant, on average, throughout the network area) and for bathymetry data along the path. Full customization of surficial sediment, bathymetry and SSP data is also possible if more accurate data is available. The power delay profile obtained with Bellhop can be used to model multiple receptions: in detail the user can choose to coherently combine the complex channel gains with a custom resolution time window and obtain one or more complex channel taps.

Figs. 1.1 and 1.2 show the coherent sum of all channel taps, obtained with WOSS for an example scenario. Fig. 1.1, represents the attenuation incurred by an acoustic wave at 4 kHz transmitted in August, approximately 20 km offshore the harbor of the city of Taranto, Southern Italy, with the transmitter located at 40.32N, 17.12E, 300 m depth. Darker shades of gray represent stronger signal power. The figure shows that the signal reaches the top right corner surface, about 11 km from the transmitter, bearing sufficient power to allow correct reception. In contrast, Fig 1.2 shows the same scenario in March: in this case, the average temperature of the water is lower, changing the way the sound is refracted and reflected by the sea bottom. Here, any acoustic receiver deployed between 9 and 11 km from the transmitter may be unable to acquire the transmitted signal. Furthermore, a shadow zone of about 2 km appears in front of the transmitter, causing failure in a hypothetical short range communication scenario.

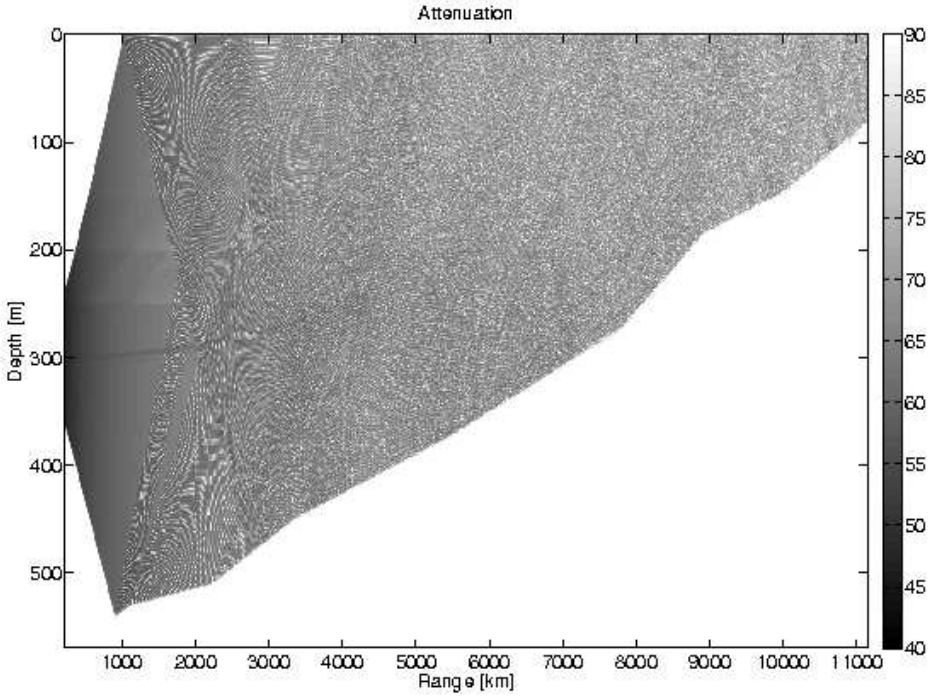


Figure 1 Bellhop in August

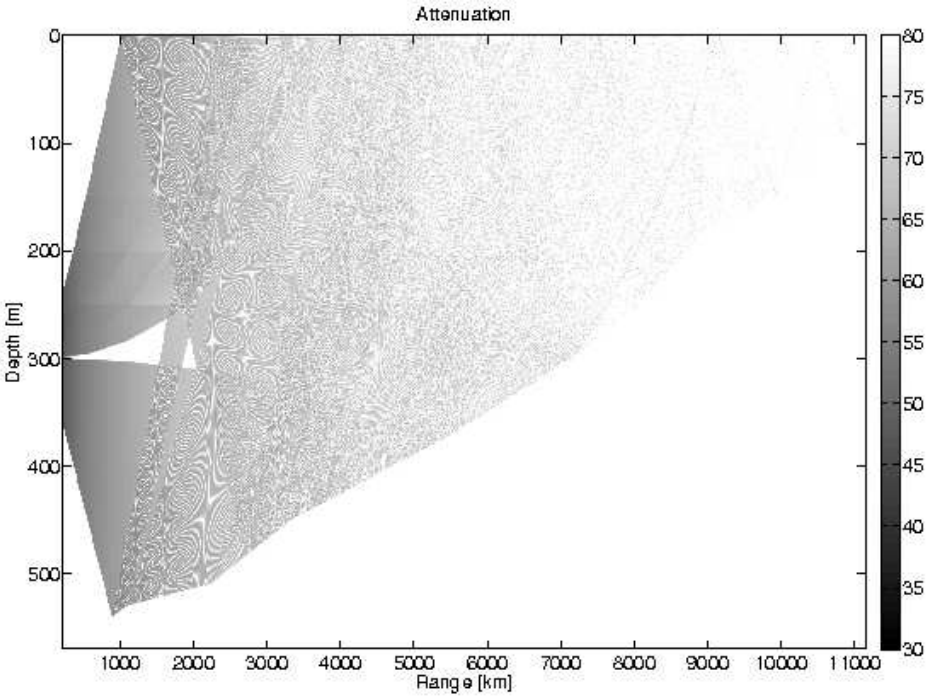


Figure 2 Bellhop in March

1.2 Technical Overview

In this section, a short techical overview of the WOSS library and its *API*s is provided.

1.2.1 The Foundation Classes

WOSS foundation is a complete set of classes for:

- *geographical coordinates* [woss::Coord, woss::CoordZ]: measured in decimal degrees, this classes offers methods for distance, bearing, and marsden square calculations and other mathematical formulas;
- *date time* [woss::Time]: date time arithmetics;
- *simulation time reference* [woss::TimeReference]: importing a simulation time reference from the external network simulator;
- *random number generation* [woss::RandomGenerator]: importing a custom random generator from an external network simulator or library;
- *sound speed profile* [woss::SSP]: sound speed calculations and manipulations with equations taken from the scientific literature;
- *surficial sediment* [woss::Sediment]: geoacoustic sediment parameter definitions and manipulations;
- *channel power delay profile* [woss::TimeArr]: profile sampling, averaging etc.
- *pressure* [woss::Pressure]: arithmetic and conversion operators;
- *transducer* [woss::Transducer]: accurate power consumption calculations and beam pattern capabilities of real transducers;
- *altimetry* [woss::Altimetry]: a base class that should be used for implementing and simulating different sea surface wave spectra.

- a definitions handler class for "plugging-in" at run-time any class that inherits from the ones described above [woss::DefHandler].
- *test suite* [woss::WossTest]: a base class that should be used for writing tests of specific WOSS framework APIs.

1.2.2 The Database APIs

The woss::WossDb API separates database technology implementation (SQL, NetCDF, textual...) from data behaviour implementation, permitting maximum code modularity and re-usability. Data behaviour are defined with WossBathymetryDb, WossSedimentDb, WossSSPDb.

The woss::WossDbCreator API provides interface for creation and initializiation of WossDb objects, providing the user a ready-made database instance.

Custom defined bathymetry, SSP and sediment databases are accessed through the woss::WossDbManager API, to provide a further refinement of data (eg. averaging, arithmetics...).

The woss::WossResPressDb and woss::WossResTimeArrDb classes provide the possibility to store and retrieve channel simulation results in any user defined fashion.

1.2.3 The Channel Simulator APIs

The woss::Woss API has the task of integrating a channel simulator. It gives interface for:

- setting up the enviroment from databases or from custom data;
- configuring, running and reading results of any channel simulator;
- averaging and giving output results over any number of runs and frequencies;
- optionally reading channel simulator's results [woss::ResReader, woss::WossResReader].

The woss::WossCreator API provides interface for creation and initializiation of woss::Woss objects, relieving the user from this task.

The woss::WossManager API has the logic to control and manipulate and create woss::Woss objects. It offers the abilities to:

- carefully create and utilize Woss objects to minimize CPU load;
- plan a strategy for time-varying and/or multi-frequency channel simulations;
- use on-the-fly custom bathymetry, sediment or SSP data for any tx-rx pair.

Finally the woss::WossController API has the task of inter-connecting all major APIs involved.

1.2.4 WOSS block diagram

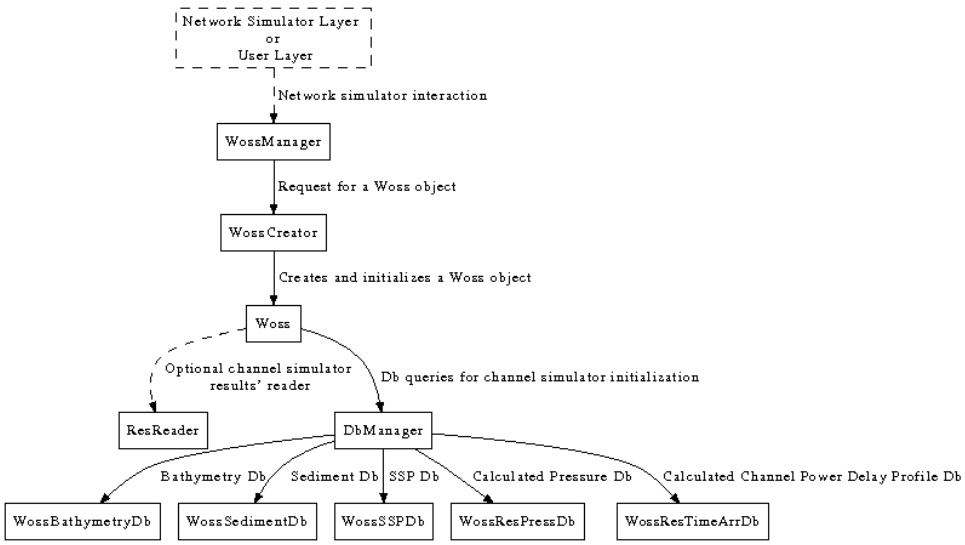


Figure 3 WOSS block diagram

1.2.5 Current capabilities

The current version provides:

- interface implementation for the Bellhop ray tracing program [4], [woss::BellhopWoss, woss::BellhopCreator];
- interface implementation and custom NetCDF db of monthly averaged SSPs, calculated with the TEOS-10 exact formula [6] based on the temperature, salinity and depth data taken from the World Ocean Atlas database of 2001 [7], 2005 [8], 2009 [9], 2013 [10] and 2018 [11] [woss::SspWoa2005Db, woss::SspWoa2005DbCreator];
- interface implementation for the GEBCO NetCDF bathymetry databases (all versions: 1D, 2D, 2008, 2014, 2019, 2020, 2021, 2022) [12], [woss::BathyGebcoDb, woss::BathyGebcoDbCreator];
- interface implementation and custom NetCDF data analysis taken from the DECK41 database, for bottom sediments composition [13], [woss::SedimDeck41Db, woss::SedimDeck41DbCreator, woss::SedimDeck41CoordDb, woss::SedimDeck41MarsdenDb, woss::SedimDeck41MarsdenOneDb];
- A **multi-threaded** version that allows the parallel execution of multiple channel simulators [woss::WossManagerResDbMT];
- a database of real acoustic transducers made from publicly avalaible datasheets;
- an implementation of the Bretschneider sea surface wave spectrum for altimetry simulations;
- support for time evolution, to account for channel variability.
- support for unit testing that are integrated with autotools.

See also

Database Descriptions for further info on custom database, Underwater Acoustic Transducers for further info on transducers, Altimetry models for further info on altimetry models, Time evolution modeling for further info on time evolution modeling.

1.2.6 References

1. R. Urick, *Principles of Underwater Sound*. McGraw-Hill, 1983.
2. M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM Mobile Comput. and Commun. Review*, vol. 11, no. 4, pp. 34–43, 2007.
3. F. Jensen, W. Kuperman, M. Porter, and H. Schmidt, *Computational Ocean Acoustics*. Springer-Verlag, 2000.
4. Bellhop code, http://oalib.hlsresearch.com/Rays/index.html .
5. P. C. Etter, *Underwater acoustic modeling and simulation*. Spon Press, Taylor & Francis group, 2003.
6. TEOS-10 source code http://www.TEOS-10.org .
7. World Ocean Atlas 2001, https://www.nodc.noaa.gov/OC5/WOA01/pr_woa01.html .
8. World Ocean Atlas 2005, https://www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html .
9. World Ocean Atlas 2009, https://www.nodc.noaa.gov/OC5/WOA09/pr_woa09.html .
10. World Ocean Atlas 2013, https://www.nodc.noaa.gov/OC5/woa13/ .
11. World Ocean Atlas 2018, https://www.nodc.noaa.gov/OC5/woa18/ .
12. General bathymetric chart of the oceans, http://www.gebco.net .
13. National geophysical data center, seafloor surficial sediment descriptions, http://www.ngdc.noaa.gov/mgg/geolo
.

2 Database Descriptions

2.1 GEBCO bathymetric NetCDF Database

Please refer to official GEBCO documentation for further info on this NetCDF file.

See also

woss::BathyGebcoDb, woss::BathyGebcoDbCreator

2.2 World Ocean Atlas 2005 and 2009 SSP Databases

These databases contain monthly, seasonal and annual average of sound speed profiles, calculated with the TEOS-10 exact formula, and based on the depth, temperature and salinity provided by the World Ocean Atlas. The available resolution is 1° x 1°. Data are provided at *standard depths* [m] :

0, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500.

The NetCDF files have been generated with NetCDF classic and are compatible with NetCDF4 library. Each NetCDF file has the following dimensions:

- *latitude*: 180 values, from 89.5° to -89.5°
- *longitude*: 360 values, from -179.5° to 179.5°
- *depth*: 33 values (standard depths).

Each NetCDF file has the following variables description:

- *float ssp(latitude, longitude, depth)* ;
  ssp:units = "m/s" ;
- *float latitude(latitude)* ;
  latitude:units = "decimal degrees" ;
- *float longitude(longitude)* ;
  longitude:units = "decimal degrees" ;
- *short depth(depth)* ;
  depth:units = "m" ;

See also

woss::SSP, woss::SspWoa2005Db, woss::SspWoa2005DbCreator

2.3 World Ocean Atlas 2001, 2013 and 2018 SSP Databases

These databases contain monthly, seasonal and annual average of sound speed profiles, calculated with the TEOS-10 exact formula, and based on the depth, temperature and salinity provided by the World Ocean Atlas. The available resolution is 0.25° x 0.25°. Data are provided at *standard depths* [m] :

0, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500.

The databases require NetCDF4 + HDF5 support. Each NetCDF4 file has the following dimensions:

- *latitude*: 720 values, from -89.875° to 89.875°
- *longitude*: 1440 values, from -179.875° to 179.875°
- *depth*: 33 values (standard depths).

Each NetCDF4 file has the following variables description:

- *float ssp(latitude, longitude, depth)* ;
  ssp:units = "m/s" ;
- *float latitude(latitude)* ;
  latitude:units = "decimal degrees" ;
- *float longitude(longitude)* ;
  longitude:units = "decimal degrees" ;
- *short depth(depth)* ;
  depth:units = "m" ;

See also

woss::SSP, woss::SspWoa2005Db, woss::SspWoa2005DbCreator

2.4 DECK41 Sediment NetCDF Database

The DECK41 collection contains surficial sediment descriptions for over 36,000 seafloor samples worldwide. Data include collecting source, ship, cruise, sample id, latitude/longitude, date of collection, water depth, sampling device, dominant lithology, secondary lithology, and a brief description of the surficial sediment at the location

The NetCDF files provided indexes the dominant and secondary lithology for geoacoustical purposes. These values are represented by an unsigned integer number between 0 and 11:

0. Coarser than sand
1. Sand
2. Silt
3. Clay
4. Ooze
5. Mud
6. Rocks, Rocks fragment
7. Organic material (shell, peat, wood, coral, etc.)
8. Nodules, slab or concretions (manganese, phosphate, iron, glauconite)
9. Hard bottom
11. NO_VALUE

These values are collected in three different databases, and they come in different database format versions.

See also

woss::Sediment, woss::SedimDeck41Db, woss::SedimDeck41DbCreator

2.4.1 DECK41 NetCDF database

This database contains all the valid DECK41 coordinates, quantized as per data formats below.

See also

woss::SedimDeck41CoordDb

2.4.1.1 V1 Data Format This database has a NetCDF legacy format, with resolution of 0.016° x 0.016°, with the following dimension:

- a single index dimension covering:
- *latitude*: 10801 values, from 90.0° to -90.0°
- *longitude*: 21601 values, from -180° to 180.0°
- *spacing*: 0.0166666666666667°
- *total indexes* (xysize) 233312401

This NetCDF file has the following variables description:

- *byte seafloor_main_type(xysize)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(xysize)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.1.2 V2 Data Format This database has a NetCDF4 format, with resolution of 0.016° x 0.016°, with the following dimensions:

- *latitude*: 10800 values, from -89.9833333333333333° to 89.9833333333333333°
- *longitude*: 21600 values, from -179.9833333333333333° to 179.9833333333333333°

The *spacing* is 0.0166666666666667° This NetCDF file has the following variables description:

- *byte seafloor_main_type(latitude, longitude)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(latitude, longitude)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.2 DECK41 marsden square NetCDF database

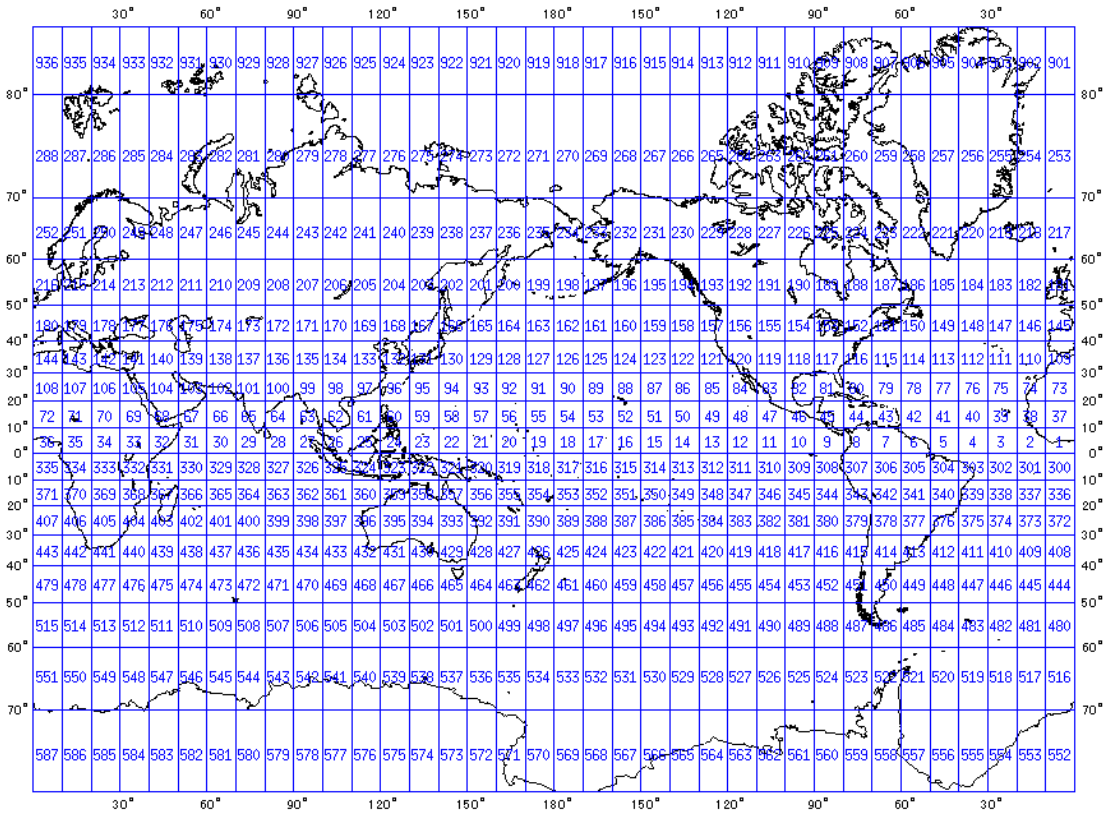


Figure 4 Marsden square map

This database covers all marsden squares. Lithology data of a marsden square are a representative set of all original data for that particular square.

See also

woss::SedimDeck41MarsdenDb

2.4.2.1 V1 Data Format The database is in NetCDF legacy format. It has a single dimension:

- *marsden_square*: 937 values

This NetCDF file has the following variables description:

- *byte seafloor_main_type(marsden_square)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(marsden_square)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.2.2 V2 Data Format The database is in NetCDF4 format. It has a single dimension:

- *marsden_square*: 936 values

This NetCDF file has the following variables description:

- *byte seafloor_main_type(marsden_square)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(marsden_square)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.3 DECK41 marsden coordinates NetCDF database

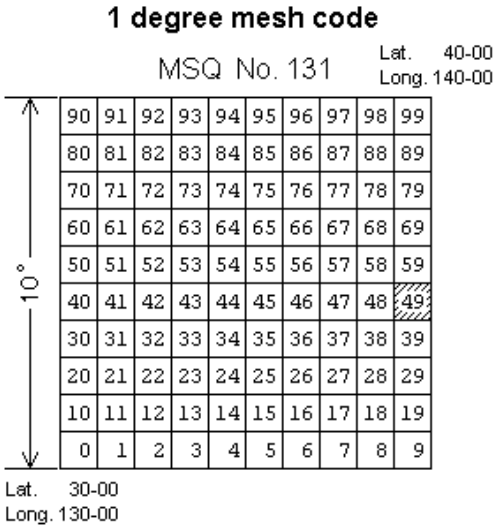


Figure 5 Marsden one degree squares

This database covers all marsden coordinates squares (marsden square, marsden one degree square). Lithology data of a marsden coordinates square are a representative set of all original data for that particular set of marsden coordinates.

See also

woss::SedimDeck41MarsdenOneDb

2.4.3.1 V1 Data Format

The database is in NetCDF legacy data format. It has two dimensions:

- *marsden_square*: 937 values
- *marsden_one_degree*: 100 values

The database has the following variables description:

- *byte seafloor_main_type(marsden_square, marsden_one_degree)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(marsden_square, marsden_one_degree)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.3.2 V2 Data Format

The database is in NetCDF4 data format. It has two dimensions:

- *marsden_square*: 936 values
- *marsden_one_degree*: 100 values

The database has the following variables description:

- *byte seafloor_main_type(marsden_square, marsden_one_degree)* ;
  seafloor_main_type:units = "DECK 41 type" ;
  seafloor_main_type:valid_min = 0 ;
  seafloor_main_type:valid_max = 9 ;

- *byte seafloor_secondary_type(marsden_square, marsden_one_degree)* ;
  seafloor_secondary_type:units = "DECK 41 type" ;
  seafloor_secondary_type:valid_min = 0 ;
  seafloor_secondary_type:valid_max = 9 ;

2.4.4 DECK41 Sediment selection logic

woss::SedimDeck41Db uses all three NetCDF database for its Sediment query process:

1. it first queries the woss::SedimDeck41CoordDb first with provided coordinates;
2. if a suitable result is not found, then it continues the search in woss::SedimDeck41MarsdenOneDb, e.g. in the corrispondant marsden one degree square associated with given coordinates;
3. if even this step fails, it finally searches woss::SedimDeck41MarsdenDb for the marsden square associated.

See also

woss::SedimDeck41Db, woss::Deck41TypeTests

2.4.5 DECK41 geoacoustic parameters

A thorough analysis has been done and is still in process to correctly provide geoacoustical parameters for DECK41 lithology data. The available categories are:

- Coarser than sand - woss::SedimentGravel
- Sand - woss::SedimentSand
- Silt - woss::SedimentSilt
- Clay - woss::SedimentClay
- Ooze - woss::SedimentOoze
- Mud - woss::SedimentMud
- Rocks, Rocks fragment - woss::SedimentRocks
- Organic material (shell, peat, wood, coral, etc.) - woss::SedimentOrganic
- Nodules, slab or concretions (manganese, phosphate, iron, glauconite) - woss::SedimentNodules
- Hard bottom - woss::SedimentHardBottom

See also

woss::Sediment

This is a short list of references for the geoacoustical parameter analysis:

- L. D. Hampton, “Acoustic Properties of Sediments,” *J. Acoust. Soc. Am.*, vol. 42, pp. 882-890, year 1967.
- E. L. Hamilton, “Sound velocity-density relations in sea-floor sediments and rocks,” *J. Acoust. Soc. Am.*, vol. 63, no. 2, pp. 366-377, year 1978.
- E. L. Hamilton, “Geoacoustic modeling of the sea floor,” *J. Acoust. Soc. Am.*, vol. 68, no. 5, pp. 1313-1340, year 1980.
- P. Milholland, M. H. Manghnani, S. O. Schlanger and G. H. Sutton, “Geoacoustic modeling of deep-sea carbonate sediments,” *J. Acoust. Soc. Am.*, vol. 68, no. 5, pp. 1351-1360, year 1980.
- F. A. Bowles, “A Geoacoustic Model for Fine-Grained, Unconsolidated Calcareous Sediments (ARSRP Natural Laboratory),” *Naval Research Lab Stennis Space Center MS*, pp. 58, year 1994.
- F. A. Bowles, “Observations on attenuation and shear-wave velocity in fine-grained, marine sediments,” *J. Acoust. Soc. Am.*, vol. 101, no. 6, pp. 3385-3397, year 1997.
- M. Siderius and J.-P. Hermand, “Yellow Shark Spring 1995: Inversion results from sparse broadband acoustic measurements over a highly range-dependent soft clay layer,” *J. Acoust. Soc. Am.*, vol. 106, no. 2, pp. 637-651, year 1999.

3 Underwater Acoustic Transducers

3.1 DISCLAIMER

Federico Guerra and SIGNET lab do not endorse or recommend the use of any of the transducer listed below.
The copyright and property of of each transducer design is retained by the respective owners. The specifications of each transducer have been inferred from the transducer data sheets courtesy of the respective owners.
The list below is to be taken as a best-effort list, meaning that data has been taken from publicly avalaible documents, at the programmer's discretion.
This list is NOT final and will undergo continuous changes, aimed at providing support for an increasing number of transducers in WOSS.
If you would like any specific, not currently supported transducer to be included in WOSS, please feel free to contribute by sending transducer data (in the format mentioned below) to `nsmiracle-users@dei.unipd.it`.

WOSS accurately simulates power consumption, Sound Pressure Level (SPL) output, and vertical beam pattern of real acoustic transducers.

Transducers can be imported into simulation, and their beam pattern can be rotated or modified with a additive and a multiplicative constant to better suit the user needs.

Furthermore, woss::Location, WossPosition and WossWpPosition support simulation of dynamic orientation of the associated transducer to better simulate how AUV movements impact on transmission.

See also

woss::Location, WossPosition, WossWpPosition

Finally UwMPhyBpskTransducer and WossMPhyBpsk now supports array of transducers to permit spectral optimizations with realistic power computations and ray attenuation.

See also

new sample with waypoints

3.2 Transducer file format (ASCII and binary)

The format of transducer .txt or .dat file can be changed by the user (extending the woss::Transducer class) and is the followings (**`format`** in bold, comments are inline and in *italics*):

```
BT-25UF  transducer's type name
25000.0 10000.0  resonance frequency, -3dB bandwith around the resonance frequency
338 0.1  max input power [W], duty cycle in (0,1]
1.0 1.0 1.0 1.0  TVR precision [hz], OCV precision [hz], conductance precision [hz], beam pattern precision [dec degrees]
5  total number of TVR values
10000.0 120  frequency [hz], TVR [dB re uPa/V @ 1m]
23000.0 140  ...
25000.0 140
38000.0 136
50000.0 135
```

```
4  total number of OCV values
10000.0 -197  frequency [hz], OCV [dB re V/uPa]
20000.0 -190  ...
23000.0 -190
50000.0 -212
6  total number of conductance values
10000.0 10  frequency [hz], conductance [uS]
13000.0 180  ...
23000.0 940
30000.0 410
40000.0 390
50000.0 590
14  total number of angles in the vertical/horiz beam pattern
-180 0  angle between [-180, 180] in decimal degrees, [dB]
-150 -1  if the transducer has conical or != toroidal symmetry
-120 -5  the rotation axis has to lie on angle = 0 axis
-105 -14
-75 -14
-60 -5
-30 -1
0 0
30 -2
60 -4
90 -5
120 -4
150 -2
180 0
```

3.3 Transducer's list

These are the currently available transducers in WOSS (in alphabetical order).

3.4 BTech Acoustics, LLC

3.4.1 BT-1RCL @ 28 kHz

3.4.1.1 Datasheet: Datasheet

3.4.1.2 SPL:

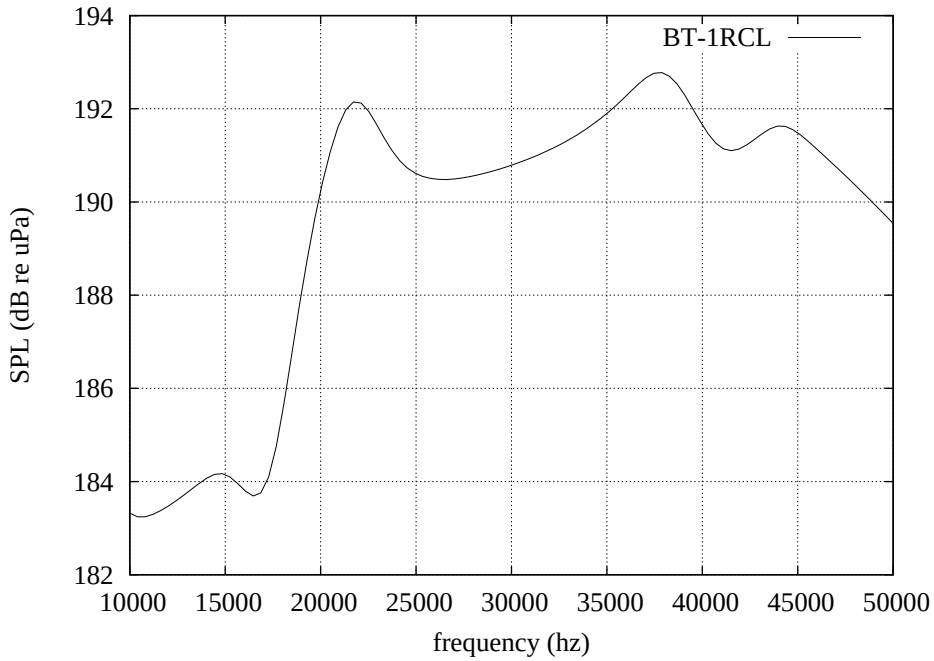


Figure 6 BTech BT-1RCL SPL

3.4.1.3 Vertical beam pattern:

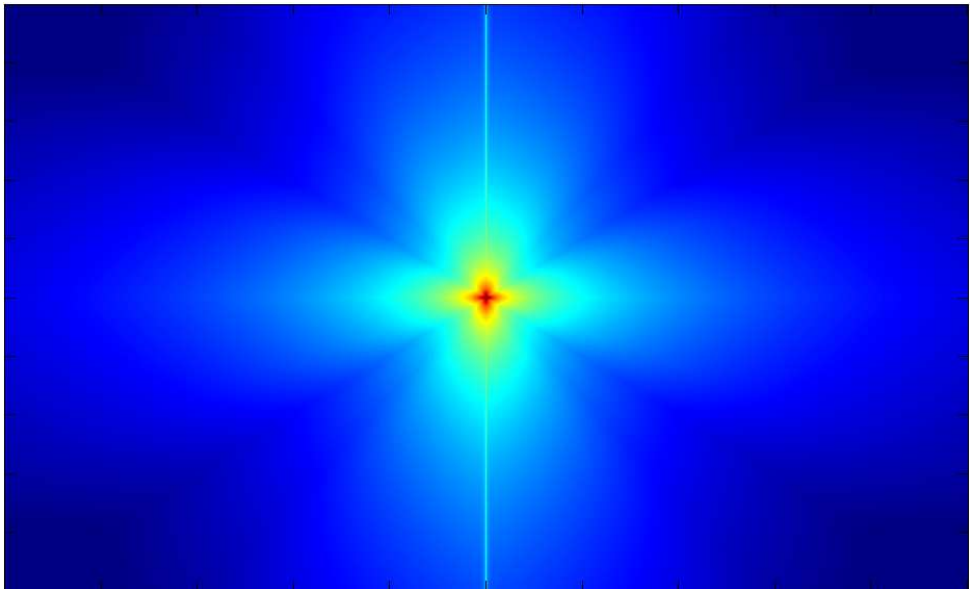


Figure 7 BTech BT-1RCL @ 20 kHz

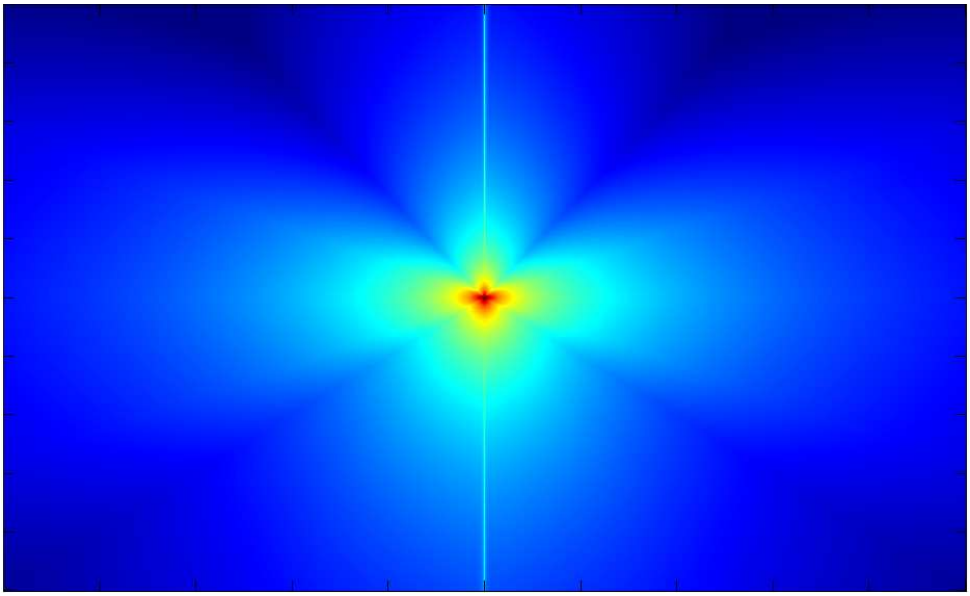


Figure 8 BTech BT-1RCL @ 25 kHz

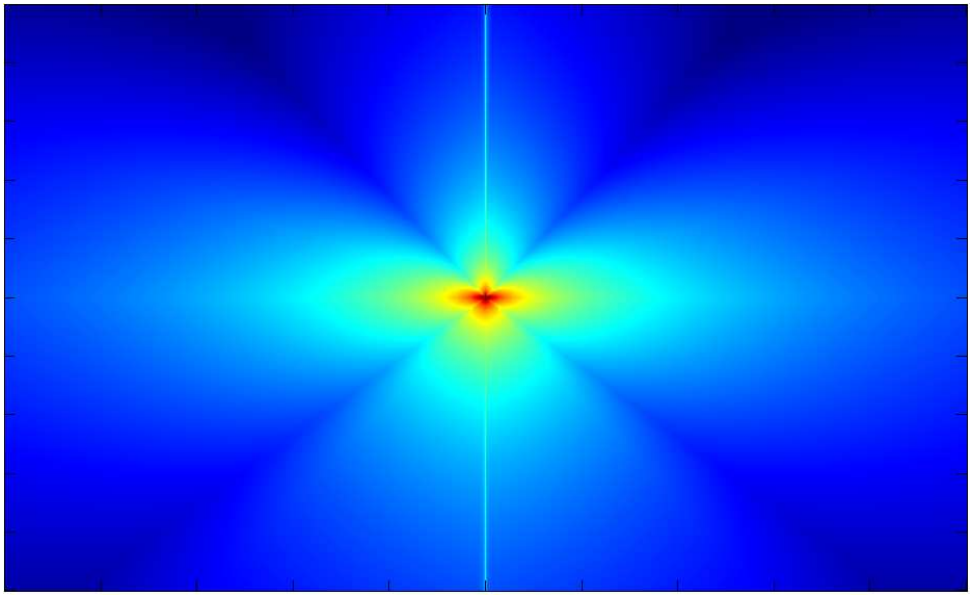


Figure 9 BTech BT-1RCL @ 30 kHz

3.4.2 BT-2RCL @ 28 kHz

3.4.2.1 Datasheet: Datasheet

3.4.2.2 SPL:

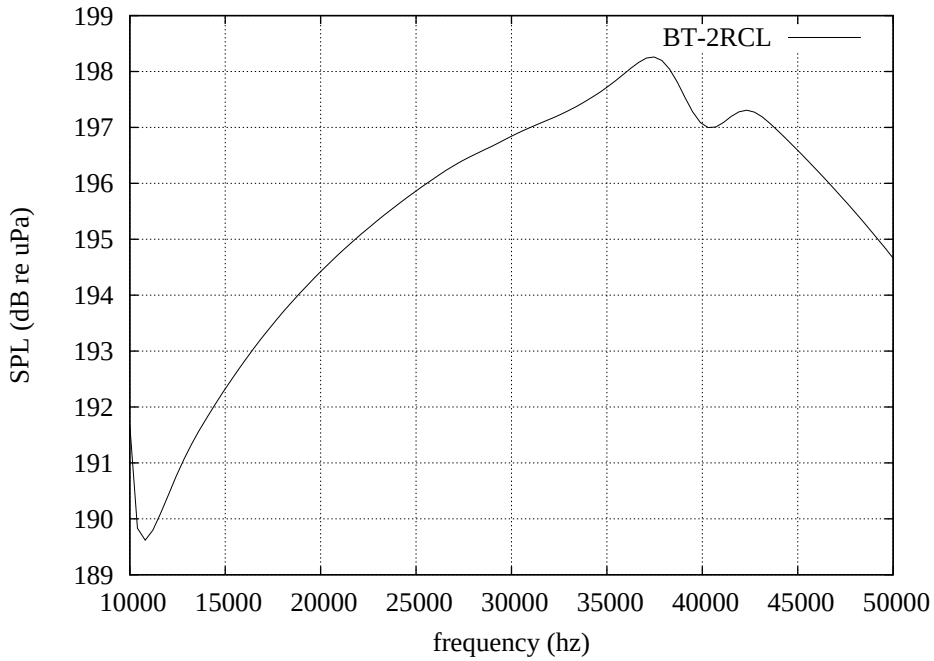


Figure 10 BTech BT-2RCL SPL

3.4.2.3 Vertical beam pattern:

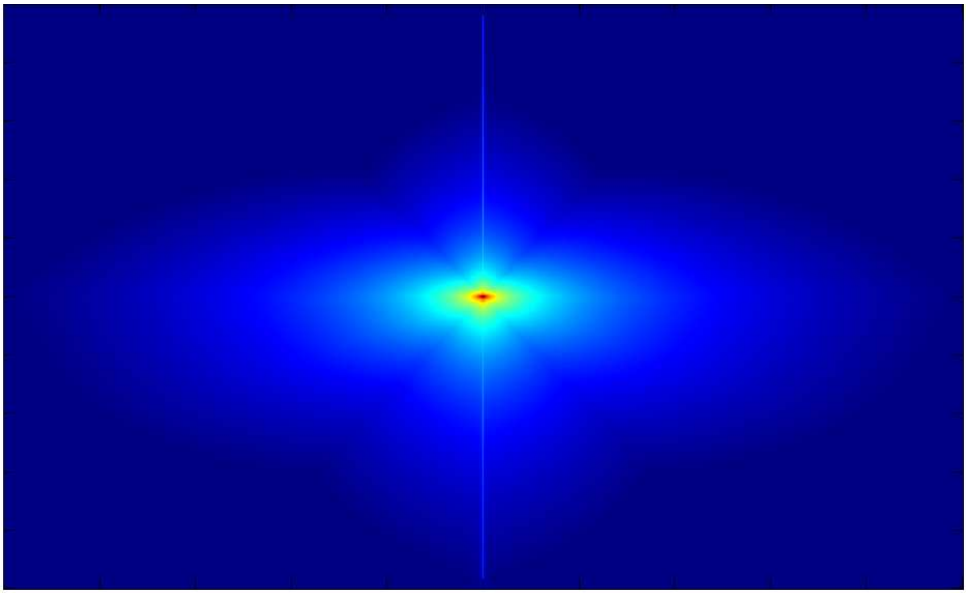


Figure 11 BTech BT-2RCL @ 20 kHz

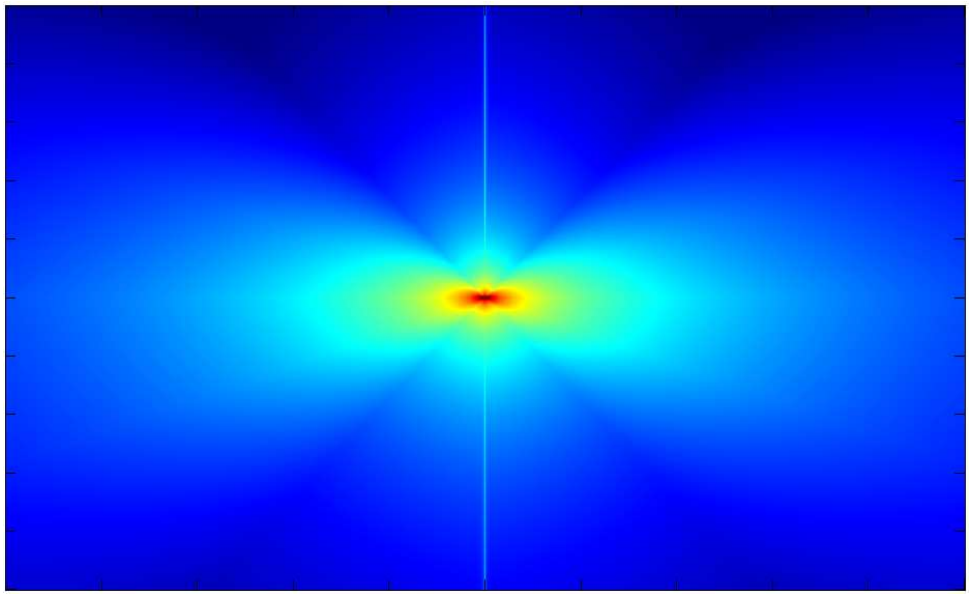


Figure 12 BTech BT-2RCL @ 25 kHz

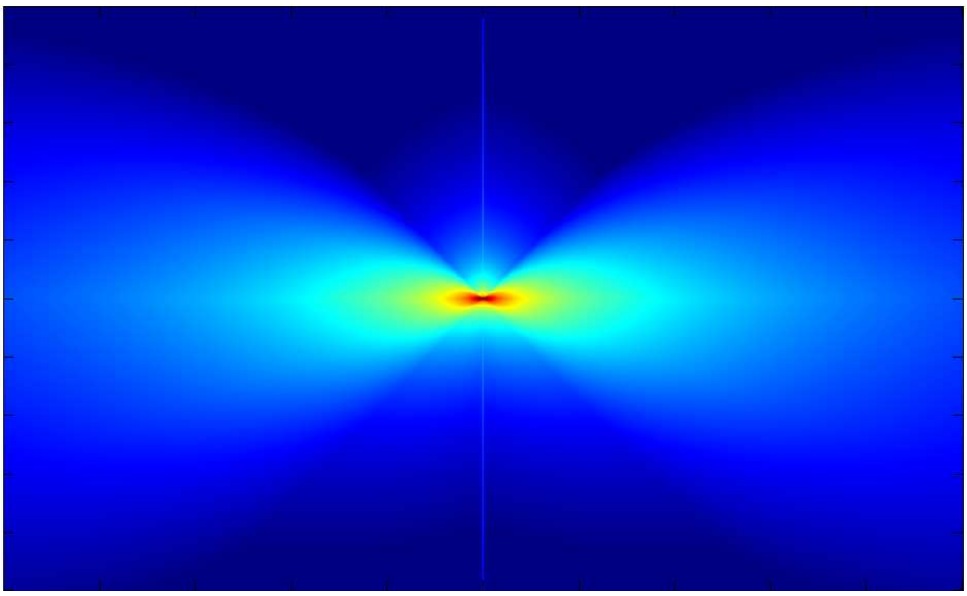


Figure 13 BTech BT-2RCL @ 30 kHz

3.4.3 BT-25UF @ 25 kHz

3.4.3.1 Datasheet: Datasheet

3.4.3.2 SPL:

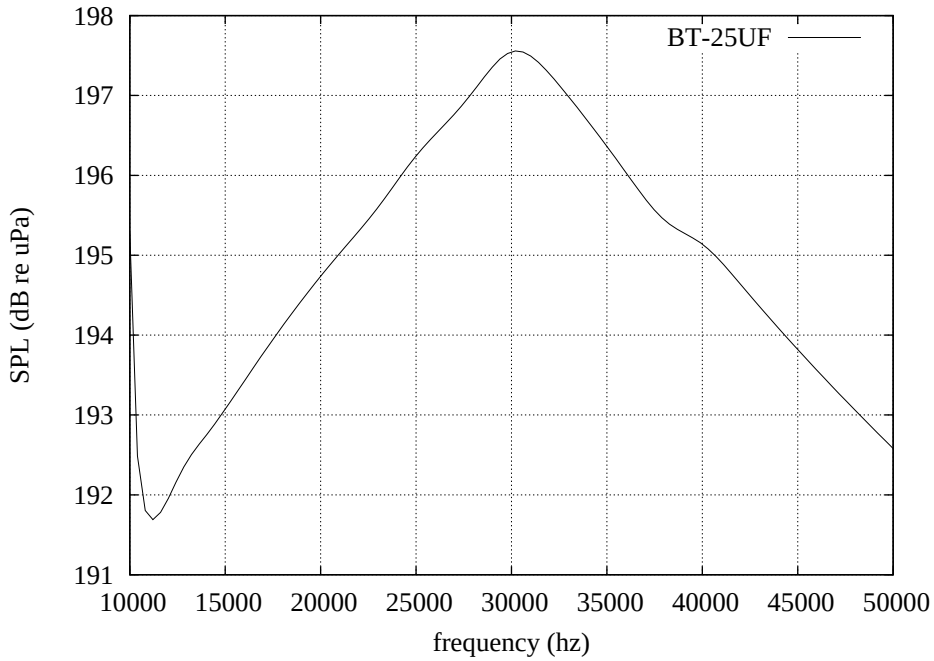


Figure 14 BTech BT-25UF SPL

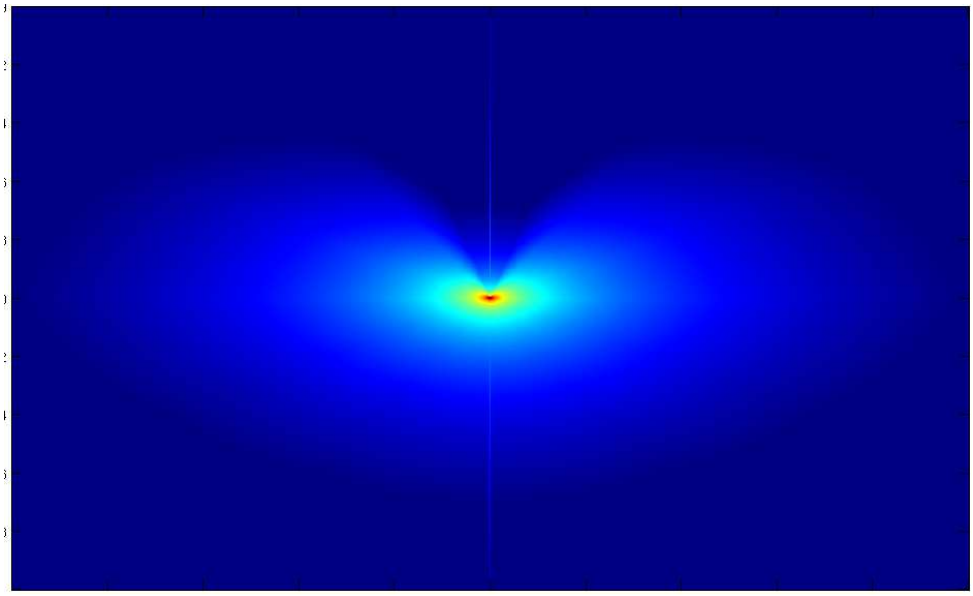


Figure 15 BTech BT-25UF @ 20 kHz

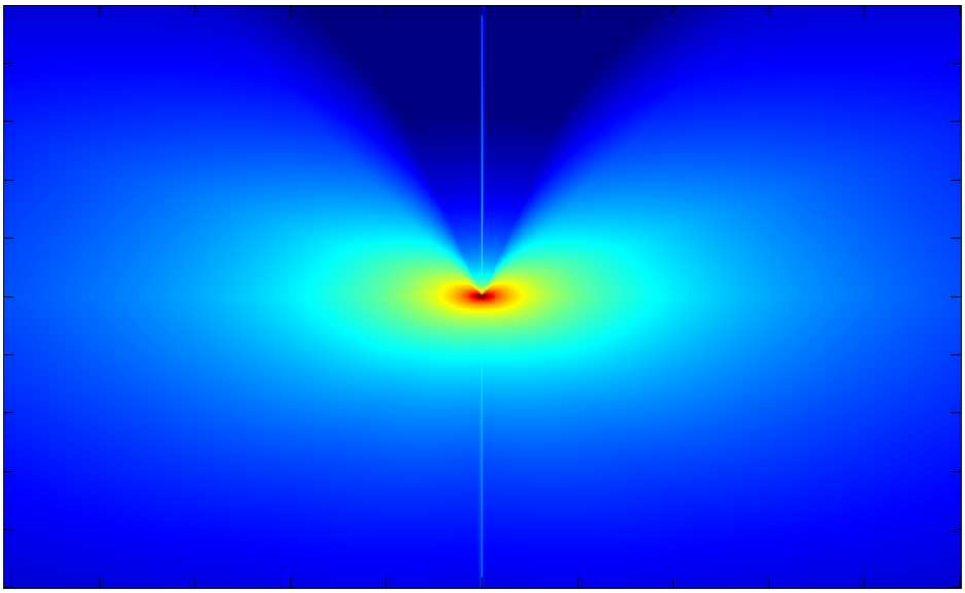


Figure 16 BTech BT-25UF @ 25 kHz

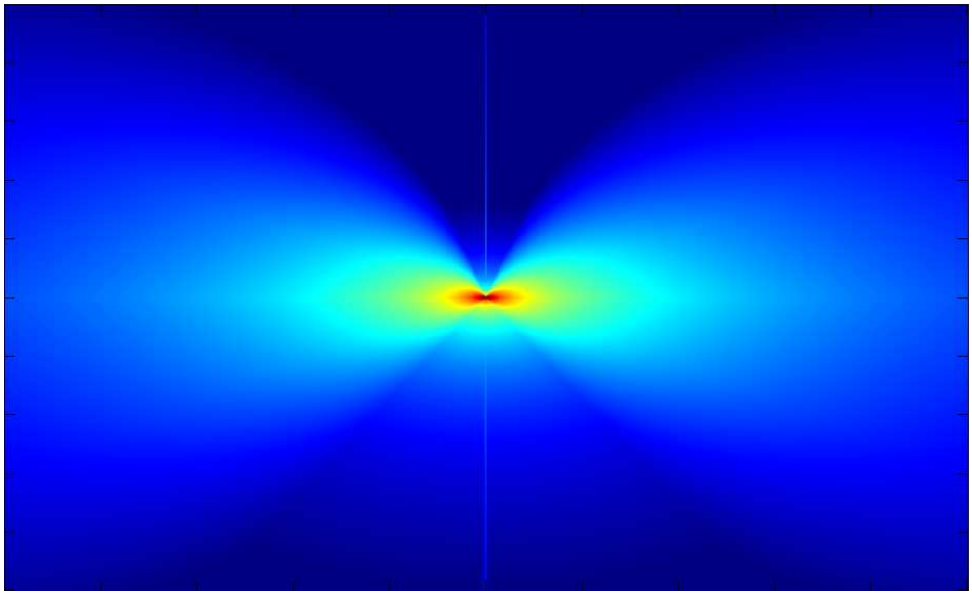


Figure 17 BTech BT-25UF @ 30 kHz

3.5 ITC

3.5.1 ITC-2003 @ 5,9.5 kHz

3.5.1.1 Datasheet: Datasheet

3.5.1.2 SPL:

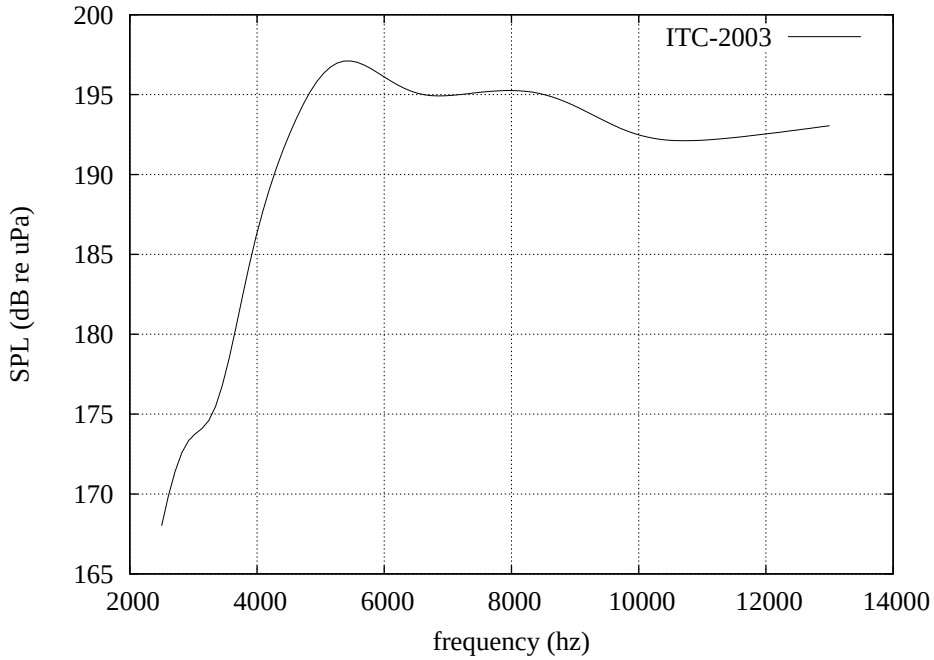


Figure 18 ITC ITC-2003 SPL

3.5.1.3 Vertical beam pattern:

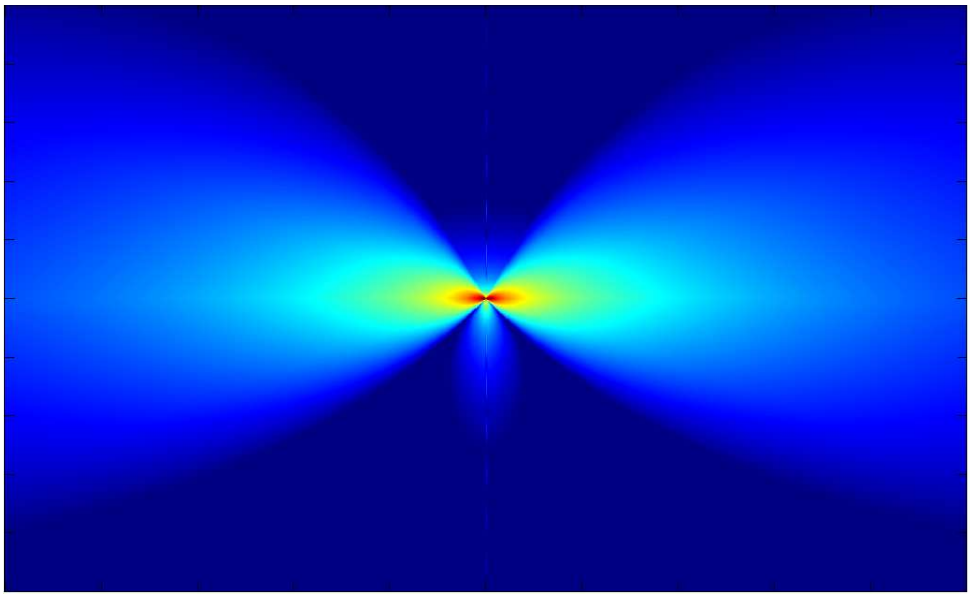


Figure 19 ITC ITC-2003 @ 5 kHz

3.5.2 ITC-2010 @ 1,2.5 kHz

3.5.2.1 Datasheet: Datasheet

3.5.2.2 SPL:

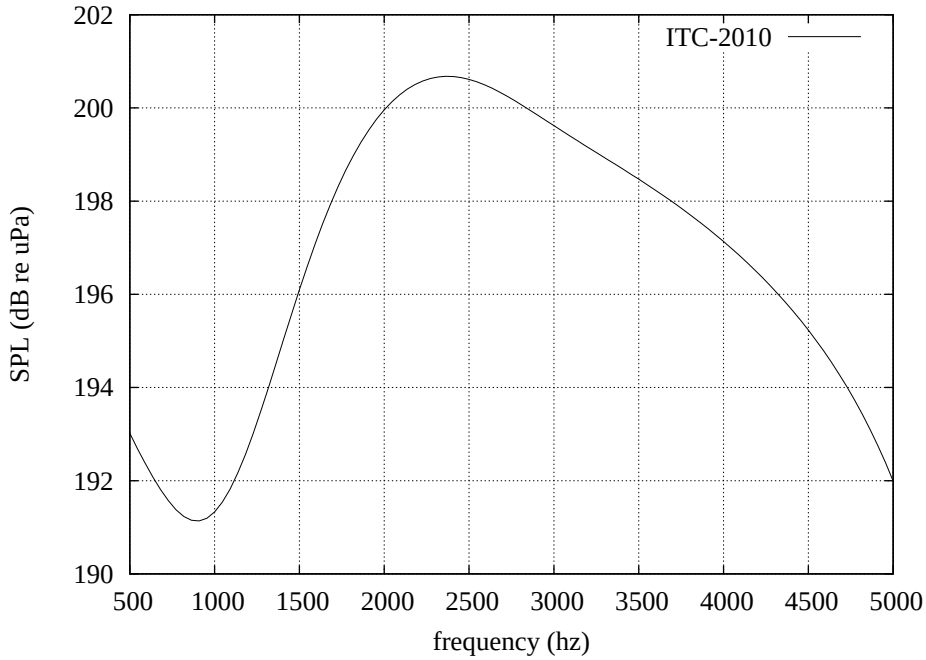


Figure 20 ITC ITC-2010 SPL

3.5.2.3 Vertical beam pattern:

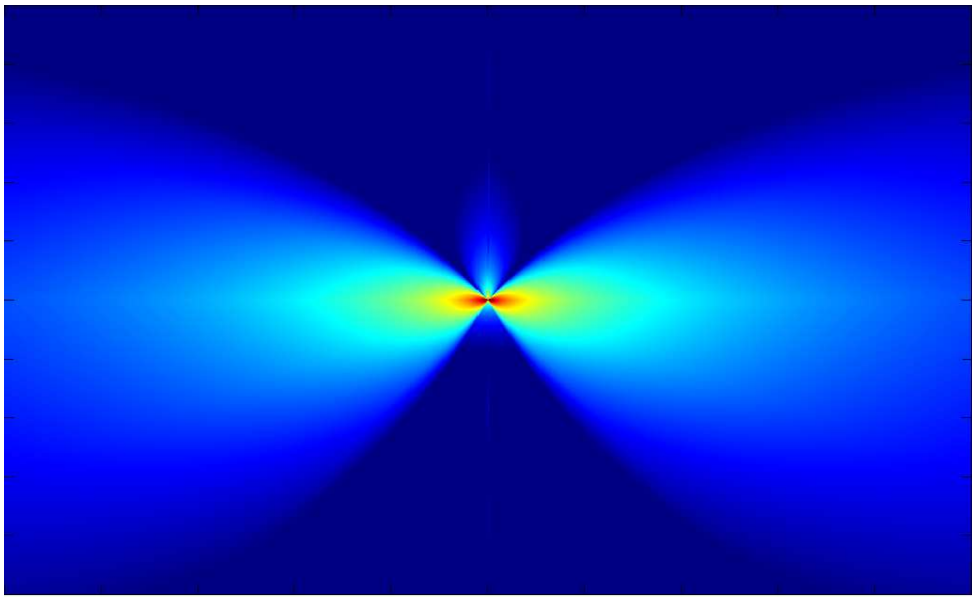


Figure 21 ITC ITC-2010 @ 2 kHz

3.5.3 ITC-2015 @ 1.8,2.8 kHz

3.5.3.1 Datasheet: Datasheet

3.5.3.2 SPL:

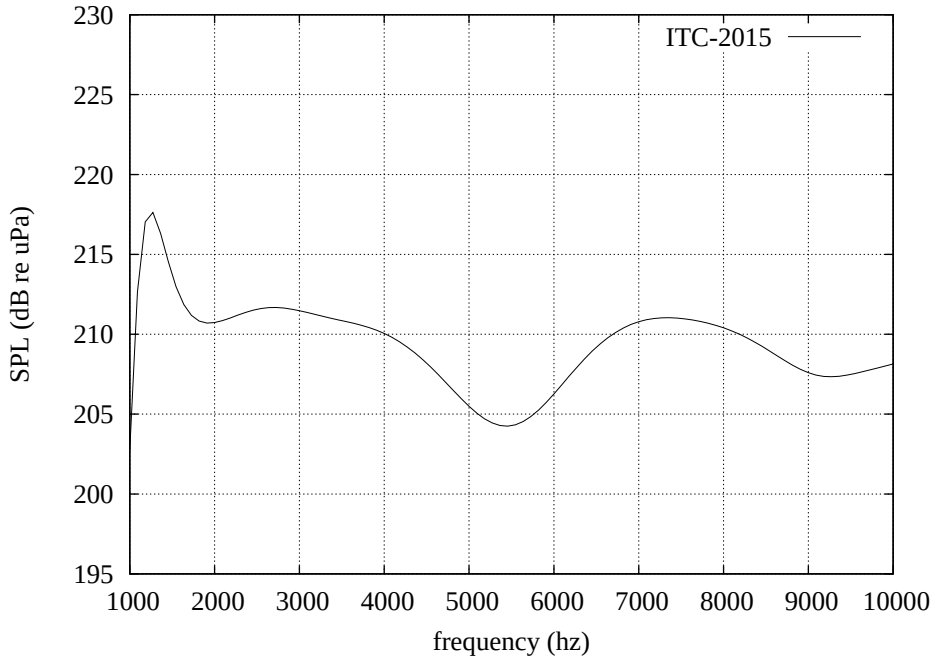


Figure 22 ITC ITC-2015 SPL

3.5.3.3 Vertical beam pattern:

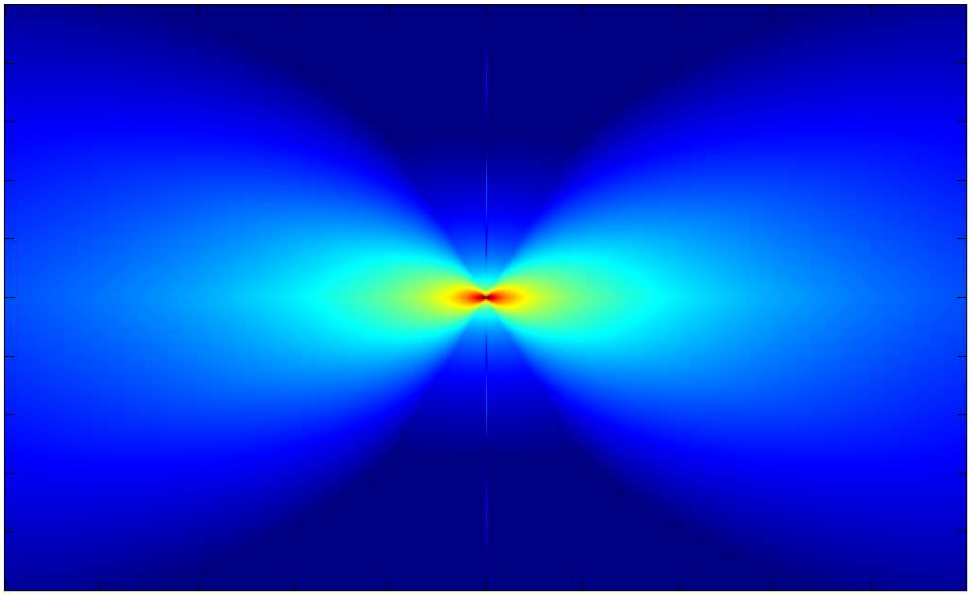


Figure 23 ITC ITC-2015 @ 2 kHz

3.5.4 ITC-2044 @ 8,14 kHz

3.5.4.1 Datasheet: Datasheet

3.5.4.2 SPL:

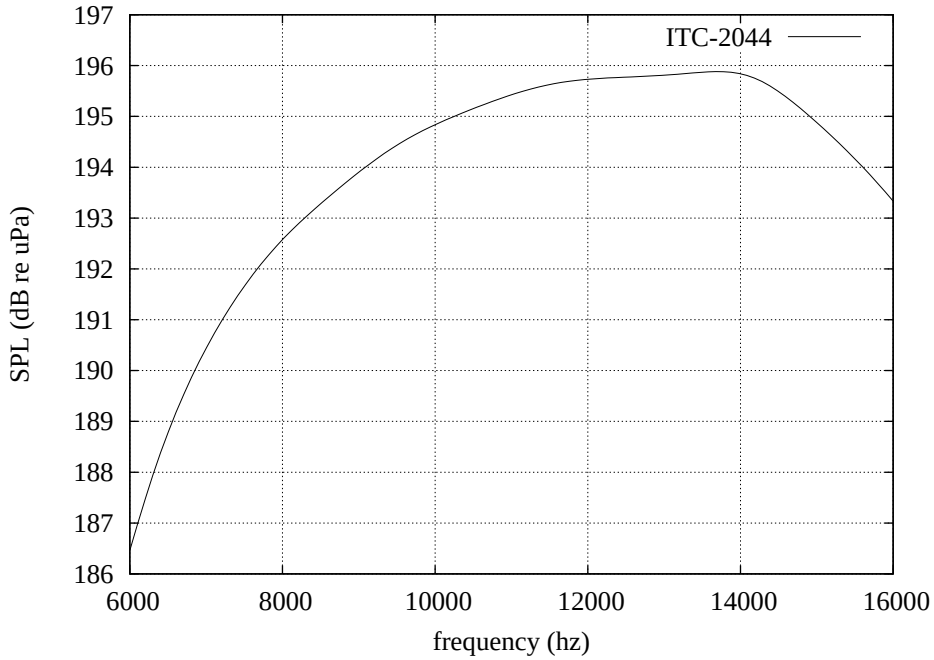


Figure 24 ITC ITC-2044 SPL

3.5.4.3 Vertical beam pattern:

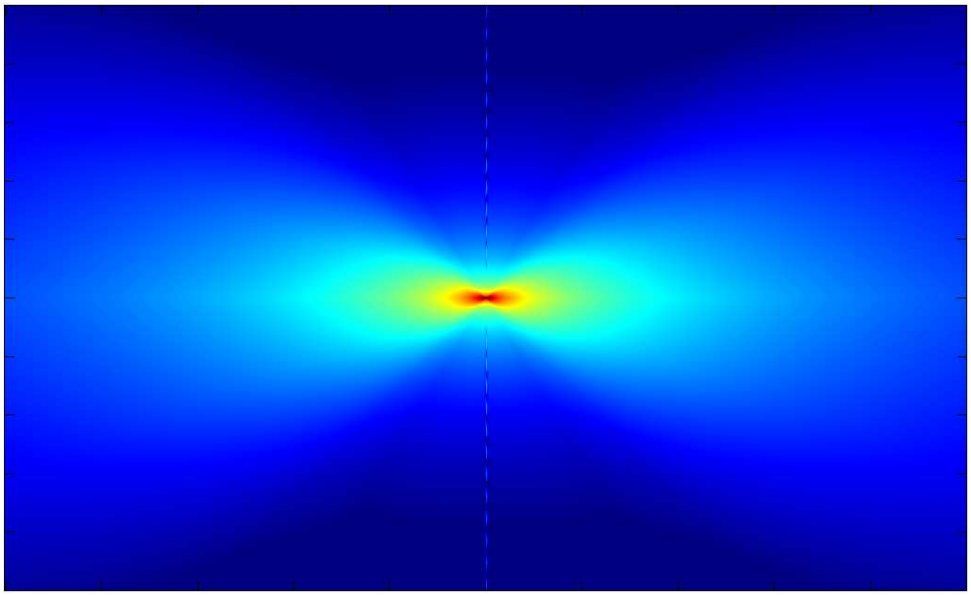


Figure 25 ITC ITC-2044 @ 10 kHz

3.5.5 ITC-2062 @ .44,1.4 kHz

3.5.5.1 Datasheet: Datasheet

3.5.5.2 SPL:

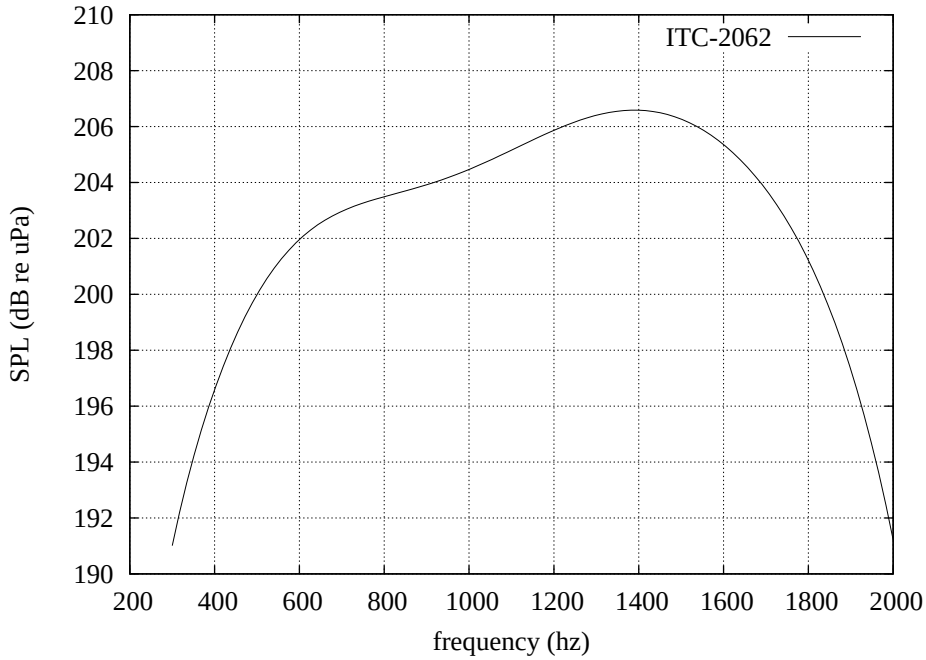


Figure 26 ITC ITC-2062 SPL

3.5.5.3 Vertical beam pattern:

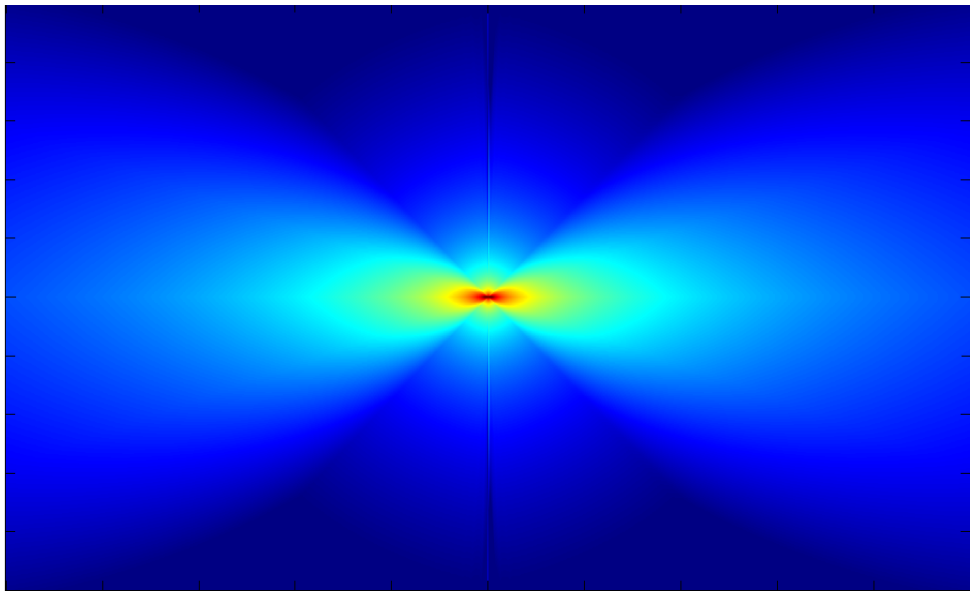


Figure 27 ITC ITC-2062 @ 0.44 kHz

3.5.6 ITC-3001 @ 17.5 kHz

3.5.6.1 Datasheet: Datasheet

3.5.6.2 SPL:

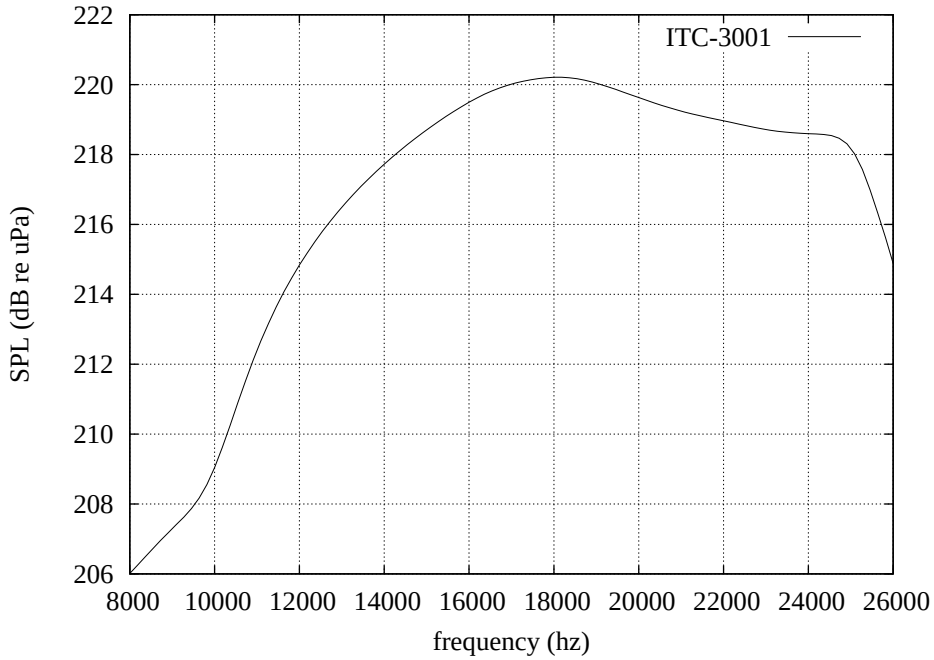


Figure 28 ITC ITC-3001 SPL

3.5.6.3 Vertical beam pattern:

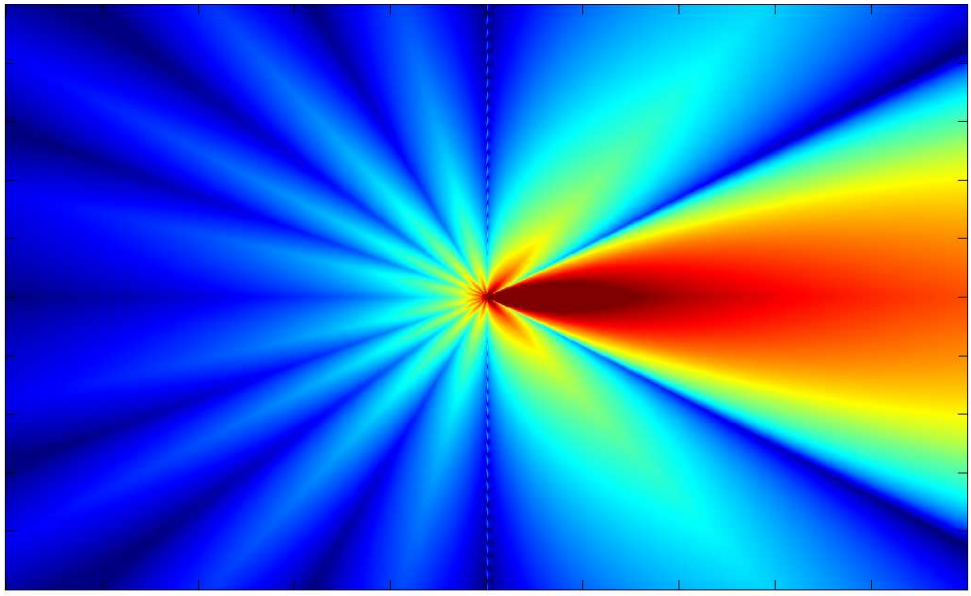


Figure 29 ITC ITC-3001 @ 17.5 kHz

3.5.7 ITC-3013 @ 12.5 kHz

3.5.7.1 Datasheet: Datasheet

3.5.7.2 SPL:

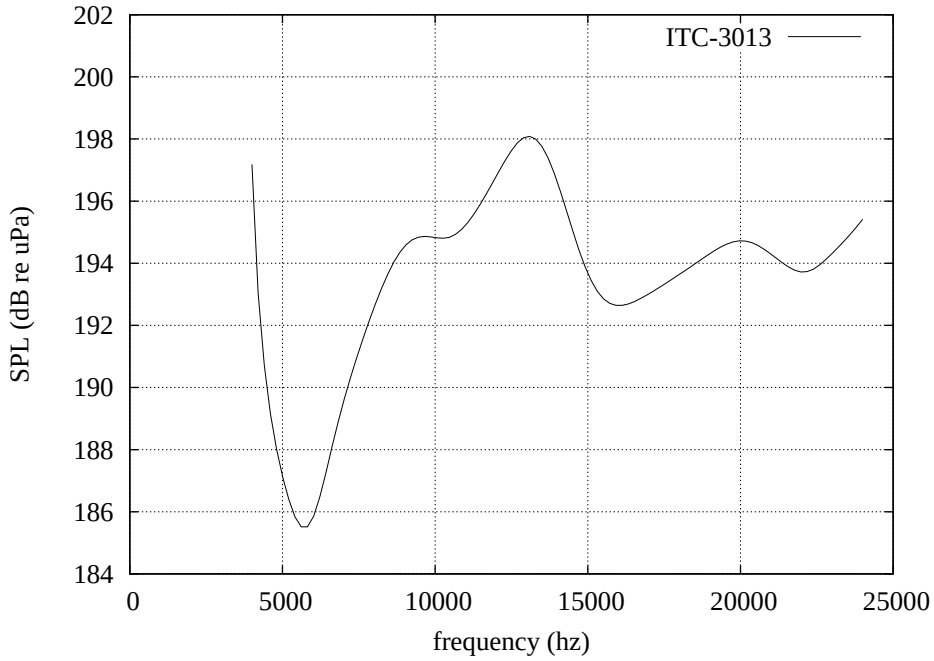


Figure 30 ITC ITC-3013 SPL

3.5.7.3 Vertical beam pattern:

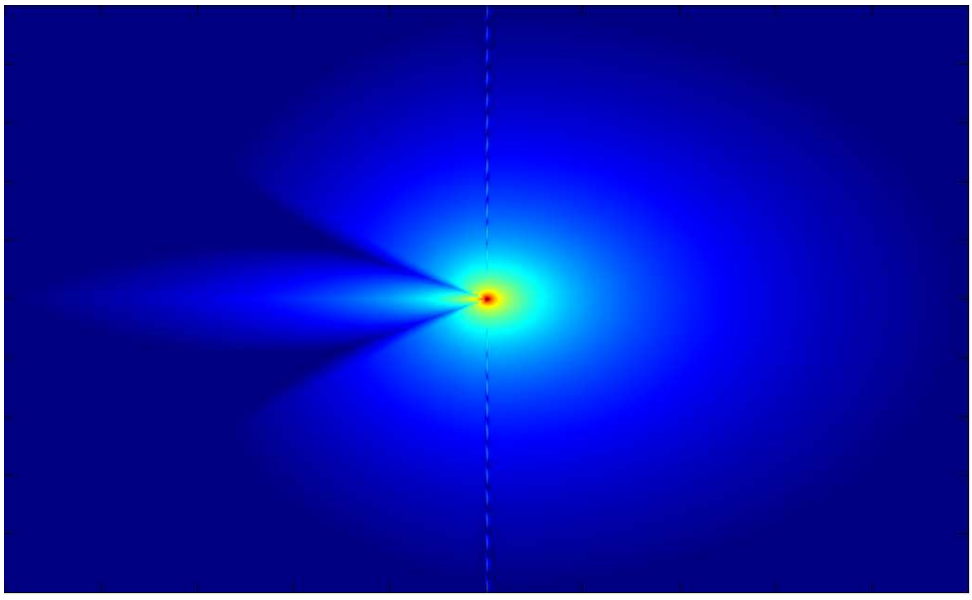


Figure 31 ITC ITC-3013 @ 10 kHz

3.5.8 ITC-3148 @ 12.5 kHz

3.5.8.1 Datasheet: Datasheet

3.5.8.2 SPL:

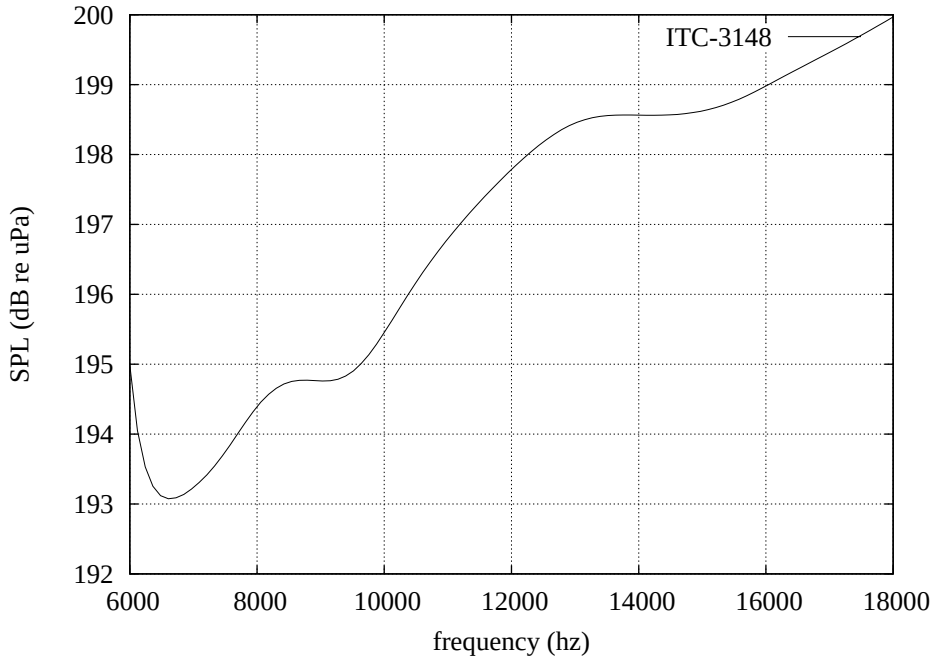


Figure 32 ITC ITC-3148 SPL

3.5.8.3 Vertical beam pattern:

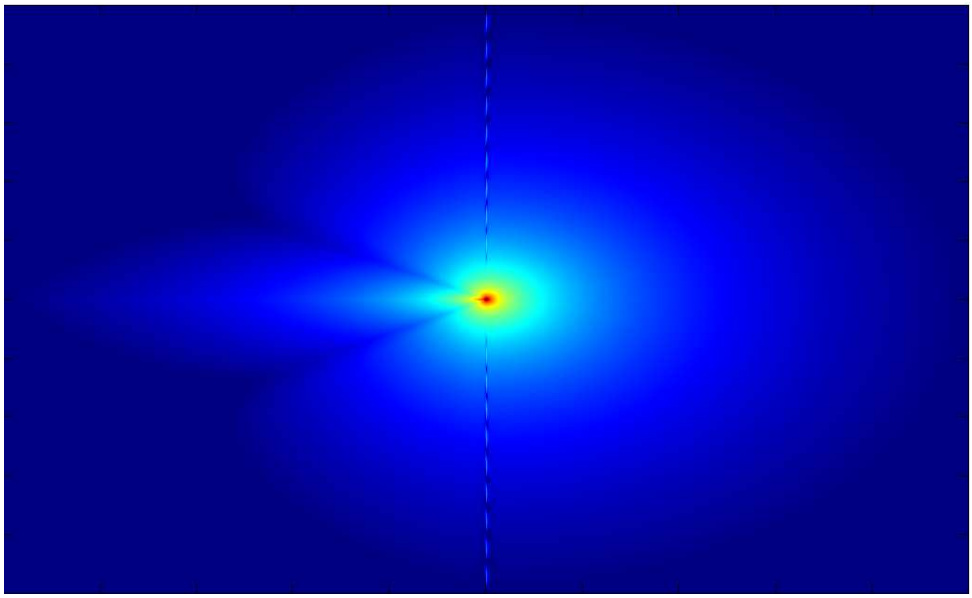


Figure 33 ITC ITC-3148 @ 8 kHz

3.5.9 ITC-3167 @ 11 kHz

3.5.9.1 Datasheet: Datasheet

3.5.9.2 SPL:

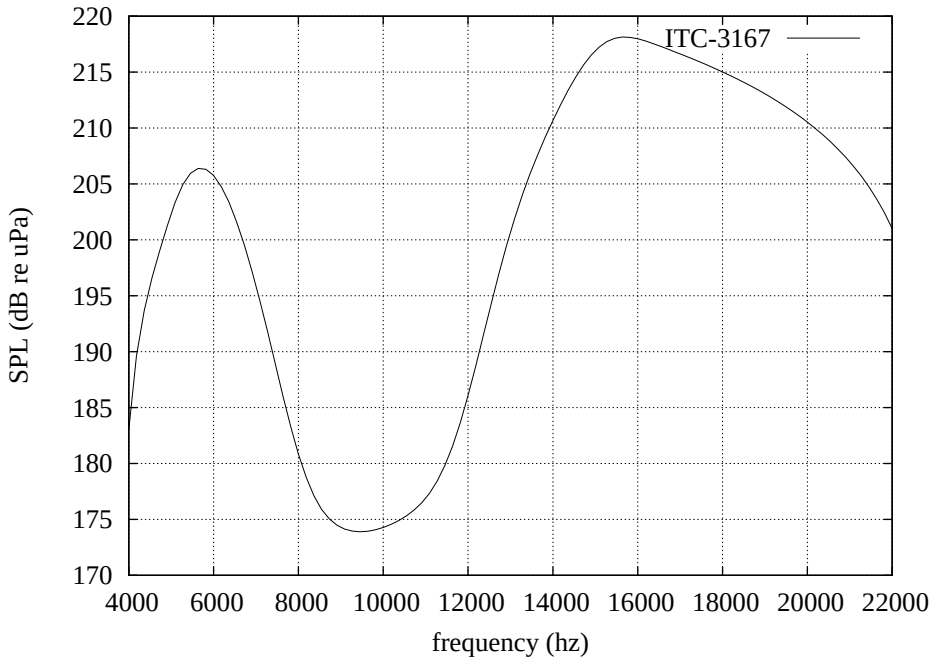


Figure 34 ITC ITC-3167 SPL

3.5.9.3 Vertical beam pattern:

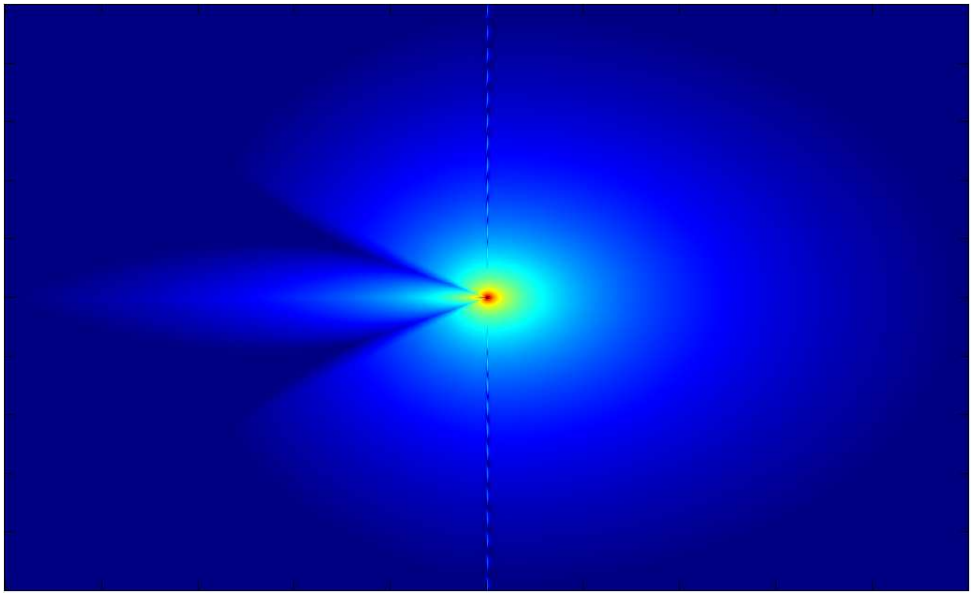


Figure 35 ITC ITC-3167 @ 10 kHz

3.6 ITT

3.6.1 SB31CT @ 10 kHz

3.6.1.1 Datasheet: Datasheet

3.6.1.2 SPL:

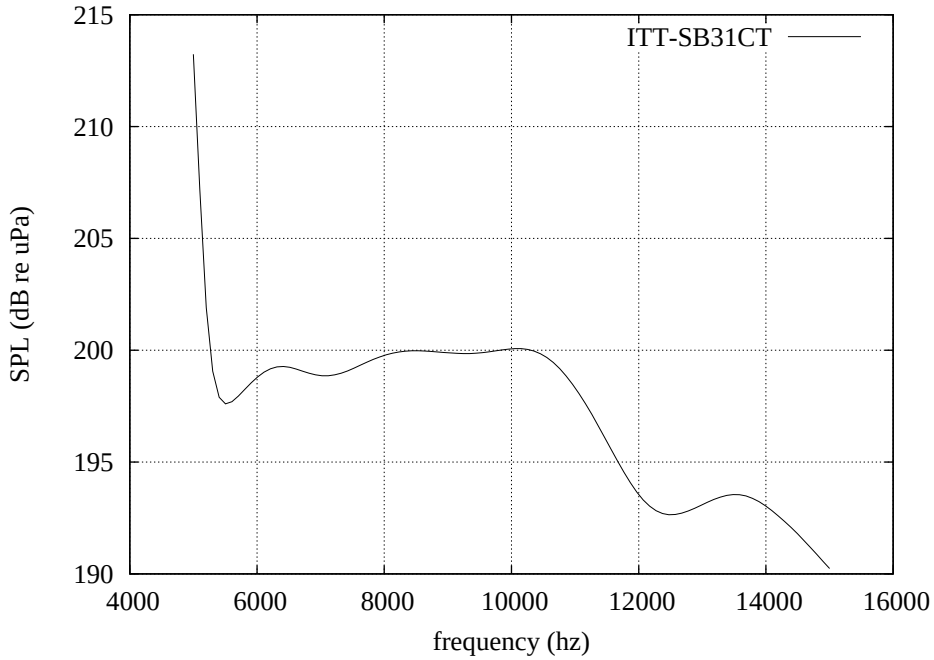


Figure 36 ITT SB31CT SPL

3.6.1.3 Vertical beam pattern:

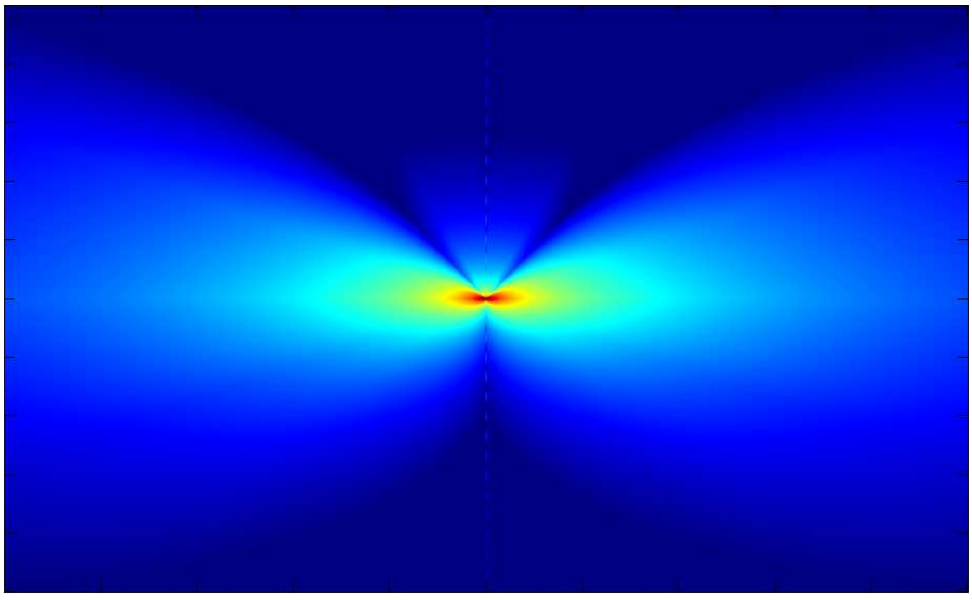


Figure 37 ITT BSB31CT @ 10 kHz

3.7 Neptune Sonar Limited

NOTE: vertical beam patterns inferred from datasheets generic description. All Neptune transducers report only one value of conductance, SPL calculations could be inaccurate

3.7.1 T54 @ 13 kHz

3.7.1.1 Datasheet: Datasheet

3.7.1.2 SPL:

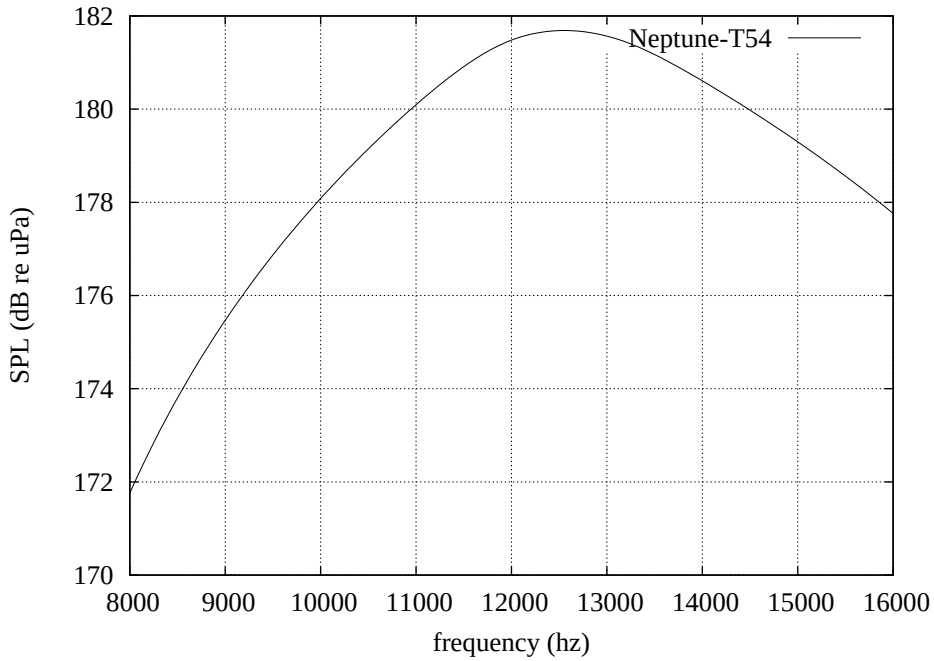


Figure 38 Neptune T54 SPL

3.7.1.3 Vertical beam pattern:

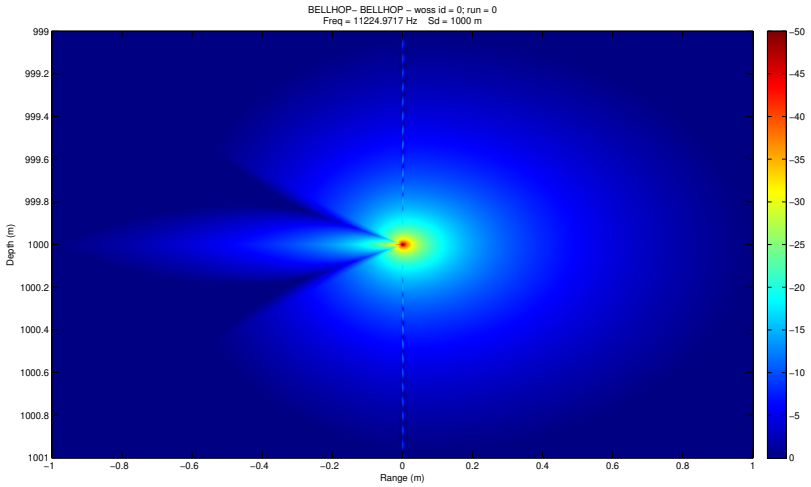


Figure 39 Neptune T54 @ 13 kHz

3.7.2 T186 @ 17 kHz

3.7.2.1 Datasheet: Datasheet

3.7.2.2 SPL:

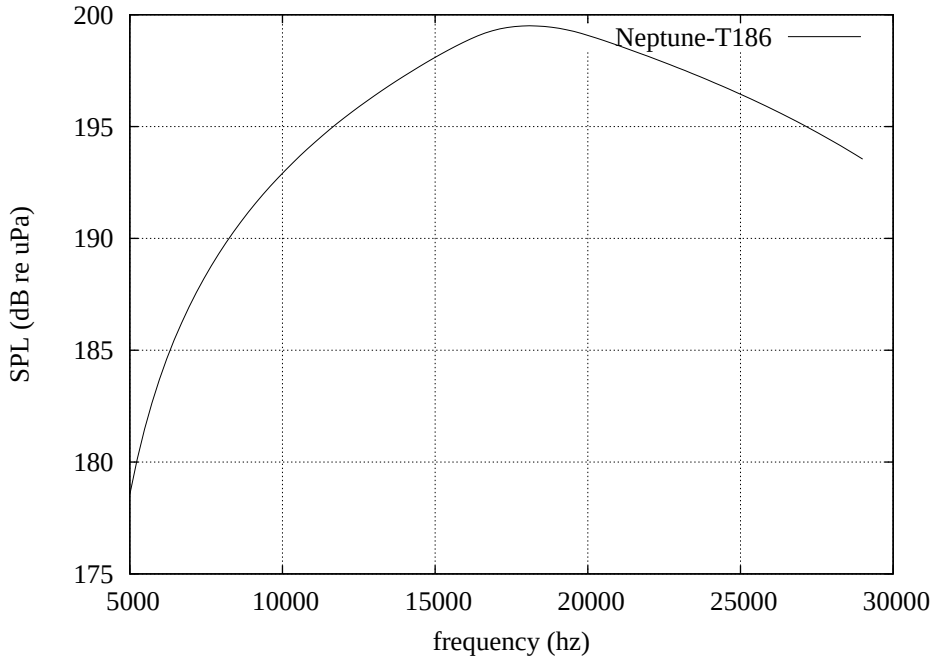


Figure 40 Neptune T186 SPL

3.7.2.3 Vertical beam pattern:

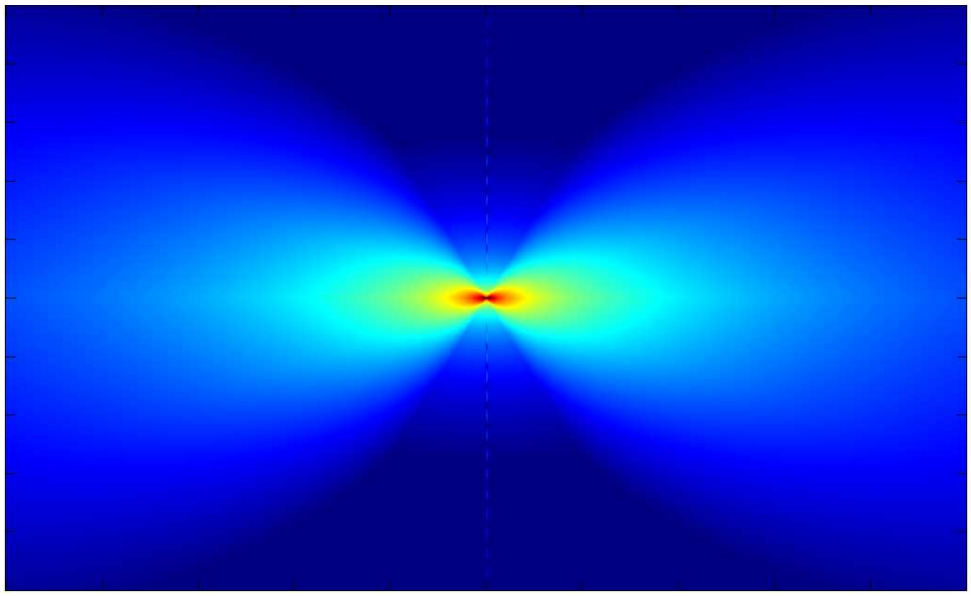


Figure 41 Neptune T186 @ 17 kHz

3.7.3 T204 @ 54 kHz

3.7.3.1 Datasheet: Datasheet

3.7.3.2 SPL:

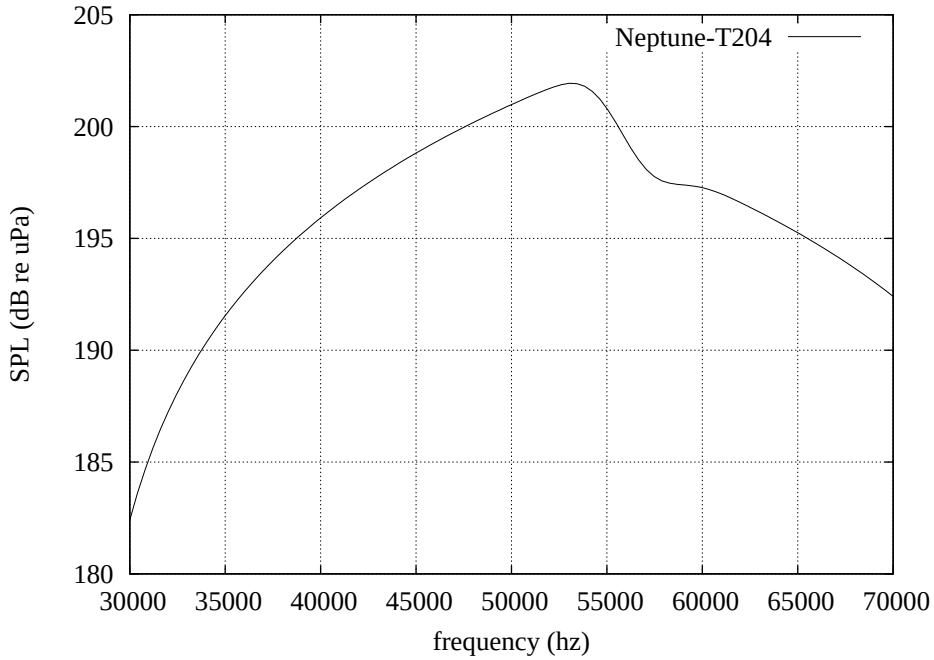


Figure 42 Neptune T204 SPL

3.7.3.3 Vertical beam pattern:

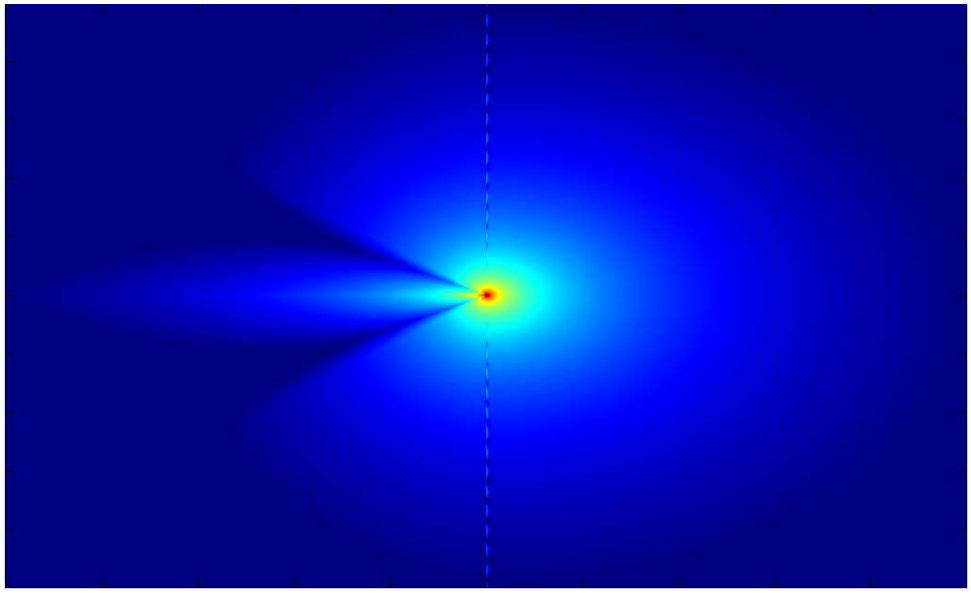


Figure 43 Neptune T204 @ 54 kHz

3.7.4 T216 @ 58 kHz

3.7.4.1 Datasheet: Datasheet

3.7.4.2 SPL:

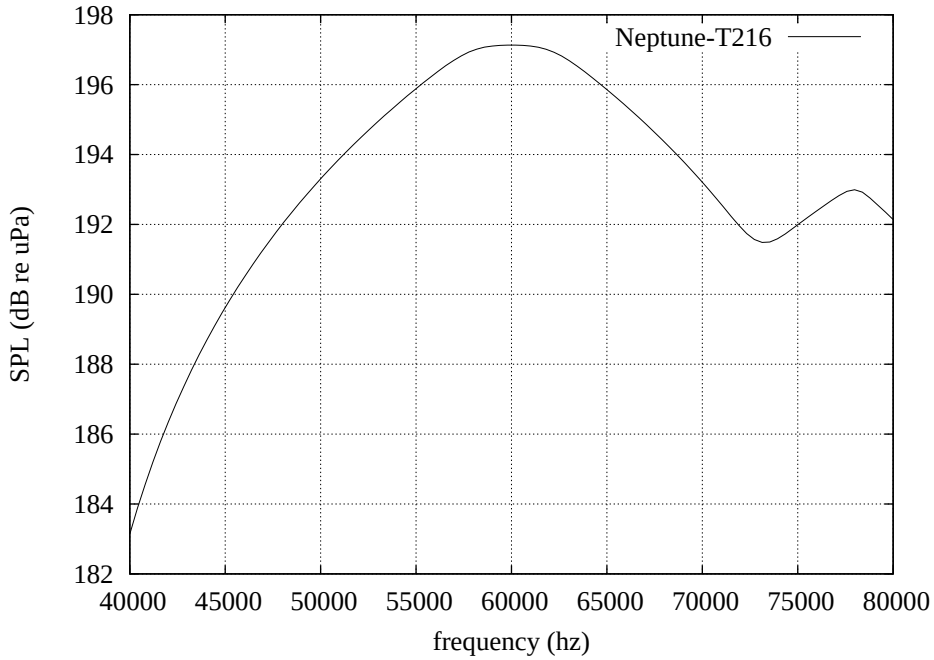


Figure 44 Neptune T216 SPL

3.7.4.3 Vertical beam pattern:

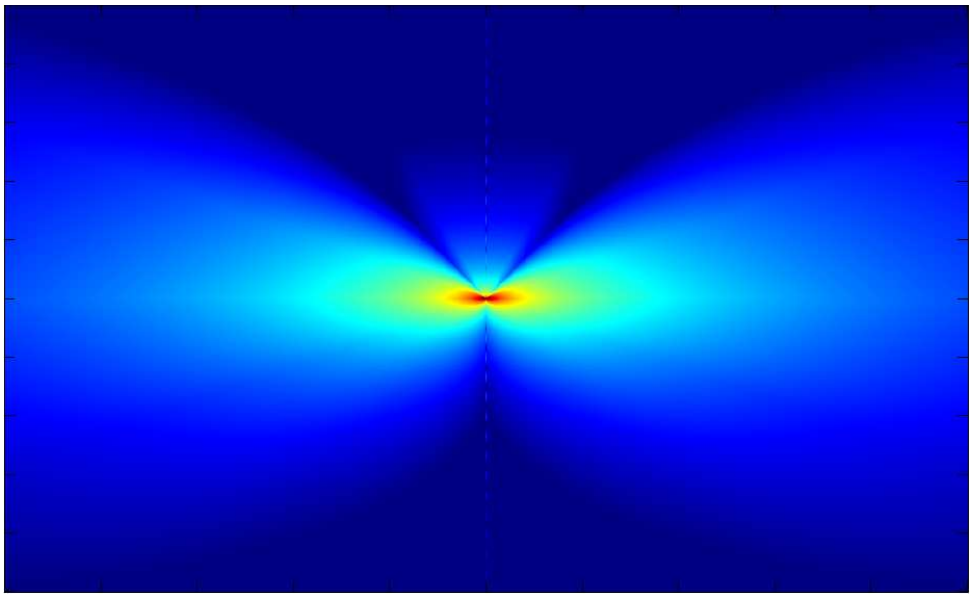


Figure 45 Neptune T216 @ 58 kHz

3.7.5 T217 @ 25 kHz

3.7.5.1 Datasheet: Datasheet

3.7.5.2 SPL:

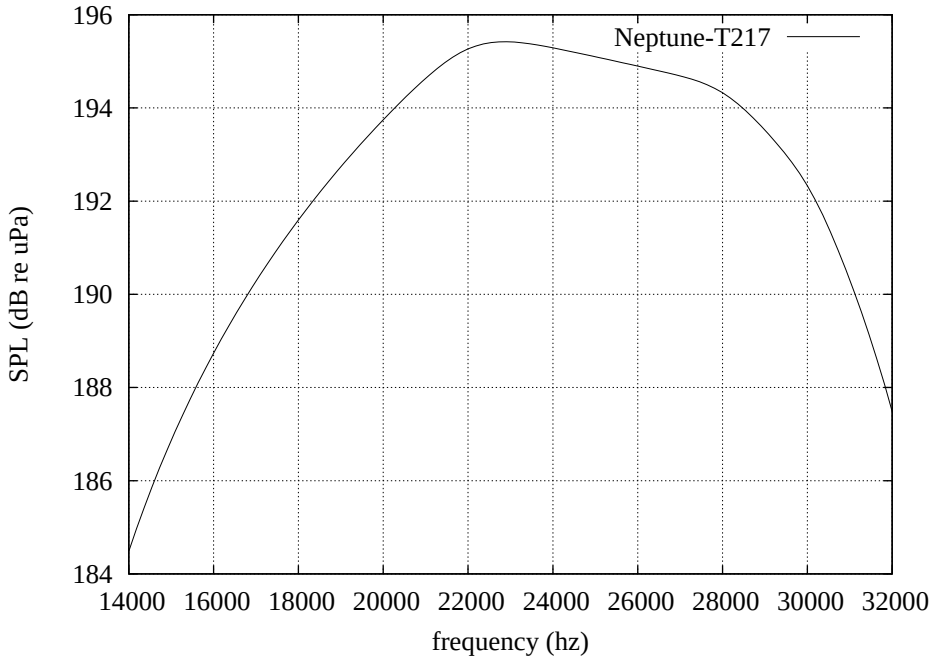


Figure 46 Neptune T217 SPL

3.7.5.3 Vertical beam pattern:

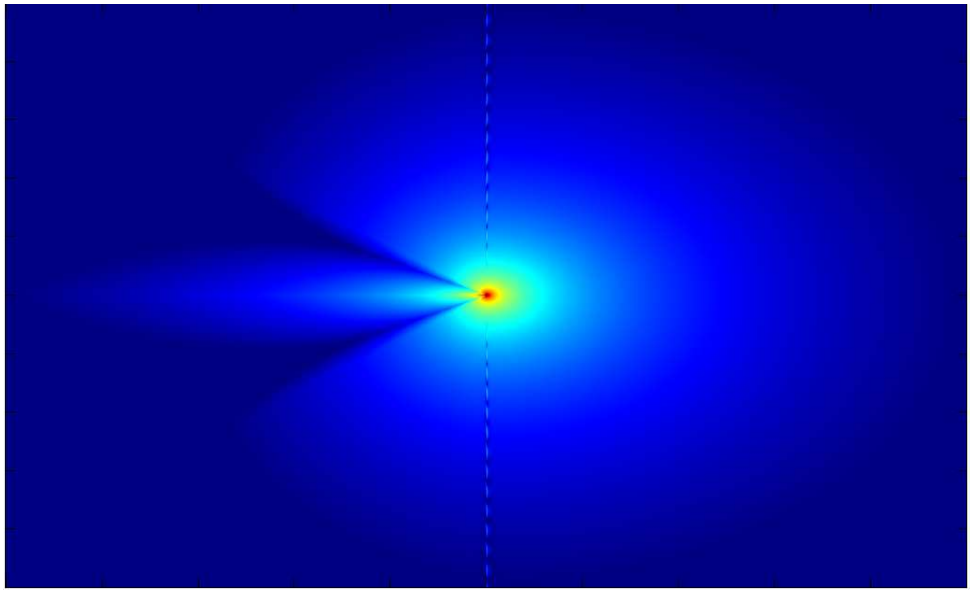


Figure 47 Neptune T217 @ 25 kHz

3.7.6 T218 @ 22 kHz

3.7.6.1 Datasheet: Datasheet

3.7.6.2 SPL:

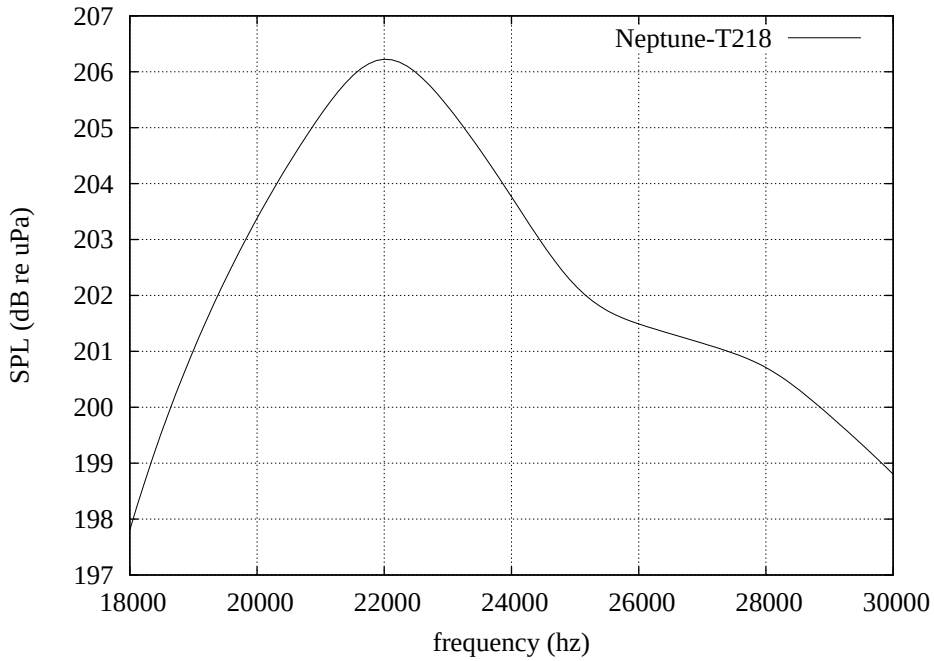


Figure 48 Neptune T218 SPL

3.7.6.3 Vertical beam pattern:

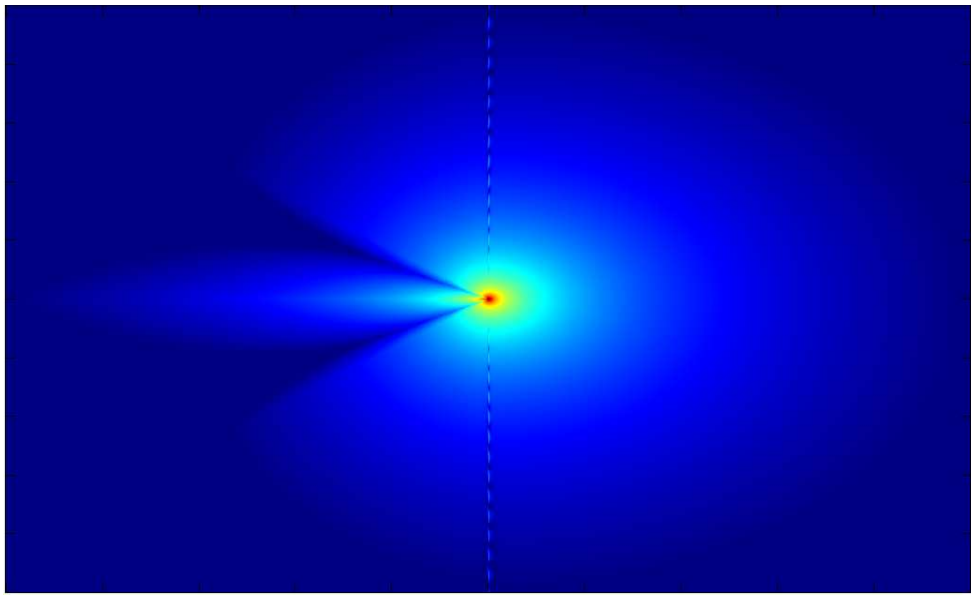


Figure 49 Neptune T218 @ 22 kHz

3.8 RESON

3.8.1 TC1026 @ 36 kHz

3.8.1.1 Datasheet: Datasheet

3.8.1.2 SPL:

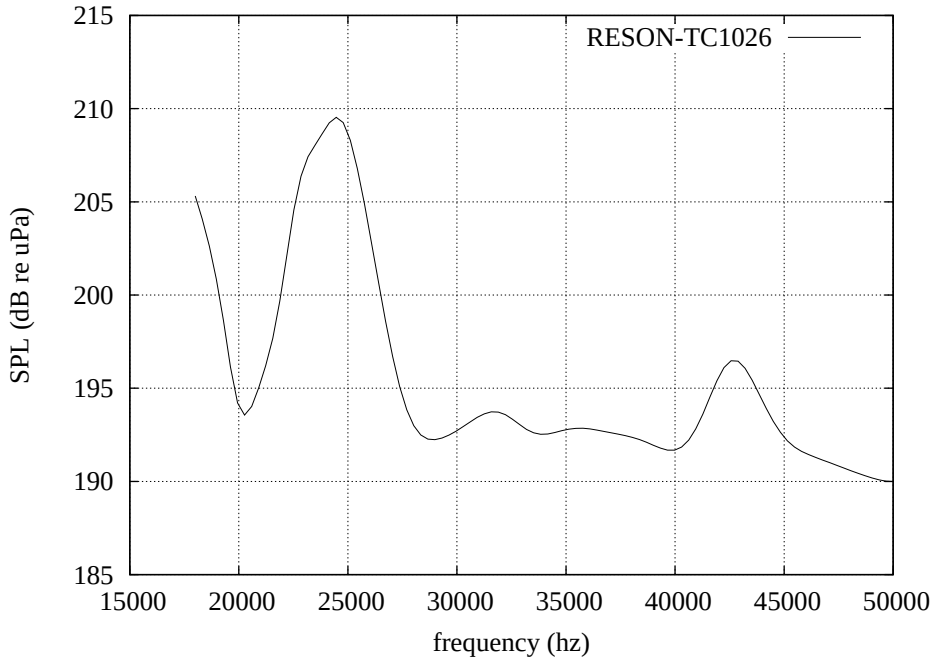


Figure 50 RESON TC1026 SPL

3.8.1.3 Vertical beam pattern:

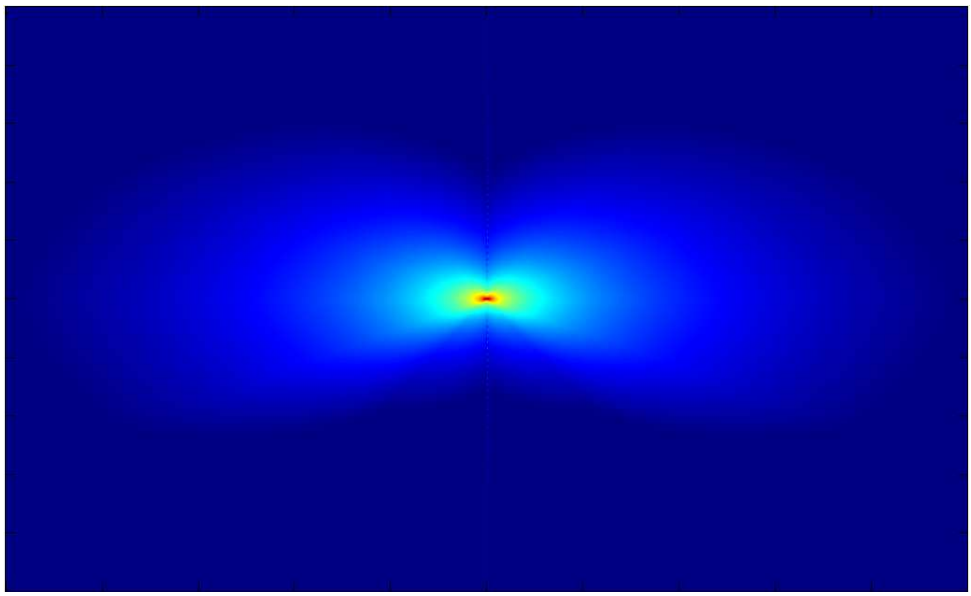


Figure 51 RESON TC1026 @ 36 kHz

4 New NS-Miracle classes

WOSS introduces a new Packet header struct hdr_woss, WossChannelModule, WossMPropagation and Woss↩ MInterferenceMIV classes for channel computations and interference calculations.

It also sports a ChannelEstimator, ChEstimatorPlugIn, ClMsgChannelEstimation for channel estimation purposes.

It adds support for positioning with WossPosition and for waypoint mobility with WossWpPosition.

Finally it provides the WossMPhyBpsk class for BPSK modulation with power control based on channel estimations.

5 Altimetry models

WOSS allows the user to create and simulate sea wave models. The altimetry framework supports time evolution.

5.1 Flat Model

The basic model is the Flat Altimetry model: it represents a flat sea surface.

5.2 Bretschneider Model

The Bretschneider Altimetry class implements the Bretschneider (ITTC two parameters) wave spectrum *[1]*.

Its input parameters are:

- *characteristic wave height [m]* - ***H*** -
- *wave period [s]* - ***T*** -

Additional parameters are:

- *range [m]*
- *range steps*
- *scenario depth [m]*

The latter are automatically set by the WOSS framework.

5.2.1 Examples:

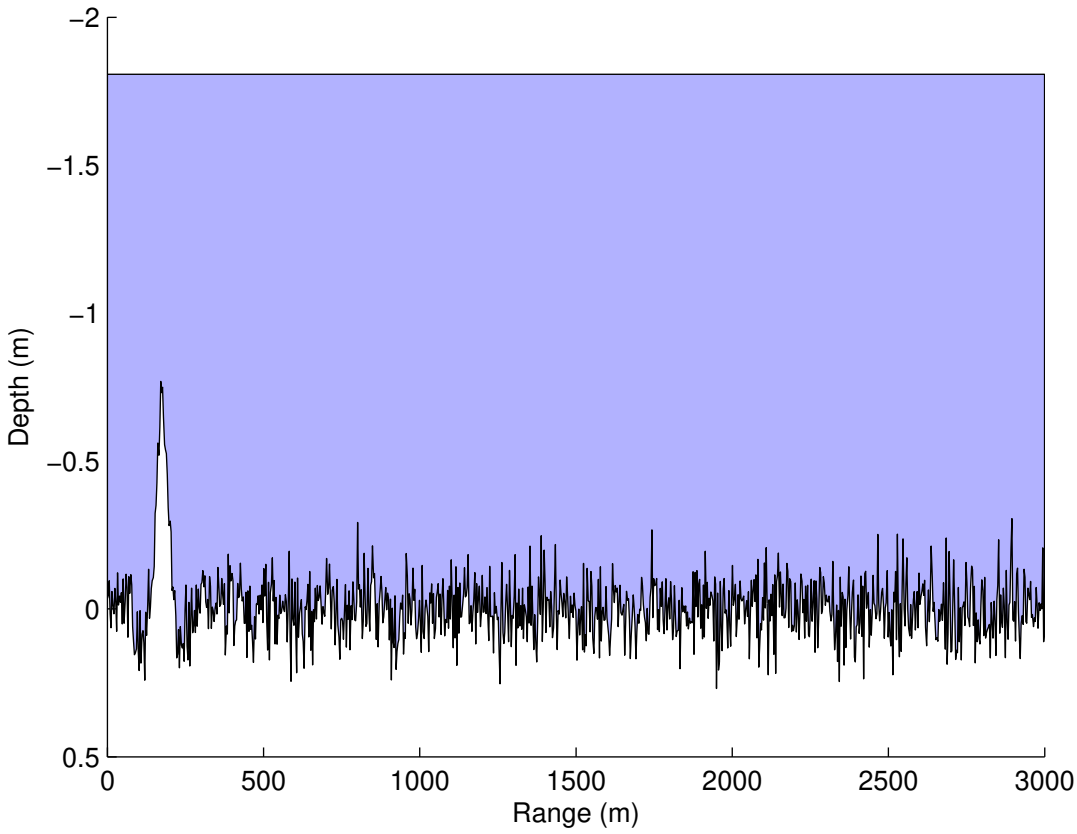


Figure 52 Example of a Bretschneider spectrum with T=1 and H=1

5.2.2 References:

1. G. J. Komen et al., *Dynamics and modeling of ocean waves.* Cambridge University Press, 1994.

6 Time evolution modeling

Time evolution is supported by the following object/classes:

- woss::Woss - controls its evolution by querying a new SSP for the new time evolution quantum and by asking its woss::Altimetry to evolve.
- woss::Altimetry - performs an evolution of its mathematical model.
- woss::AltimBretschneider - generates a new realization of its statistical process.

Each class supports an independent time evolution quantum, measured in seconds, and the enabling/disabling of the feature, for the maximum flexibility.

The framework controls the time evolution through the following classes:

- woss::WossManager - the class now supports queries for a specific woss::Time object or for a specific number of seconds after the start of the simulation (woss::SimTime).
- woss::WossController - for each pair of tx-rx woss::Location the user can set a different woss::SimTime object.
- woss::WossDbManager, woss::WossDb - all databases now supports queries for a specific woss::Time object
- woss::CustomDataTimeContainer - A new template for custom db data has been introduced to properly support time evolution. This is best used for time dependent custom SSP data, as each returned value is an linear interpolation of the SSP related to the two most close Time values.

See also

For a clarifying example please check out the related tcl file in **./samples/** directory.

7 Installation

7.1 PLEASE NOTE:

Note

No super-user credentials are needed for any of these installations.

WOSS now supports the installation as a stand alone library (e.g. with no NS2 and NS-Miracle support). This installation method is useful if you require the library to work in any project that doesn't need NS2/NS-Miracle libraries and related source code.

7.2 Compatibilities

The current WOSS version has the following compatibilities:

- **Acoustic Toolbox**: May 2023
- **HDF5 library**: v1.14.2
- **NetCDF C library**: v4.9.2
- **NetCDF4 C++ library**: v4.3.1
- **GEBCO databases**: 1D and 2D, 2008, 2014, 2019, 2020, 2022, 2023

7.3 Requirements

- Download the **recommended** `Acoustic Toolbox Library` and follow installation instructions. The directory path that contains the binaries should be in the $PATH environment.
- woss::BellhopArrSyntax::BELLHOP_CREATOR_ARR_FILE_SYNTAX_1 is compliant with Bellhop arr file syntax of the acoustic toolbox library up until the 16 Aug 2016 version.
- woss::BellhopArrSyntax::BELLHOP_CREATOR_ARR_FILE_SYNTAX_2 is compliant with Bellhop arr file syntax of the acoustic toolbox library >= 31 March 2019 version. This is the new factory value.

See also

woss::BellhopCreator::setBellhopArrSyntax or tcl binded value bellhop_arr_syntax.

- **In order to use the latest GEBCO 15 seconds of arc databases, NetCDF4 + HDF5 support had to be added to the framework.** If you wish to use NetCDF classes and databases, download the recommended `HDF5 library` `NetCDF C library` and `NetCDF C++ library`;
  All libraries have to be built **with support for dynamic libraries** and with **NetCDF4 support**.
  Please refer to NetCDF documentation for comprehensive set of installation instructions of the C and C++ libraries. Please note that the `--prefix=<optional_netcdf4_install_path>` option is strongly suggested, and should be the same for HDF5, NetCDF-C and NetCDF-C++4.
  The previous NetCDF legacy installation instructions are still supported by are obsolete and not recommended.
  A simple installation script is given here as example:
  - install the recommended HDF5 library with these exact steps:
    * `./configure --enable-shared --prefix=<optional_netcdf4_install_path>`
    * `make`
    * `make check`
    * should any test fail, based on the test severity you could still try to continue with the installation by issuing `make install`
  - install the recommended NETCDF4 C library by passing the common installation path in both CPPFLAGS and LDFLAGS:
    * `./configure --prefix=<optional_netcdf4_install_path> --enable-netcdf-4 --enable-shared --disable-dap --disable-byterange CPPFLAGS="$CPPFLAGS -I<optional_netcdf4_install_path>/include" LDFLAGS="$LDFLAGS -L<optional_netcdf4_install_path>/lib"`
    * `make`
    * `make check`
    * should any test fail, based on the test severity you could still try to continue with the installation by issuing `make install`
  - install the recommended NETCDF4 C++ library by passing the common installation path in both CPPFLAGS and LDFLAGS:
    * `./configure --prefix=<optional_netcdf4_install_path> --enable-shared CPPFLAGS="$CPPFLAGS -I<optional_netcdf4_install_path>/include" LDFLAGS="$LDFLAGS -L<optional_netcdf4_install_path>/lib"`
    * `make`
    * `make check`
    * `make install`
- If you wish to add NS-Miracle support, download the **latest** version of NS-Miracle from the `official git repository` :
  Read documentation provided in the above link for installation instructions.
- If you wish to use WOSS with world environmental data, download and extract the latest WOSS databases. Download the `GEBCO 2023 2D Fifteen seconds zip archive` or any of the `previous data sets` and put them in the same directory.

7.4 How to install WOSS library

1. extract the compressed file in a directory of your choice and cd into that directory;
2. run `./autogen.sh`
3. run `./configure` with the following options:
   - `--with-ns-allinone=<ns2-allinone_path>` ***optional**, needed if you want to compile the library for NS2 and NSMIRACLE.*

See also

Examples to avoid gcc-13 error compilation.

- `--with-nsmiracle=<ns-miracle_path>` ***optional****, needed if you want to compile the library for NS2 and NSMIRACLE.*
  See also

  Examples to avoid gcc-13 error compilation.
- `--with-netcdf4=<NetCDF4_install_path>` ***optional****, needed if you want to use NetCDF4 + HDF5 databases. (The path is the same mentioned in the Requirements section, if NetCDF4 was installed with no –prefix, the default path SHOULD be /usr/local)*
- `--with-pthread` *optional* but **recommended**
- `--prefix=<path_where_libraries_will_be_installed>`
  this path is optional and should be the same one used with NS-Miracle installation. If used, this path should be added to the environmental variable *LD_LIBRARY_PATH*. Please refer to NS-Miracle documentation for more info

4. run `make`
5. run `make check` to optionally perform any framework test
6. should any test fail, please enable its debug, rerun the test, collect the debug output file written `woss/tests/` directory and send a mail to `WOSS@guerra-tlc.com`
7. run `make install`

7.5 Examples

The following line would install WOSS with pthread, and NetCDF4, but with no NS2/NS-Miracle support (stand-alone library).

```
./configure --with-netcdf4=<NetCDF4_install_path> --with-pthread --prefix=<path↩
_where_libraries_will_be_installed>
```

The next line would install WOSS with NetCDF4, NS2 and NS-Miracle support and it removes the gcc-13 overloaded-virtual warning

```
CXXFLAGS="$CXXFLAGS -Wno-overloaded-virtual" ./configure --with-netcdf4=<Net↩
CDF4_install_path> --with-ns-allinone=<ns2-allinone_path> --with-nsmiracle=<ns-miracle↩
_path> --with-pthread --prefix=<path_where_libraries_will_be_installed>
```

8 Changelog

WOSS changelog:

PLEASE NOTE: WOSS is in its development stage, API changes will be limited at minimum, but this cannot be guaranteed.

- **v1.0.0**
  - release version
- **v1.0.1**
  - woss::Coord, woss::CoordZ : latitude, longitude and depth are now double value (instead of woss::PDouble)

- woss::ResPressureTxtDb, woss::ResTimeArrTxtDb : added a space_sampling feature
- woss::WossDbManager : the managing of custom enviromental data has been reworked, check doxygen documentation and basic_samples
- WossCbrModule class added

- **v.1.1.0**
  - World Ocean Atlas 2005 and 2009 SSP Databases : data from WOA2009 has been released; these files are fully compatible with the 2005 version code
  - woss::CustomDataContainer : bugs fixed
  - woss::Location: class for positioning added
  - WossPosition: new class added. WOSS now expect a WossPosition as Miracle Position class

    See also

    basic_samples for usage.
  - WossWpPosition : class has been reworked
  - woss::WossCreatorContainer : class has been added
  - woss::WossCreator, woss::BellhopCreator : interfaces expanded, they now handles custom values for woss::Location objects
  - woss::ACToolboxWoss : bugs fixed
  - woss::BellhopWoss : bugs fixed
  - woss::TimeReference, woss::TimeReferenceTcl : classes for providing WOSS a time reference
  - woss::RandomGenerator, woss::RandomGeneratorTcl : classes for providing WOSS a random numbers generators
  - WossRandomGenerator, WossRandomGeneratorTcl : hooks for NS2 random generators
  - woss::DefHandler : now can set and return a time reference and a random generator objects

    See also

    basic_samples for usage.
  - woss::SSP : class has been updated to work with woss::RandomGenerator

- **v.1.2.0**
  - woss::RandomGenerator : bug fixed
  - woss::WossManagerResDbMT : some bugs fixed in thread allocation
  - woss::ACToolboxWoss : bug fixed
  - woss::ResTimeArrBinDb, woss::ResTimeArrBinDbCreator, woss::ResPressureBinDb, woss::ResPressureBinDbCreator : classes for storing computed results in a binary file
  - ChannelEstimator, ChannelEstimatorPlugin : now works with MAC address instead of IP address to permit power control at MAC level. This now requires that only the MLL ARP table has to be filled instead of routing tables and ARP tables. Space sampling capability added
  - woss::Transducer : transducer definitions added
  - woss::TransducerHandler : transducer handler definitions added
  - woss::BellhopWoss : addded beam pattern capabilities
  - woss::BellhopCreator : added custom beam pattern capabilities
  - woss::Location : default comparison_distance value changed, added support for dinamic vertical beam pattern orientation
  - WossPosition, WossWpPosition : added support for dynamic vertical beam pattern orientation
  - UnderwaterMPhyBpsk : some tcl variables **RENAMED**

See also

UnderwaterMPhyBpsk, tcl samples

- UwMPhyBpskTransducer, WossMPhyBpsk : added support for woss::Transducer, to properly simulate SPL and energy consumption
- added beam pattern description of BTech, ITC, ITT, RESON transducers

  See also

  beam_pattern_page
- test_aloha_no_dbs.tcl, test_aloha_with_dbs.tcl : now channel estimation process properly works
- new tcl sample added

  See also

  samples subdirectory
- doxygen basic samples page removed, all the comments are inline within the tcl files
- **NOTE**: from version 1.2.0, the standard configuration of MIRACLE's MIV interface retrieves multiple arrivals from WOSS and computes interference using a Gaussian model or, equivalently, by summing all taps related to the interferers the same way as in Bellhop's "incoherent mode". In addition, if the interference level varies throughout the duration of a packet, the typically different error rates that result are separately accounted for in the computation of the overall packet error rate. This is the default version now.
  This change has been applied as part of the work for the CLAM project, funded by the European Commission, G.A. 258359.
  Credits to **Roberto Petroccia**, University of Rome, for helping with the implementation.

- **v.1.3.0**
  - WossChannelModule : *channel_eq_time_*, *channel_eq_snr_threshold_db_* and *channel_symbol_time_* introduced, *channel_time_resolution_* removed
    - the channel now coherently sums all taps at *channel_symbol_resolution_*
    - the attenuation threshold in db is calculated from *channel_eq_snr_threshold_db_*
    - the first tap that has txloss less or equal than the attenuation threshold is found. If the threshold is less than 0 then the threshold is set to 0, and the first tap is taken.
    - then all taps after the above tap + *channel_equalization_time_* are incoherently summed if *channel_eq_time_* == 0 no eq is done. if *channel_eq_time_* $<$ 0 then all taps are incoherently summed
    - all taps after *channel_eq_time_* are not processed

      See also

      WossChannelModule, tcl samples
  - Altimetry modeling has been introduced:
    - woss::Altimetry is the standard flat altimetry
    - woss::AltimBretschneider is the Bretschneider wave model.

      See also

      Altimetry models, woss::Altimetry, woss::AltimBretschneider, tcl samples
  - time evolution support has been added for the whole framework.
- **v.1.3.1**
  - TCL dependancies have been moved to WossPhy library, leaving main library libWoss free of any unwanted dependancy
- **v.1.3.2**

  - Minor bug fixes

- **v.1.3.3**
  - Minor bug fixes
- **v.1.3.4**
  - Minor bug fixes
- **v.1.3.5**
  - UwmStd library has been moved into NS-Miracle for a better compatibility with DESERT framework
  - Minor bug fixes
- **v.1.3.6**
  - compilation warnings removal
  - Minor bug fixes
- **v.1.3.7**
  - warnings system revised.
  - tcl warnings removed
  - bug fixes (thanks to Raúl Sáez Cañete)
- **v.1.3.8**
  - minor bug fixes
  - license changed to GPLv2
- **v.1.3.9**
  - minor bug fixes
  - syntax now compliant with C++11 standard
- **v.1.4.0**
  - minor bug fixes
- **v.1.5.0**
  - memory leak fixed (thanks to Raúl Sáez Cañete)
  - new bathymetry write mode introduced (thanks to Randall Plate)
  - new attenuation complex tap calculation introduced:
    * travel time phase is now properly used
    * a bellhop arr file syntax option has been introduced, in order to make WOSS work with both acoustic tool box >= 16 Aug 2016 and previous versions.
    * BE ADVISED THAT factory value is using the latest syntax (i.e. acoustic toolbox >= 16 Aug 2016)
- **v.1.5.1**
  - gcc-6 warning removal
- **v.1.6.0**
  - bug fix in volumetrical attenuation computations
  - new APIs introduced
- **v.1.6.1**
  - LICENSE revised
- **v.1.6.2**

  - bug fix in box_depth and box_range for ns2 tcl hooks

- **v.1.7.0**
  - bug fixing in WossManager
  - new feature in WossChannelModule

- **v.1.8.0**
  - new UTM CSV Database model added
  - minor fixes in WossChannelModule

- **v.1.9.0**
  - Added support for NetCDF4 and HDF5, which is required in order to open_ the latest GEBCO 2019 database.
  - A new bellhop arr file syntax option has been introduced, in order to be aligned with Acoustic Toolbox >= 31 March 2019.
  - BE ADVISED THAT factory value is using the latest syntax (i.e. acoustic toolbox >= 31 March 2019)
  - Credits to **Randall Plate** for helping with the implementation.
  - WossDb GEBCO has been refactored in order to support 2D netcdf format.
  - Bug fixes and improvements in woss::ACToolboxWoss and woss::BellhopWoss

- **v.1.10.0**
  - Added support for SSP NetCDF4 databases generated from WOA2013 dataset.
  - Fix for Bellhop quad SSP scenario.

- **v.1.11.0**
  - Added support for SSP NetCDF4 databases generated from WOA2018 and WOA2001 dataset.
  - Fix for woss::BellhopWoss ray angles env file write operation.
  - Added support for Bellhop latest shade file reader syntax.

- **v.1.12.0**
  - Added support for DECK41V2 databases in NetCDF4 format with revised data format.
  - Added support for GEBCO 2020 database.
  - Added support for autotools tests, installation procedure has been updated.
  - Fix for woss::BellhopWoss SSP max depth.

- **v.1.12.1**
  - Fixed issue with autotools distribution.
  - Added SSP truncate option to both woss::SSP and woss::BellhopWoss (thanks to Randall Plate).
  - Improved woss::CoordZ::getCoordZFromCartesianCoords() function (thanks to Randall Plate).

- **v.1.12.2**
  - Fixed issue with SSP::truncate c+11 syntax

- **v.1.12.3**
  - New geographical to cartesian conversion model, based on spherical, GRS80 and WGS84 models

- **v.1.12.4**
  - Added support for GEBCO 2022
  - Fixed TCL samples
  - Refactor of function member's static variables in order to guarantee re-entrancy

  - Fixed bug with WossDbManager
  - Added TCL debug capabilities to Altimetry classes
  - Refactored usage of streams' precision throughout the whole framework

- **v.1.12.5**
  - minor fixes to ns-Miracle classes
- **v.1.12.6**
  - fixed gcc-13 warnings
  - TCL examples updated to GEBCO 2023
  - removed warning in woss::BellhopWoss
  - changed URL of main WOSS website

9 Acknowledgements

This software has been developed by Federico Guerra and SIGNET lab, University of Padova, in collaboration with the NATO Centre for Maritime Research and Experimentation (http://www.cmre.nato.int ; E-mail: pao@cmre.nato.int), whose support is gratefully acknowledged.

10 Copyright Terms

10.0.1 ----------\n

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University of Padova (SIGNET lab) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

10.0.2 ----------\n

The source files containing NetCDF code refer to this BSD-like and GPL compatible copyright:

Copyright 1993-2008 University Corporation for Atmospheric Research/Unidata

Portions of this software were developed by the Unidata Program at the University Corporation for Atmospheric Research.

Access and use of this software shall impose the following obligations and understandings on the user. The user is granted the right, without any fee or cost, to use, copy, modify, alter, enhance and distribute this software, and any derivative works thereof, and its supporting documentation for any purpose whatsoever, provided that this entire notice appears in all copies of the software, derivative works and supporting documentation. Further, UCAR requests that the user credit UCAR/Unidata in any publications that result from the use of this software or in any product that includes this software, although this is not an obligation. The names UCAR and/or Unidata, however, may not be used in any advertising or publicity to endorse or promote any products or commercial entity unless specific written permission is obtained from UCAR/Unidata. The user also understands that UCAR/Unidata is not obligated to provide the user with any support, consulting, training or assistance of any kind with regard to the use, operation and performance of this software nor to provide the user with any updates, revisions, new versions or "bug fixes."

THIS SOFTWARE IS PROVIDED BY UCAR/UNIDATA "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UCAR/UNIDATA BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE ACCESS, USE OR PERFORMANCE OF THIS SOFTWARE.

10.0.3 ----------\n

Bellhop program is published under the following copyright:

Copyright (C) 2009 Michael B. Porter

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

10.0.4 ----------\n

TEOS-10 SSP formula code has been taken from has been taken from Gibbs SeaWater (GSW) Oceanographic Toolbox, which has published under the folllowing copyright:

The software is available from http://www.TEOS-10.org

11 License Terms

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

1. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change. b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your

work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

1. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

1. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
2. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
3. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

4. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

1. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
2. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

1. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

1. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

2. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and an idea of what it does. Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA. Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details. The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

12 Bug List

Class woss::SedimentNodules

Needs a better geoacoustic representation

Class woss::SedimentOrganic

No geoacoustic parameters avalaible

Class woss::SedimentRocks

Needs a better geoacoustic representation

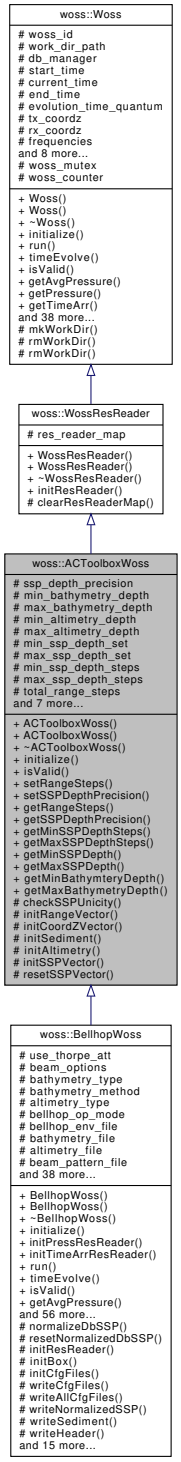
13 Class Documentation

13.1 woss::ACToolboxWoss Class Reference

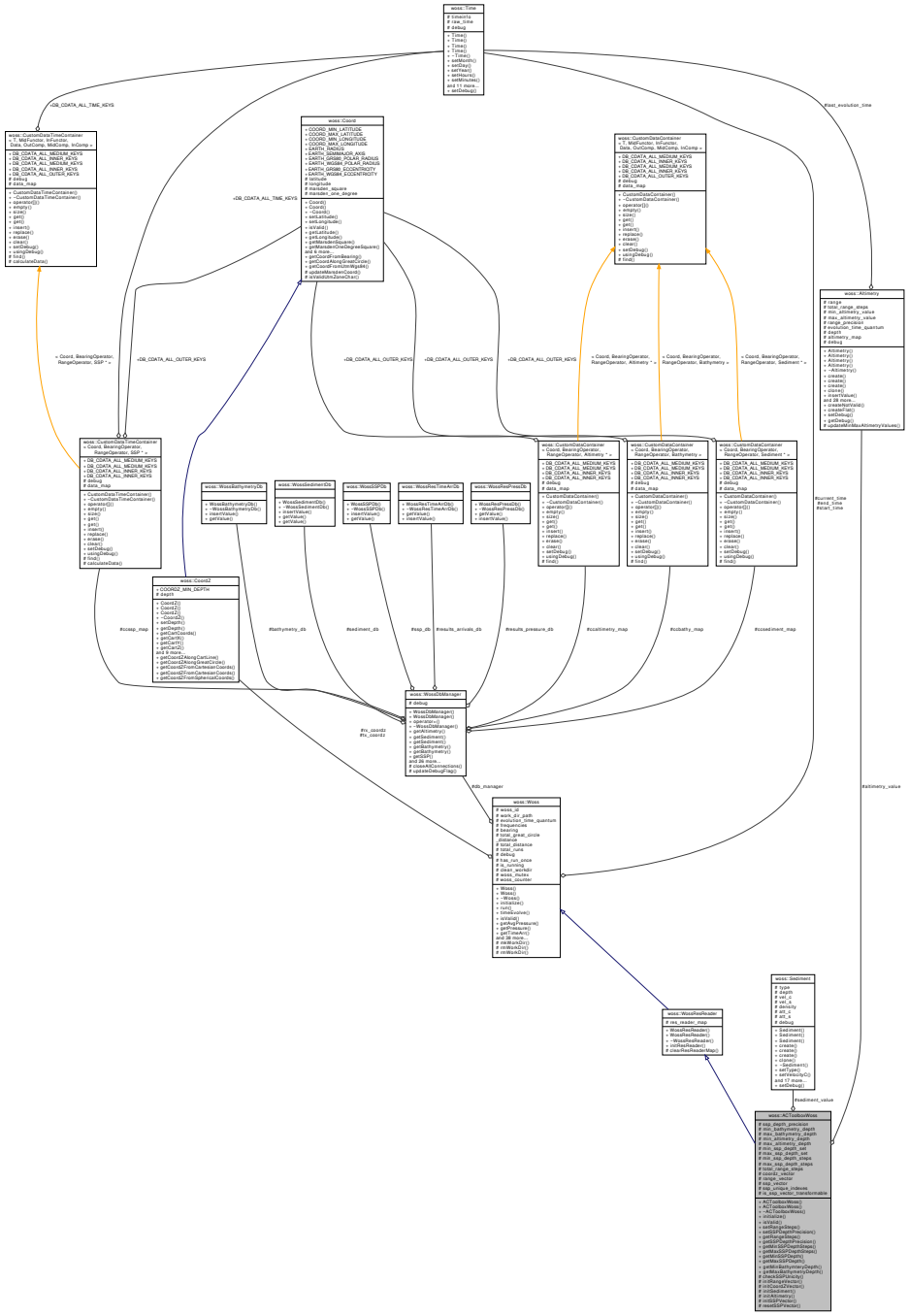
base class for implementing acoustic-toolbox channel simulators (Bellhop, Kraken, etc...)

`#include <ac-toolbox-woss.h>`

Inheritance diagram for woss::ACToolboxWoss:



Collaboration diagram for woss::ACToolboxWoss:



Public Member Functions

- ACToolboxWoss ()
- ACToolboxWoss (const CoordZ &tx, const CoordZ &rx, const Time &start_t, const Time &end_t, double start_freq, double end_freq, double freq_step)
- virtual ~ACToolboxWoss ()
- virtual bool initialize ()
- virtual bool isValid () const
- ACToolboxWoss & setRangeSteps (int steps)
- ACToolboxWoss & setSSPDepthPrecision (long double precision)

- int getRangeSteps () const
- long double getSSPDepthPrecision () const
- int getMinSSPDepthSteps () const
- int getMaxSSPDepthSteps () const
- double getMinSSPDepth () const
- double getMaxSSPDepth () const
- double getMinBathymteryDepth () const
- double getMaxBathymetryDepth () const

Protected Member Functions

- virtual bool checkSSPUnicity (SSP *&ptr)
- virtual bool initRangeVector ()
- virtual bool initCoordZVector ()
- virtual bool initSediment ()
- virtual bool initAltimetry ()
- virtual bool initSSPVector ()
- virtual void resetSSPVector ()

Protected Attributes

- long double ssp_depth_precision
- double min_bathymetry_depth
- double max_bathymetry_depth
- double min_altimetry_depth
- double max_altimetry_depth
- ::std::set< double > min_ssp_depth_set
- ::std::set< double > max_ssp_depth_set
- int min_ssp_depth_steps
- int max_ssp_depth_steps
- int total_range_steps
- CoordZVector coordz_vector
- RangeVector range_vector
- SSPVector ssp_vector
- ::std::set< int > ssp_unique_indexes
- Sediment * sediment_value
- Altimetry * altimetry_value
- bool is_ssp_vector_transformable

Additional Inherited Members

13.1.1 Detailed Description

base class for implementing acoustic-toolbox channel simulators (Bellhop, Kraken, etc...)

Class ACToolboxWoss should be the base class of any Woss that wants to implement any acoustic-toolbox channel simulator (Bellhop, Kraken, Fields, Sparc, etc...)

13.1.2 Constructor & Destructor Documentation

13.1.2.1 ACToolboxWoss() [1/2] ACToolboxWoss::ACToolboxWoss ( )

ACToolboxWoss default constructor. Default constructed objects are not valid

13.1.2.2 ACToolboxWoss() [2/2] ACToolboxWoss::ACToolboxWoss (

```
const CoordZ & tx,
const CoordZ & rx,
const Time & start_t,
const Time & end_t,
double start_freq,
double end_freq,
double freq_step )
```

ACToolboxWoss constructor

Parameters

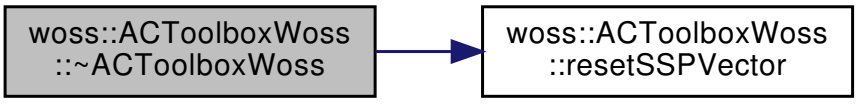
tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_t	const reference to a valid Time object for SSP's averaging purposes (start date time)
end_t	const reference to a valid Time object for SSP's averaging purposes (end date time)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
freq_step	frequency step [Hz]

13.1.2.3 ~ACToolboxWoss() ACToolboxWoss::~ACToolboxWoss ( ) [virtual]

Destructor. It properly delete all pointers involved

References altimetry_value, resetSSPVector(), and sediment_value.

Here is the call graph for this function:



13.1.3 Member Function Documentation

13.1.3.1 checkSSPUnicity() bool ACToolboxWoss::checkSSPUnicity (

```
SSP *& ptr ) [protected], [virtual]
```

Checks if the given SSP is not equal to previous values

Parameters

value	pointer to a valid SSP object

Returns

true if input is unique, *false* otherwise

References ssp_unique_indexes, and ssp_vector.

Referenced by initSSPVector().

13.1.3.2 getMaxBathymetryDepth() `double woss::ACToolboxWoss::getMaxBathymetryDepth ( ) const [inline]`

Gets the maximum depth of bathymetry in use

Returns

maximum bathymetry depth [m]

References max_bathymetry_depth.

13.1.3.3 getMaxSSPDepth() `double woss::ACToolboxWoss::getMaxSSPDepth ( ) const [inline]`

Gets the maximum depth of all SSP currently in use

Returns

maximum SSP depth [m]

References max_ssp_depth_set.

13.1.3.4 getMaxSSPDepthSteps() `int woss::ACToolboxWoss::getMaxSSPDepthSteps ( ) const [inline]`

Gets the maximum number depth steps of all SSP currently in use

Returns

maximum SSP depth steps

References max_ssp_depth_steps.

13.1.3.5 getMinBathymteryDepth() `double woss::ACToolboxWoss::getMinBathymteryDepth ( ) const [inline]`

Gets the minimum depth of bathymetry in use

Returns

mimimum bathymetry depth [m]

References min_bathymetry_depth.

13.1.3.6 getMinSSPDepth() `double woss::ACToolboxWoss::getMinSSPDepth ( ) const [inline]`

Gets the minimum depth of all SSP currently in use

Returns

mimimum SSP depth [m]

References min_ssp_depth_set.

13.1.3.7 getMinSSPDepthSteps() `int woss::ACToolboxWoss::getMinSSPDepthSteps ( ) const [inline]`

Gets the minimum number depth steps of all SSP currently in use

Returns

mimimum SSP depth steps

References min_ssp_depth_steps.

13.1.3.8 getRangeSteps() `int woss::ACToolboxWoss::getRangeSteps ( ) const [inline]`

Gets the total number of range steps

Returns

total number of range steps

References total_range_steps.

13.1.3.9 getSSPDepthPrecision() `long double woss::ACToolboxWoss::getSSPDepthPrecision ( ) const [inline]`

Gets the depth precision for all SSP that will be created

Returns

depth precision [m]

References ssp_depth_precision.

13.1.3.10 initAltimetry() `bool ACToolboxWoss::initAltimetry ( ) [protected], [virtual]`

Initializes altimetry_value

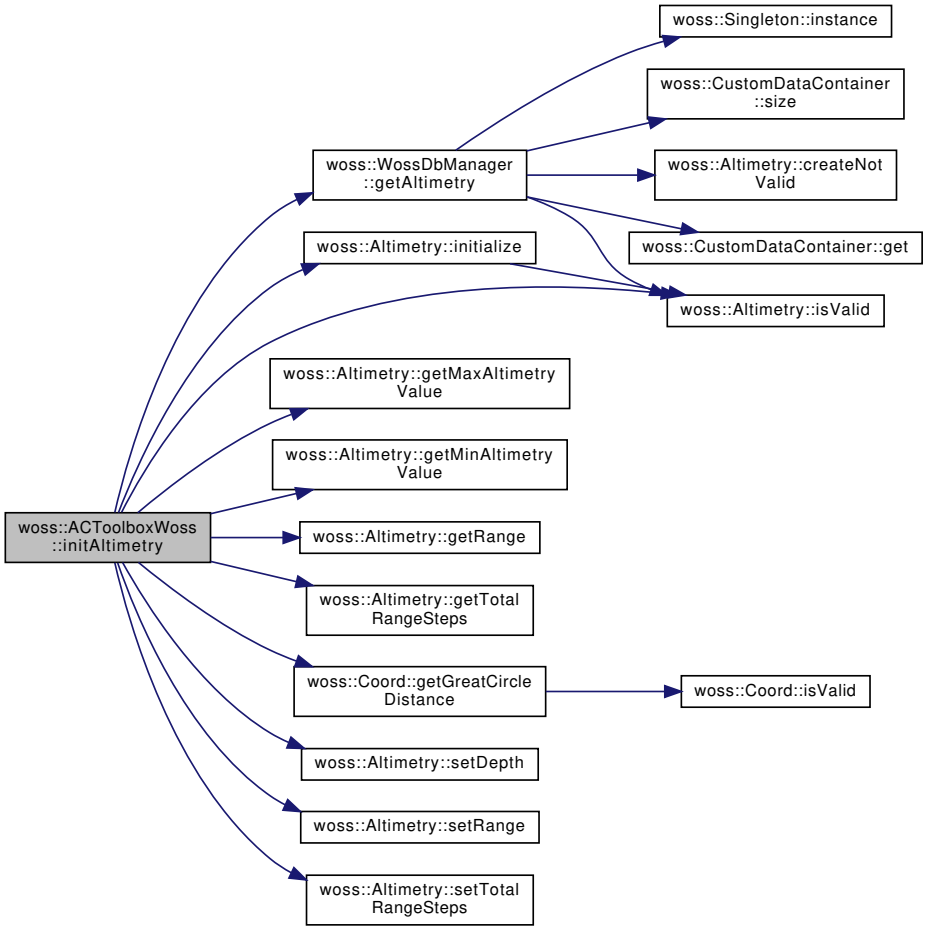
Returns

true if method succeeded, *false* otherwise

References altimetry_value, woss::Woss::db_manager, woss::Woss::debug, woss::WossDbManager::getAltimetry(), woss::Coord::getGreatCircleDistance(), woss::Altimetry::getMaxAltimetryValue(), woss::Altimetry::getMinAltimetryValue(), woss::Altimetry::getRange(), woss::Altimetry::getTotalRangeSteps(), woss::Altimetry::initialize(), woss::Altimetry::isValid(), max_altimetry_depth, max_bathymetry_depth, min_altimetry_depth, woss::Woss::rx_coordz, woss::Altimetry::setDepth(), woss::Altimetry::setRange(), woss::Altimetry::setTotalRangeSteps(), total_range_steps, woss::Woss::tx_coordz, and woss::Woss::woss_id.

Referenced by initialize().

Here is the call graph for this function:



13.1.3.11 initCoordZVector() `bool ACToolboxWoss::initCoordZVector ( ) [protected], [virtual]`

Initializes coordz_vector

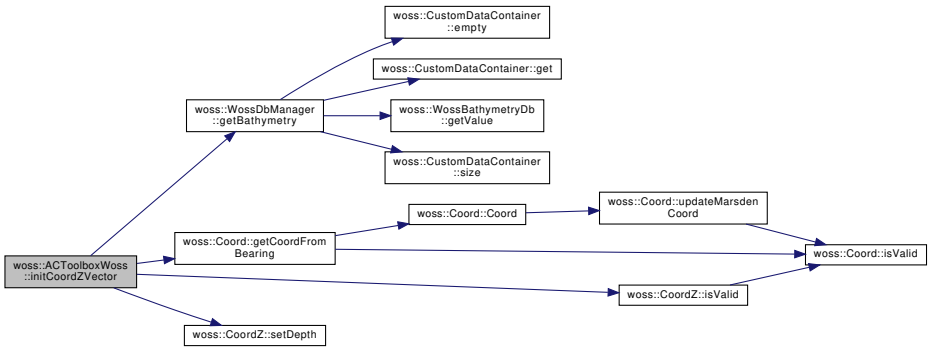
Returns

true if method succeeded, *false* otherwise

References woss::Woss::bearing, coordz_vector, woss::Woss::db_manager, woss::Woss::debug, woss::WossDbManager::getBathym woss::Coord::getCoordFromBearing(), woss::CoordZ::isValid(), max_bathymetry_depth, min_bathymetry_depth, range_vector, woss::Woss::rx_coordz, woss::CoordZ::setDepth(), total_range_steps, woss::Woss::tx_coordz, and woss::Woss::woss_id.

Referenced by initialize().

Here is the call graph for this function:



13.1.3.12 initialize() `bool ACToolboxWoss::initialize ( ) [virtual]`

Initializes the enviroment for acoustic-toolbox channel simulator

Returns

true if method was successful, *false* otherwise

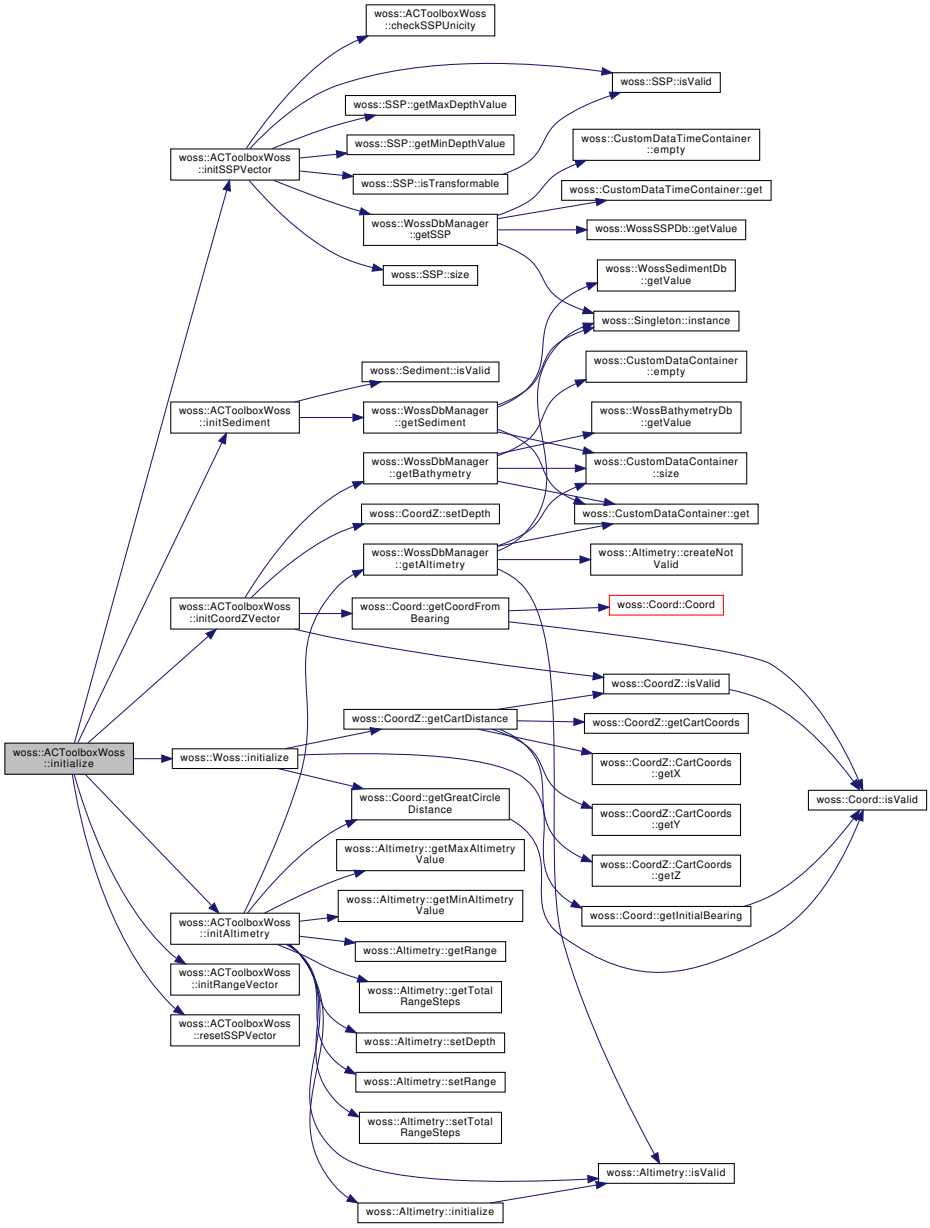
Implements woss::Woss.

Reimplemented in woss::BellhopWoss.

References altimetry_value, coordz_vector, initAltimetry(), initCoordZVector(), woss::Woss::initialize(), initRangeVector(), initSediment(), initSSPVector(), max_altimetry_depth, max_bathymetry_depth, min_altimetry_depth, min_bathymetry_depth, range_vector, resetSSPVector(), and sediment_value.

Referenced by woss::BellhopWoss::initialize().

Here is the call graph for this function:



13.1.3.13 initRangeVector() `bool ACToolboxWoss::initRangeVector ( ) [protected], [virtual]`

Initializes range_vector

Returns

true if method succeeded, *false* otherwise

References woss::Woss::debug, range_vector, woss::Woss::total_great_circle_distance, total_range_steps, and woss::Woss::woss_id.

Referenced by initialize().

13.1.3.14 initSediment() `bool ACToolboxWoss::initSediment ( ) [protected], [virtual]`

Initializes sediment_value

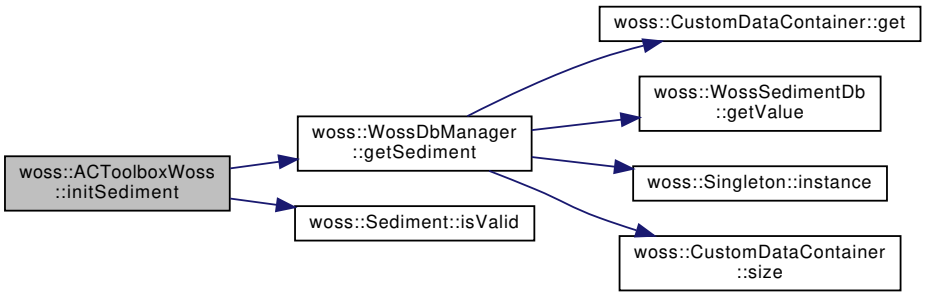
Returns

true if method succeeded, *false* otherwise

References coordz_vector, woss::Woss::db_manager, woss::Woss::debug, woss::WossDbManager::getSediment(), woss::Sediment::isValid(), sediment_value, woss::Woss::tx_coordz, and woss::Woss::woss_id.

Referenced by initialize().

Here is the call graph for this function:



13.1.3.15 initSSPVector() `bool ACToolboxWoss::initSSPVector ( ) [protected], [virtual]`

Initializes ssp_vector

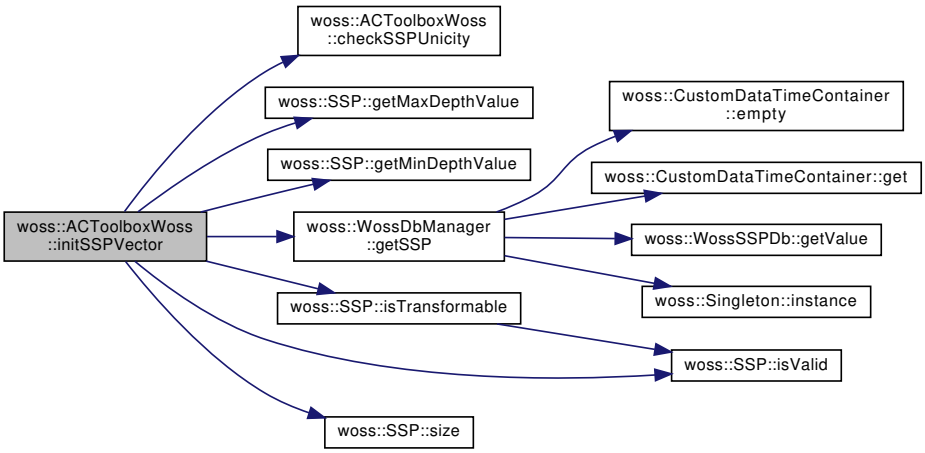
Returns

true if method succeeded, *false* otherwise

References checkSSPUnicity(), coordz_vector, woss::Woss::current_time, woss::Woss::db_manager, woss::Woss::debug, woss::SSP::getMaxDepthValue(), woss::SSP::getMinDepthValue(), woss::WossDbManager::getSSP(), is_ssp_vector_transformable, woss::SSP::isTransformable(), woss::SSP::isValid(), max_ssp_depth_set, max_ssp_depth_steps, min_ssp_depth_set, min_ssp_depth_steps, woss::SSP::size(), ssp_unique_indexes, ssp_vector, total_range_steps, woss::Woss::tx_coordz, and woss::Woss::woss_id.

Referenced by initialize().

Here is the call graph for this function:



13.1.3.16 isValid() `bool ACToolboxWoss::isValid ( ) const [virtual]`

Checks the validity of ACToolboxWoss

Returns

true if it's valid, *false* otherwise

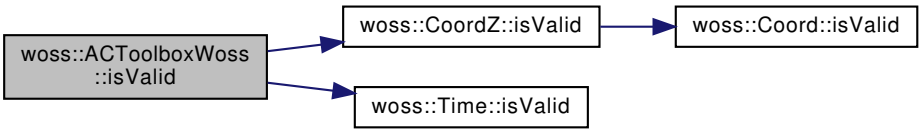
Implements woss::Woss.

Reimplemented in woss::BellhopWoss.

References woss::Woss::end_time, woss::Woss::frequencies, woss::CoordZ::isValid(), woss::Time::isValid(), woss::Woss::rx_coordz, woss::Woss::start_time, and woss::Woss::tx_coordz.

Referenced by woss::BellhopWoss::isValid().

Here is the call graph for this function:



13.1.3.17 resetSSPVector() `void ACToolboxWoss::resetSSPVector ( ) [protected], [virtual]`

Resets ssp_vector

References ssp_vector.

Referenced by initialize(), and ~ACToolboxWoss().

13.1.3.18 setRangeSteps() `ACToolboxWoss & woss::ACToolboxWoss::setRangeSteps ( int steps ) [inline]`

Sets the total number of range steps

Parameters

steps	total number of range steps

References coordz_vector, range_vector, ssp_vector, and total_range_steps.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.1.3.19 setSSPDepthPrecision() `ACToolboxWoss & woss::ACToolboxWoss::setSSPDepthPrecision ( long double precision ) [inline]`

Sets the depth precision for all SSP that will be created

Parameters

precision	depth precision [m]

References ssp_depth_precision.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.1.4 Member Data Documentation

13.1.4.1 altimetry_value `Altimetry* woss::ACToolboxWoss::altimetry_value [protected]`

Altimetry in use

Referenced by initAltimetry(), initialize(), woss::BellhopWoss::timeEvolve(), woss::BellhopWoss::writeNormalizedSSP(), and ~ACToolboxWoss().

13.1.4.2 coordz_vector `CoordZVector woss::ACToolboxWoss::coordz_vector [protected]`

Vector of all CoordZ involved. Its size is equal to total_range_steps

Referenced by woss::BellhopWoss::checkDepthOffsets(), initCoordZVector(), initialize(), initSediment(), initSSPVector(), and setRangeSteps().

13.1.4.3 is_ssp_vector_transformable `bool woss::ACToolboxWoss::is_ssp_vector_transformable [protected]`

True if ssp_vector is transformable

See also

SSP::isTransformable()

Referenced by initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.4 max_altimetry_depth `double woss::ACToolboxWoss::max_altimetry_depth [protected]`

Maximum altimetry depth [m]

Referenced by initAltimetry(), and initialize().

13.1.4.5 max_bathymetry_depth `double woss::ACToolboxWoss::max_bathymetry_depth [protected]`

Maximum bathymetry depth [m]

Referenced by woss::BellhopWoss::checkDepthOffsets(), getMaxBathymetryDepth(), initAltimetry(), initCoordZVector(), initialize(), woss::BellhopWoss::initialize(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.6 max_ssp_depth_set `::std::set< double > woss::ACToolboxWoss::max_ssp_depth_set [protected]`

Set of all maximum SSP depth [m] currently in use

Referenced by getMaxSSPDepth(), initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.7 max_ssp_depth_steps `int woss::ACToolboxWoss::max_ssp_depth_steps [protected]`

Maximum number of currently in use SSP depth steps

Referenced by getMaxSSPDepthSteps(), initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.8 min_altimetry_depth `double woss::ACToolboxWoss::min_altimetry_depth [protected]`

Minimum altimetry depth [m]

Referenced by initAltimetry(), initialize(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.9 min_bathymetry_depth `double woss::ACToolboxWoss::min_bathymetry_depth [protected]`

Minimum bathymetry depth [m]

Referenced by getMinBathymteryDepth(), initCoordZVector(), and initialize().

13.1.4.10 min_ssp_depth_set ::std::set< double > woss::ACToolboxWoss::min_ssp_depth_set [protected]

Set of all minimum SSP depth [m] currently in use

Referenced by getMinSSPDepth(), initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.11 min_ssp_depth_steps int woss::ACToolboxWoss::min_ssp_depth_steps [protected]

Miminum number of currently in use SSP depth steps

Referenced by getMinSSPDepthSteps(), initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.12 range_vector RangeVector woss::ACToolboxWoss::range_vector [protected]

Vector of all ranges [m] involved. Its size is equal to total_range_steps

Referenced by initCoordZVector(), initialize(), initRangeVector(), woss::BellhopWoss::normalizeDbSSP(), and setRangeSteps().

13.1.4.13 sediment_value Sediment* woss::ACToolboxWoss::sediment_value [protected]

Sediment in use

Referenced by initialize(), initSediment(), woss::BellhopWoss::writeSediment(), and ~ACToolboxWoss().

13.1.4.14 ssp_depth_precision long double woss::ACToolboxWoss::ssp_depth_precision [protected]

Depth precision of all SSP that will be created [m]

Referenced by getSSPDepthPrecision(), and setSSPDepthPrecision().

13.1.4.15 ssp_unique_indexes ::std::set< int > woss::ACToolboxWoss::ssp_unique_indexes [protected]

Set of of ssp_vector indexes that link to unique SSP

Referenced by checkSSPUnicity(), initSSPVector(), and woss::BellhopWoss::normalizeDbSSP().

13.1.4.16 ssp_vector `SSPVector woss::ACToolboxWoss::ssp_vector [protected]`

Vector of all SSP involved. Its size is less or equal to total_range_steps

Referenced by checkSSPUnicity(), initSSPVector(), woss::BellhopWoss::normalizeDbSSP(), resetSSPVector(), and setRangeSteps().

13.1.4.17 total_range_steps `int woss::ACToolboxWoss::total_range_steps [protected]`

Number of range steps

Referenced by getRangeSteps(), initAltimetry(), initCoordZVector(), initRangeVector(), initSSPVector(), woss::BellhopWoss::isValid(), and setRangeSteps().

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

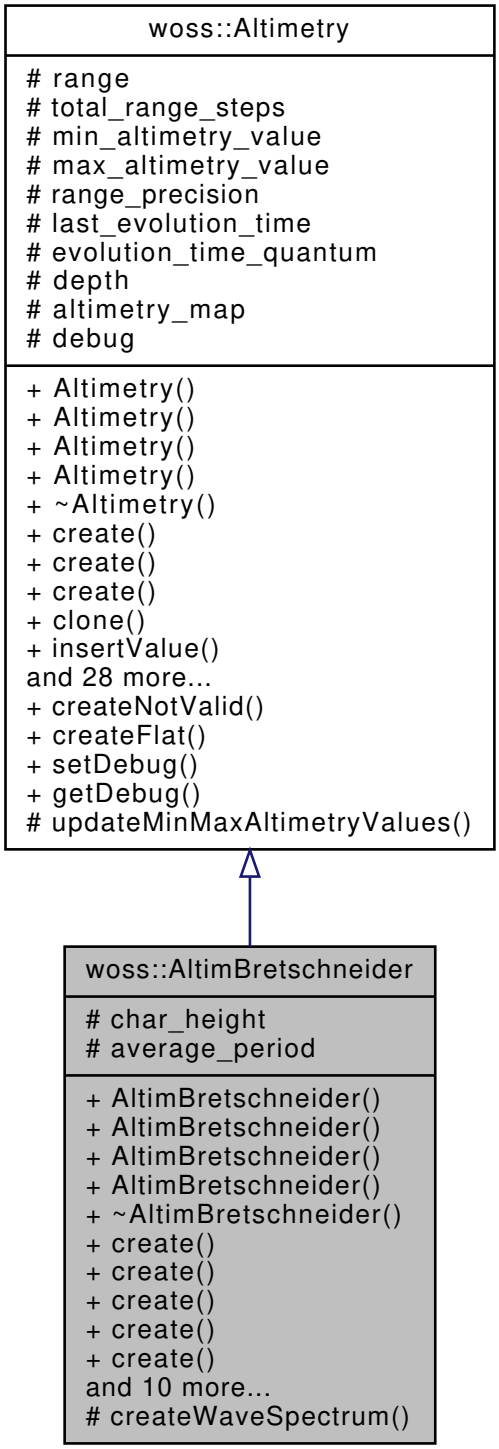
- woss/ac-toolbox-woss.h
- woss/ac-toolbox-woss.cpp

13.2 woss::AltimBretschneider Class Reference

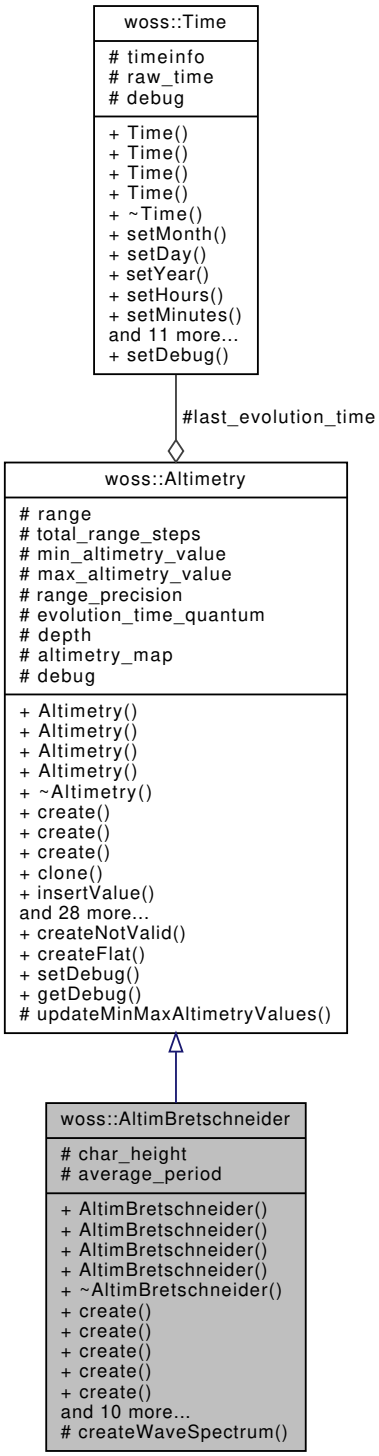
AltimBretschneider supports Bretschneider wave model.

`#include <altimetry-definitions.h>`

Inheritance diagram for woss::AltimBretschneider:



Collaboration diagram for woss::AltimBretschneider:



Public Member Functions

- AltimBretschneider ()
- AltimBretschneider (AltimetryMap &map)
- AltimBretschneider (double ch_height, double avg_per, int total_range_steps, double depth)
- AltimBretschneider (const AltimBretschneider &copy)
- virtual AltimBretschneider * create () const

- virtual AltimBretschneider * create (AltimetryMap &map) const
- virtual AltimBretschneider * create (const AltimBretschneider &copy) const
- virtual AltimBretschneider * create (const Altimetry &copy) const
- virtual AltimBretschneider * create (double ch_height, double avg_per, int total_range_steps, double depth) const
- virtual AltimBretschneider * clone () const
- AltimBretschneider & operator= (const AltimBretschneider &copy)
- virtual bool initialize ()
- virtual bool isValid () const
- virtual AltimBretschneider * timeEvolve (const Time &time_value)
- virtual AltimBretschneider * randomize (double ratio_incr_value) const
- AltimBretschneider & setCharacteristicHeight (double h)
- AltimBretschneider & setAveragePeriod (double p)
- double getCharacteristicHeight () const
- double getAveragePeriod () const

Protected Member Functions

- virtual AltimBretschneider & **createWaveSpectrum** ()

Protected Attributes

- double char_height
- double average_period

Additional Inherited Members

13.2.1 Detailed Description

AltimBretschneider supports Bretschneider wave model.

AltimBretschneider wave model

13.2.2 Constructor & Destructor Documentation

13.2.2.1 AltimBretschneider() [1/4] AltimBretschneider::AltimBretschneider ( )

Default AltimBretschneider constructor

Referenced by clone(), and create().

13.2.2.2 AltimBretschneider() [2/4] AltimBretschneider::AltimBretschneider (AltimetryMap & *map*)

AltimBretschneider constructor

Parameters

map	custom time arrival map

13.2.2.3 AltimBretschneider() [3/4]

```
AltimBretschneider::AltimBretschneider (
        double ch_height,
        double avg_per,
        int total_range_steps,
        double depth )
```

AltimBretschneider constructor

Parameters

ch_height	characteristic wave height [m]
avg_per	averate period [s]
total_range_steps	total range steps
depth	depth [m]

References woss::Altimetry::depth, and woss::Altimetry::total_range_steps.

13.2.2.4 AltimBretschneider() [4/4]

```
AltimBretschneider::AltimBretschneider (
        const AltimBretschneider & copy )
```

AltimBretschneider copy constructor

Parameters

copy	AltimBretschneider to be copied

13.2.3 Member Function Documentation

13.2.3.1 clone()

```
virtual AltimBretschneider * woss::AltimBretschneider::clone ( ) const  [inline], [virtual]
```

AltimBretschneider virtual factory method

Returns

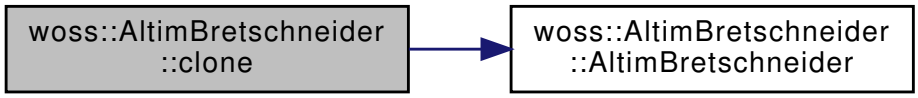
a heap-created copy of **this** instance

Reimplemented from woss::Altimetry.

References AltimBretschneider().

Referenced by randomize(), and timeEvolve().

Here is the call graph for this function:



13.2.3.2 create() [1/5]

```
virtual AltimBretschneider * woss::AltimBretschneider::create ( ) const
[inline], [virtual]
```

AltimBretschneider virtual factory method

Returns

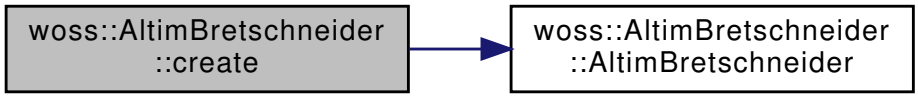
a heap-created AltimBretschneider object

Reimplemented from woss::Altimetry.

References AltimBretschneider().

Referenced by create().

Here is the call graph for this function:



13.2.3.3 create() [2/5]

```
virtual AltimBretschneider * woss::AltimBretschneider::create (
        AltimetryMap & map ) const  [inline], [virtual]
```

AltimBretschneider virtual factory method

Parameters

map	custom time arrival map

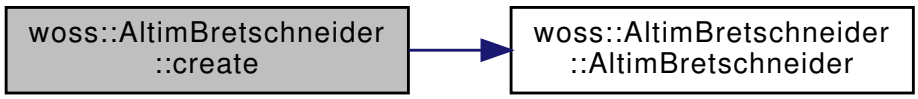
Returns

a heap-created AltimBretschneider object

Reimplemented from woss::Altimetry.

References AltimBretschneider().

Here is the call graph for this function:



13.2.3.4 create() [3/5] virtual AltimBretschneider * woss::AltimBretschneider::create (const AltimBretschneider & *copy*) const [inline], [virtual]

AltimBretschneider virtual factory method

Parameters

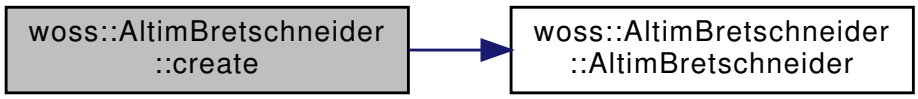
copy	AltimBretschneider to be copied

Returns

a heap-created AltimBretschneider object

References AltimBretschneider().

Here is the call graph for this function:



13.2.3.5 create() [4/5] virtual AltimBretschneider * woss::AltimBretschneider::create (const Altimetry & *copy*) const [inline], [virtual]

AltimBretschneider virtual factory method

Parameters

copy	AltimBretschneider to be copied

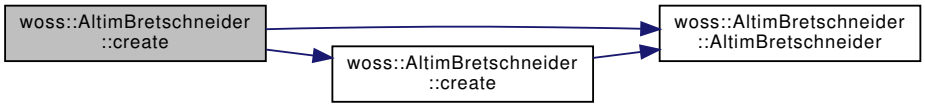
Returns

a heap-created AltimBretschneider object

Reimplemented from woss::Altimetry.

References AltimBretschneider(), and create().

Here is the call graph for this function:



13.2.3.6 create() [5/5]

```
virtual AltimBretschneider * woss::AltimBretschneider::create (
        double ch_height,
        double avg_per,
        int total_range_steps,
        double depth ) const  [inline], [virtual]
```

AltimBretschneider virtual factory method

Parameters

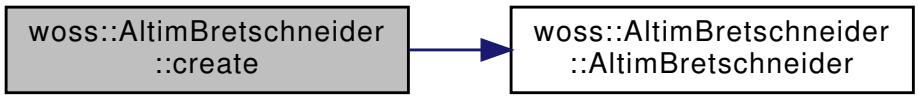
ch_height	characteristic wave height [m]
avg_per	averate period [s]
total_range_steps	total range steps
depth	depth [m]

Returns

a heap-created AltimBretschneider object

References AltimBretschneider(), woss::Altimetry::depth, and woss::Altimetry::total_range_steps.

Here is the call graph for this function:



13.2.3.7 getAveragePeriod() `double woss::AltimBretschneider::getAveragePeriod ( ) const [inline]`

Gets T - the wave average period [s]

Returns

h average period [s]

References average_period.

13.2.3.8 getCharacteristicHeight() `double woss::AltimBretschneider::getCharacteristicHeight ( ) const [inline]`

Gets H - the characteristic height of the wave [m]

Returns

characteristic height [m]

References char_height.

13.2.3.9 initialize() `bool AltimBretschneider::initialize ( ) [virtual]`

Initializes the altimetry vector. used if there is a matematical function that generates the whole vector.

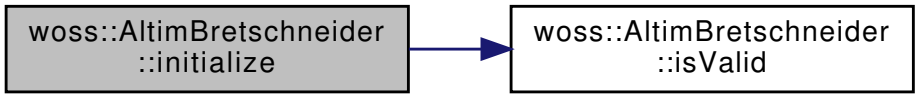
Returns

true succeeded *false* otherwise

Reimplemented from woss::Altimetry.

References isValid(), woss::Altimetry::range, woss::Altimetry::range_precision, and woss::Altimetry::total_range_steps.

Here is the call graph for this function:



13.2.3.10 isValid() `bool AltimBretschneider::isValid ( ) const [virtual]`

Checks the validity of AltimBretschneider

Returns

true if it has at least one value, *false* otherwise

Reimplemented from woss::Altimetry.

References average_period, char_height, woss::Altimetry::debug, and woss::Altimetry::total_range_steps.

Referenced by initialize().

13.2.3.11 operator=() `AltimBretschneider & AltimBretschneider::operator= ( const AltimBretschneider & copy )`

Assignment operator

Parameters

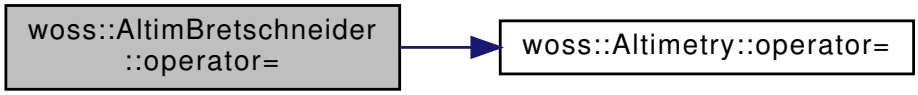
copy	const reference to a AltimBretschneider object to be copied

Returns

AltimBretschneider reference to *this*

References average_period, char_height, and woss::Altimetry::operator=().

Here is the call graph for this function:



13.2.3.12 randomize() `AltimBretschneider * AltimBretschneider::randomize ( double ratio_incr_value ) const [virtual]`

Performs a random perturbation of altimetry values with given ratio

Parameters

ratio_incr_value	perturbation ratio

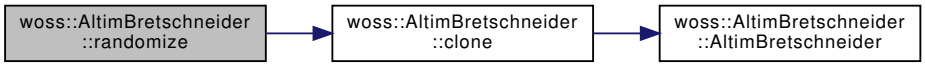
Returns

a *new* Altimetry object

Reimplemented from woss::Altimetry.

References clone(), and woss::Altimetry::debug.

Here is the call graph for this function:



13.2.3.13 setAveragePeriod() `AltimBretschneider & woss::AltimBretschneider::setAveragePeriod ( double p ) [inline]`

Configures T - the wave average period [s]

Parameters

h	average period [s]

Returns

AltimBretschneider reference to *this*

References average_period.

13.2.3.14 setCharacteristicHeight()

```
AltimBretschneider & woss::AltimBretschneider::setCharacteristic↩
Height (
            double h )  [inline]
```

Configures H - the characteristic height of the wave [m]

Parameters

h	characteristic height [m]

Returns

AltimBretschneider reference to *this*

References char_height.

13.2.3.15 timeEvolve()

```
AltimBretschneider * AltimBretschneider::timeEvolve (
            const Time & time_value )  [virtual]
```

Performs a time evoulion

Parameters

time_value	const reference to a valid Time object

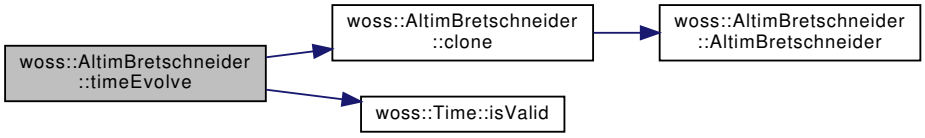
Returns

a pointer to a new heap allocated AltimBretschneider object

Reimplemented from woss::Altimetry.

References clone(), woss::Altimetry::debug, woss::Altimetry::evolution_time_quantum, woss::Time::isValid(), and woss::Altimetry::last_evolution_time.

Here is the call graph for this function:



13.2.4 Member Data Documentation

13.2.4.1 average_period `double woss::AltimBretschneider::average_period [protected]`

T - Model's average wave period [s] Refer to: G. J. Komen et al., Dynamics and modeling of ocean waves. Cambridge University Press, 1994.

Referenced by getAveragePeriod(), isValid(), operator=(), and setAveragePeriod().

13.2.4.2 char_height `double woss::AltimBretschneider::char_height [protected]`

H - Model's characteristic height [m] Refer to: G. J. Komen et al., Dynamics and modeling of ocean waves. Cambridge University Press, 1994.

Referenced by getCharacteristicHeight(), isValid(), operator=(), and setCharacteristicHeight().

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

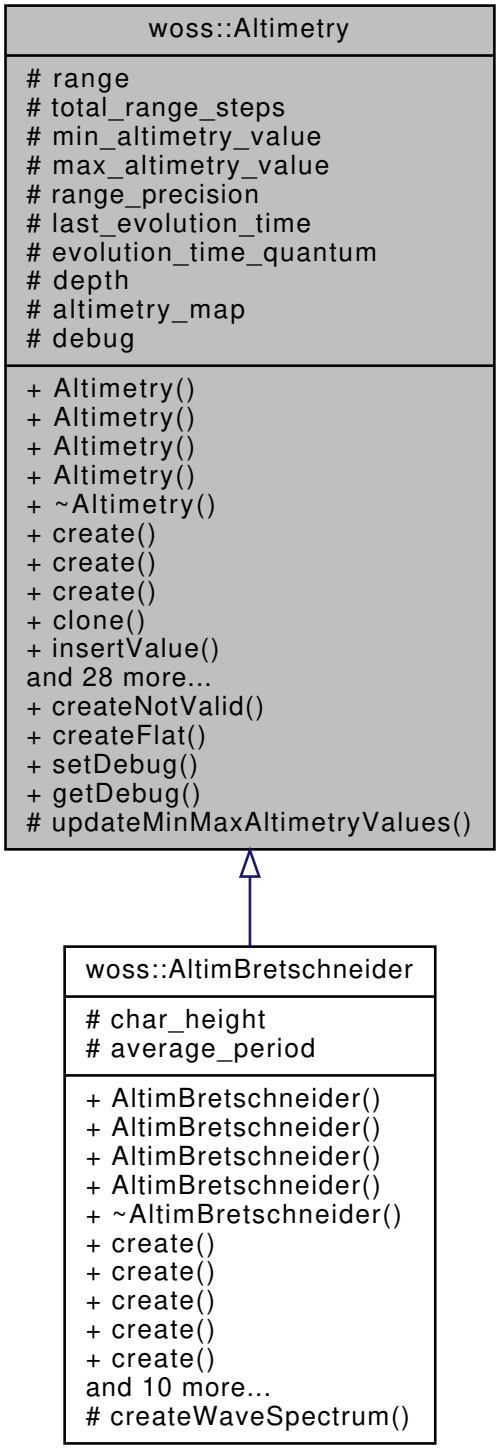
- woss/woss_def/altimetry-definitions.h
- woss/woss_def/altimetry-definitions.cpp

13.3 woss::Altimetry Class Reference

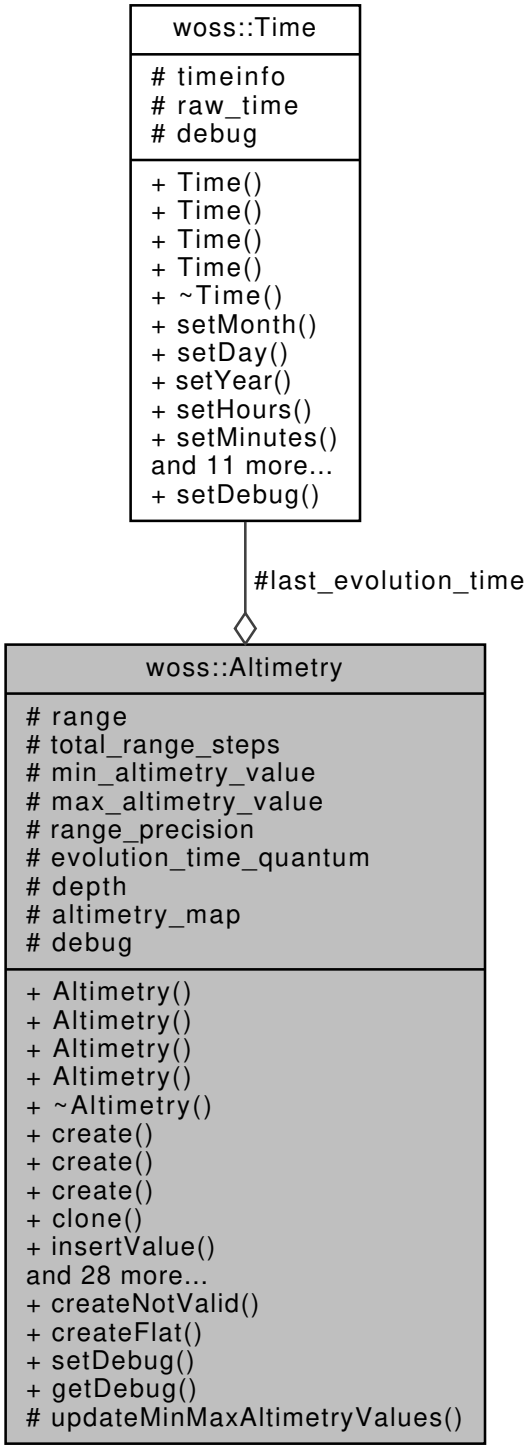
Altimetry profile class.

`#include <altimetry-definitions.h>`

Inheritance diagram for woss::Altimetry:



Collaboration diagram for woss::Altimetry:



Public Member Functions

- Altimetry ()
- Altimetry (AltimetryMap &map)
- Altimetry (double range, double altimetry)
- Altimetry (const Altimetry &copy)
- virtual Altimetry * create () const

- virtual Altimetry * create (AltimetryMap &map) const
- virtual Altimetry * create (const Altimetry &copy) const
- virtual Altimetry * clone () const
- Altimetry & insertValue (double range, double altimetry)
- Altimetry & sumValue (double range, double altimetry)
- AltCIt findValue (double range) const
- Altimetry & eraseValue (double range)
- virtual Altimetry * crop (double range_start, double range_end)
- virtual Altimetry * randomize (double ratio_incr_value) const
- AltCIt begin () const
- AltCIt end () const
- AltCIt at (const int i) const
- int size () const
- bool empty () const
- void clear ()
- Altimetry & setEvolutionTimeQuantum (double quantum)
- Altimetry & setTotalRangeSteps (int r_s)
- Altimetry & setRange (double r)
- Altimetry & setDepth (double d)
- double getMaxRangeValue () const
- double getMinRangeValue () const
- double getMaxAltimetryValue () const
- double getMinAltimetryValue () const
- long double getRangePrecision () const
- double getEvolutionTimeQuantum () const
- double getRange () const
- int getTotalRangeSteps () const
- double getDepth () const
- virtual bool isValid () const
- virtual bool initialize ()
- virtual Altimetry * timeEvolve (const Time &time_value)
- Altimetry & operator= (const Altimetry &copy)

Static Public Member Functions

- static AltimetryMap & createNotValid ()
- static AltimetryMap & createFlat (double altimetry=0)
- static void setDebug (bool flag)
- static bool getDebug (bool flag)

Protected Member Functions

- virtual Altimetry & **updateMinMaxAltimetryValues** ()

Protected Attributes

- double range
- int total_range_steps
- double min_altimetry_value
- double max_altimetry_value
- long double range_precision
- Time last_evolution_time
- double evolution_time_quantum
- double depth
- AltimetryMap altimetry_map

Static Protected Attributes

- static bool debug = false

Friends

- bool operator== (const Altimetry &left, const Altimetry &right)
- bool operator!= (const Altimetry &left, const Altimetry &right)
- const Altimetry operator+ (const Altimetry &left, const Altimetry &right)
- const Altimetry operator- (const Altimetry &left, const Altimetry &right)
- const Altimetry operator+ (const Altimetry &left, const double right)
- const Altimetry operator- (const Altimetry &left, const double right)
- const Altimetry operator/ (const Altimetry &left, const double right)
- const Altimetry operator* (const Altimetry &left, const double right)
- const Altimetry operator+ (const double left, const Altimetry &right)
- const Altimetry operator- (const double left, const Altimetry &right)
- const Altimetry operator/ (const double left, const Altimetry &right)
- const Altimetry operator* (const double left, const Altimetry &right)
- Altimetry & operator+= (Altimetry &left, const Altimetry &right)
- Altimetry & operator-= (Altimetry &left, const Altimetry &right)
- Altimetry & operator+= (Altimetry &left, double right)
- Altimetry & operator-= (Altimetry &left, double right)
- Altimetry & operator/= (Altimetry &left, double right)
- Altimetry & operator*= (Altimetry &left, double right)
- std::ostream & operator<< (std::ostream &os, const Altimetry &instance)

13.3.1 Detailed Description

Altimetry profile class.

Altimetry class offers the possibility to store and manipulate altimetry profiles, e.g. a collection of range [m] values associated to a altimetry value [m].

13.3.2 Constructor & Destructor Documentation

13.3.2.1 Altimetry() [1/4] `Altimetry::Altimetry ( )`

Default Altimetry constructor

Referenced by clone(), create(), crop(), and randomize().

13.3.2.2 Altimetry() [2/4] `Altimetry::Altimetry ( AltimetryMap & map )`

Altimetry constructor

Parameters

map	custom time arrival map

References altimetry_map, and range_precision.

13.3.2.3 Altimetry() [3/4]

```
Altimetry::Altimetry (
        double range,
        double altimetry )
```

Altimetry constructor

Parameters

range	range [m]
altimetry	altimetry [m]

References altimetry_map, max_altimetry_value, min_altimetry_value, and range.

13.3.2.4 Altimetry() [4/4]

```
Altimetry::Altimetry (
        const Altimetry & copy )
```

Altimetry copy constructor

Parameters

copy	Altimetry to be copied

13.3.3 Member Function Documentation

13.3.3.1 at()

```
AltCIt Altimetry::at (
        const int i ) const
```

Returns a const iterator to the altimetry value at i-th position

Parameters

i	integer should be between 0 and size()

Returns

const iterator to end() if position *i* is not found

References altimetry_map.

13.3.3.2 begin() `AltCIt woss::Altimetry::begin ( ) const [inline]`

Returns a const iterator to the beginning of the altimetry map

Returns

const iterator

References altimetry_map.

13.3.3.3 clear() `void woss::Altimetry::clear ( ) [inline]`

Erase all values of altimetry

References altimetry_map.

13.3.3.4 clone() `virtual Altimetry * woss::Altimetry::clone ( ) const [inline], [virtual]`

Altimetry virtual factory method

Returns

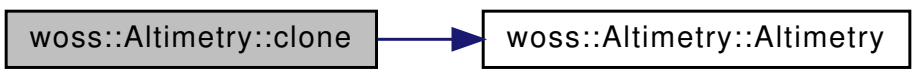
a heap-created copy of **this** instance

Reimplemented in woss::AltimBretschneider.

References Altimetry().

Referenced by woss::DefHandler::operator=(), and timeEvolve().

Here is the call graph for this function:



13.3.3.5 create() [1/3] virtual Altimetry * woss::Altimetry::create ( ) const [inline], [virtual]

Altimetry virtual factory method

Returns

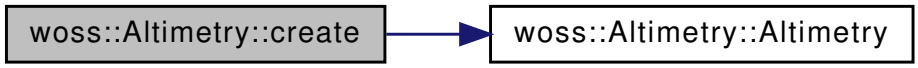
a heap-created Altimetry object

Reimplemented in woss::AltimBretschneider.

References Altimetry().

Referenced by crop().

Here is the call graph for this function:



13.3.3.6 create() [2/3] virtual Altimetry * woss::Altimetry::create (AltimetryMap & *map*) const [inline], [virtual]

Altimetry virtual factory method

Parameters

map	custom time arrival map

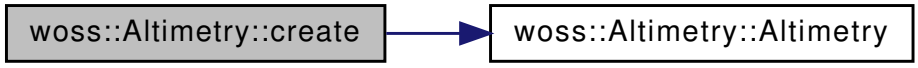
Returns

a heap-created Altimetry object

Reimplemented in woss::AltimBretschneider.

References Altimetry().

Here is the call graph for this function:



13.3.3.7 create() [3/3] virtual Altimetry * woss::Altimetry::create (const Altimetry & *copy*) const [inline], [virtual]

Altimetry virtual factory method

Parameters

copy	Altimetry to be copied

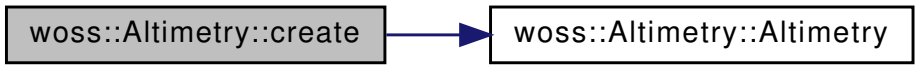
Returns

a heap-created Altimetry object

Reimplemented in woss::AltimBretschneider.

References Altimetry().

Here is the call graph for this function:



13.3.3.8 createFlat()

```
AltimetryMap & woss::Altimetry::createFlat (
        double altimetry = 0 )  [inline], [static]
```

Creates a flat altimetry

Returns

a new flat altimetry instance (e.g. range 0.0 = altimetry)

References altimetry_map.

13.3.3.9 createNotValid()

```
AltimetryMap & woss::Altimetry::createNotValid ( )  [inline], [static]
```

Creates an instance not valid

Returns

a new instance not valid (e.g. range 0.0 = +inf)

References altimetry_map.

Referenced by crop(), woss::WossDbManager::getAltimetry(), and randomize().

13.3.3.10 crop()

```
Altimetry * Altimetry::crop (
        double range_start,
        double range_end )  [virtual]
```

Crops the Altimetry between given range values and returns a new heap-allocated object. The new object will have range values in [range_start, range_end)

Parameters

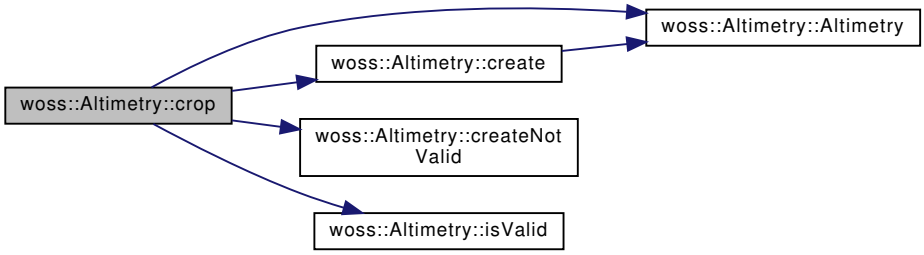
time_start	start range value [m]
time_end	end range value [m]

Returns

a new Altimetry object

References Altimetry(), altimetry_map, create(), createNotValid(), debug, isValid(), and range_precision.

Here is the call graph for this function:



13.3.3.11 empty() `bool woss::Altimetry::empty ( ) const [inline]`

Checks if the instance has stored values

Returns

true if condition applies, *false* otherwise

References altimetry_map.

13.3.3.12 end() `AltCIt woss::Altimetry::end ( ) const [inline]`

Returns a const iterator to the end of the altimetry map

Returns

const iterator

References altimetry_map.

13.3.3.13 eraseValue() `Altimetry & woss::Altimetry::eraseValue ( double range ) [inline]`

Erase the altimetry value with key == *range* parameter

Parameters

range	range value [s]

Returns

reference to ∗**this**

References altimetry_map, and range.

13.3.3.14 findValue() `AltCIt woss::Altimetry::findValue ( double range ) const [inline]`

Returns a const iterator to the altimetry with key == *range* parameter

Parameters

range	range value [m]

Returns

const iterator to end() if *range* is not found

References altimetry_map, and range.

13.3.3.15 getDebug() `static bool woss::Altimetry::getDebug ( bool flag ) [inline], [static]`

Gets debug flag for all instances

Returns

flag debug

References debug.

13.3.3.16 getDepth() `double woss::Altimetry::getDepth ( ) const [inline]`

Returns the scenario depth

Returns

depth [m]

References depth.

13.3.3.17 getEvolutionTimeQuantum() `double woss::Altimetry::getEvolutionTimeQuantum ( ) const [inline]`

Returns the evolution time quantum

Returns

time quantum [s]

References evolution_time_quantum.

13.3.3.18 getMaxAltimetryValue() `double woss::Altimetry::getMaxAltimetryValue ( ) const [inline]`

Returns the maximum altimetry value

Returns

maximum altimetry [m]

References max_altimetry_value.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.19 getMaxRangeValue() `double woss::Altimetry::getMaxRangeValue ( ) const [inline]`

Returns the maximum range value

Returns

maximum range [m]

References altimetry_map.

13.3.3.20 getMinAltimetryValue() `double woss::Altimetry::getMinAltimetryValue ( ) const [inline]`

Returns the minimum altimetry value

Returns

minimum altimetry [m]

References min_altimetry_value.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.21 getMinRangeValue() `double woss::Altimetry::getMinRangeValue ( ) const [inline]`

Returns the maximum range value

Returns

maximum range [m]

References altimetry_map.

13.3.3.22 getRange() `double woss::Altimetry::getRange ( ) const [inline]`

Returns the max range [m]

Returns

max range [m]

References range.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.23 getRangePrecision() `long double woss::Altimetry::getRangePrecision ( ) const [inline]`

Returns the range precision

Returns

range precision [m]

References range_precision.

13.3.3.24 getTotalRangeSteps() `int woss::Altimetry::getTotalRangeSteps ( ) const [inline]`

Returns the total range step

Returns

range steps

References total_range_steps.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.25 initialize() `bool Altimetry::initialize ( ) [virtual]`

Initializes the altimetry vector. used if there is a matematical function that generates the whole vector.

Returns

true succeeded *false* otherwise

Reimplemented in woss::AltimBretschneider.

References isValid(), range, range_precision, and total_range_steps.

Referenced by woss::ACToolboxWoss::initAltimetry().

Here is the call graph for this function:

13.3.3.26 insertValue() `Altimetry & Altimetry::insertValue ( double range, double altimetry )`

Inserts and replace an altimetry value at closest range step

Parameters

range	range value [m]
altimetry	altimetry value [m]; > 0 ==> under water; < 0 ==> above water

Returns

reference to ∗**this**

References altimetry_map, max_altimetry_value, min_altimetry_value, and range.

13.3.3.27 isValid() `bool Altimetry::isValid ( ) const [virtual]`

Checks the validity of Altimetry

Returns

true if it has at least one value, *false* otherwise

Reimplemented in woss::AltimBretschneider.

References altimetry_map.

Referenced by crop(), woss::WossDbManager::getAltimetry(), woss::ACToolboxWoss::initAltimetry(), initialize(), randomize(), woss::BellhopWoss::timeEvolve(), and woss::BellhopWoss::writeNormalizedSSP().

13.3.3.28 operator=() `Altimetry & Altimetry::operator= ( const Altimetry & copy )`

Assignment operator

Parameters

copy	const reference to a Altimetry object to be copied

Returns

Altimetry reference to *this*

References altimetry_map, depth, evolution_time_quantum, last_evolution_time, max_altimetry_value, min_altimetry_value, range, range_precision, and total_range_steps.

Referenced by woss::AltimBretschneider::operator=().

13.3.3.29 randomize() `Altimetry * Altimetry::randomize ( double ratio_incr_value ) const [virtual]`

Performs a random perturbation of altimetry values with given ratio

Parameters

ratio_incr_value	perturbation ratio

Returns

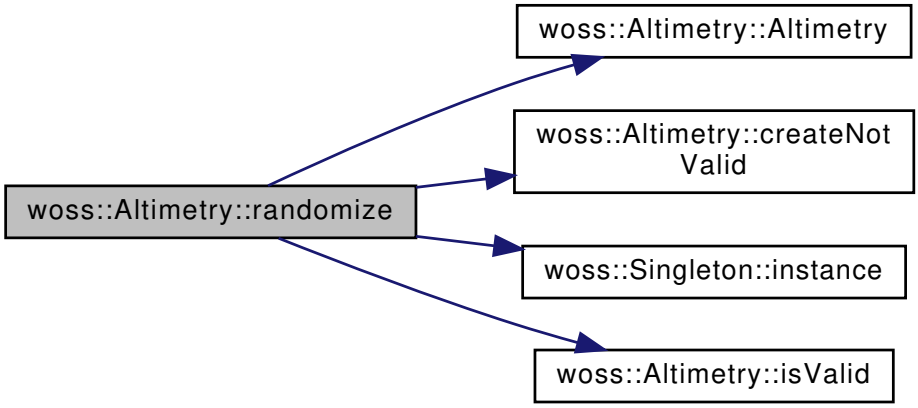
a *new* Altimetry object

Reimplemented in woss::AltimBretschneider.

References Altimetry(), altimetry_map, createNotValid(), woss::Singleton< T >::instance(), and isValid().

Referenced by timeEvolve().

Here is the call graph for this function:



13.3.3.30 setDebug() `static void woss::Altimetry::setDebug ( bool flag ) [inline], [static]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.3.3.31 setDepth() `Altimetry & woss::Altimetry::setDepth ( double d ) [inline]`

Sets the depth

Parameters

d	scenario depth [m]

References depth.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.32 setEvolutionTimeQuantum() `Altimetry & woss::Altimetry::setEvolutionTimeQuantum ( double quantum ) [inline]`

Sets the evolution time quantum

Parameters

quantum	time quantum [s]

References evolution_time_quantum.

13.3.3.33 setRange() `Altimetry & woss::Altimetry::setRange ( double r ) [inline]`

Sets the range

Parameters

r	max range [m]

References range.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.34 setTotalRangeSteps()

```
Altimetry & woss::Altimetry::setTotalRangeSteps (
        int r_s )  [inline]
```

Sets the total range step

Parameters

r_s	range steps

References total_range_steps.

Referenced by woss::ACToolboxWoss::initAltimetry().

13.3.3.35 size()

```
int woss::Altimetry::size ( ) const  [inline]
```

Returns the number of altimetry values stored

Returns

number of altimetry values stored

References altimetry_map.

13.3.3.36 sumValue()

```
Altimetry & woss::Altimetry::sumValue (
        double range,
        double altimetry )  [inline]
```

Inserts and sums an altimetry value at closest range step

Parameters

range	range value [m]
altimetry	altimetry value [m]; $>$ 0 ==$>$ under water; $<$ 0 ==$>$ above water

References altimetry_map, max_altimetry_value, min_altimetry_value, and range.

13.3.3.37 timeEvolve() `Altimetry * Altimetry::timeEvolve ( const Time & time_value ) [virtual]`

Performs a time evoulion

Parameters

time_value	const reference to a valid Time object

Returns

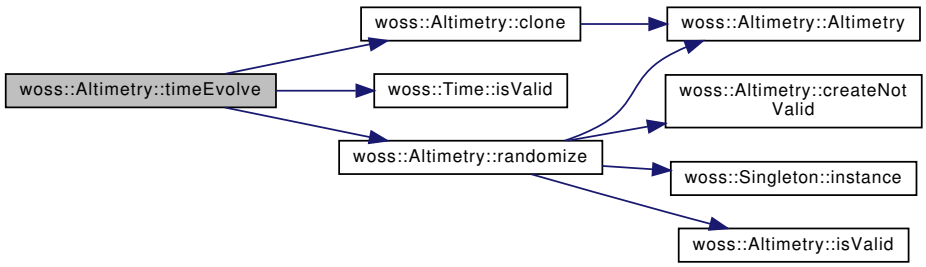
a pointer to a new heap allocated Altimetry object

Reimplemented in woss::AltimBretschneider.

References clone(), debug, evolution_time_quantum, woss::Time::isValid(), last_evolution_time, and randomize().

Referenced by woss::BellhopWoss::timeEvolve().

Here is the call graph for this function:



13.3.4 Friends And Related Function Documentation

13.3.4.1 operator"!= `bool operator!= ( const Altimetry & left, const Altimetry & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.3.4.2 operator* [1/2]

```
const Altimetry operator* (
    const Altimetry & left,
    const double right ) [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.3 operator* [2/2]

```
const Altimetry operator* (
    const double left,
    const Altimetry & right ) [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.4 operator*=

```
Altimetry & operator*= (
    Altimetry & left,
    double right ) [friend]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.5 operator+ [1/3]

```
const Altimetry operator+ (
    const Altimetry & left,
    const Altimetry & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.6 operator+ [2/3]

```
const Altimetry operator+ (
    const Altimetry & left,
    const double right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.7 operator+ [3/3]

```
const Altimetry operator+ (
        const double left,
        const Altimetry & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.8 operator+= [1/2]

```
Altimetry & operator+= (
        Altimetry & left,
        const Altimetry & right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.9 operator+= [2/2]

```
Altimetry & operator+= (
        Altimetry & left,
        double right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.10 operator- [1/3]

```
const Altimetry operator- (
    const Altimetry & left,
    const Altimetry & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.11 operator- [2/3]

```
const Altimetry operator- (
    const Altimetry & left,
    const double right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.12 operator- [3/3]

```
const Altimetry operator- (
    const double left,
    const Altimetry & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.13 operator-= [1/2] Altimetry & operator-= (Altimetry & *left*, const Altimetry & *right*) [friend]

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.14 operator-= [2/2] Altimetry & operator-= (Altimetry & *left*, double *right*) [friend]

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.15 operator/ [1/2] const Altimetry operator/ (const Altimetry & *left*, const double *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.16 operator/ [2/2] const Altimetry operator/ (const double *left*, const Altimetry & *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.3.4.17 operator/= Altimetry & operator/= (Altimetry & *left*, double *right*) [friend]

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.3.4.18 operator<< std::ostream & operator<< (std::ostream & *os*, const Altimetry & *instance*) [friend]

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Pressure reference

Returns

os reference after the operation

13.3.4.19 operator== `bool operator== ( const Altimetry & left, const Altimetry & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.3.5 Member Data Documentation

13.3.5.1 altimetry_map `AltimetryMap woss::Altimetry::altimetry_map [protected]`

altimetry values map

Referenced by Altimetry(), at(), begin(), clear(), createFlat(), createNotValid(), crop(), empty(), end(), eraseValue(), findValue(), getMaxRangeValue(), getMinRangeValue(), insertValue(), isValid(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), randomize(), size(), and sumValue().

13.3.5.2 debug `bool Altimetry::debug = false [static], [protected]`

Debug flag

Referenced by crop(), getDebug(), woss::AltimBretschneider::isValid(), woss::AltimBretschneider::randomize(), setDebug(), timeEvolve(), and woss::AltimBretschneider::timeEvolve().

13.3.5.3 depth `double woss::Altimetry::depth [protected]`

Stores the depth of the scenario [m]

Referenced by woss::AltimBretschneider::AltimBretschneider(), woss::AltimBretschneider::create(), getDepth(), operator=(), and setDepth().

13.3.5.4 evolution_time_quantum `double woss::Altimetry::evolution_time_quantum [protected]`

Stores the last evolution time quantum [s]

Referenced by getEvolutionTimeQuantum(), operator=(), setEvolutionTimeQuantum(), timeEvolve(), and woss::AltimBretschneider::timeEvolve().

13.3.5.5 last_evolution_time `Time woss::Altimetry::last_evolution_time [protected]`

Stores the last evolution's simulation time

Referenced by operator=(), timeEvolve(), and woss::AltimBretschneider::timeEvolve().

13.3.5.6 max_altimetry_value `double woss::Altimetry::max_altimetry_value [protected]`

Stores the maximum altimetry value [m] i.e. the min negative value (if present)

Referenced by Altimetry(), getMaxAltimetryValue(), insertValue(), operator=(), and sumValue().

13.3.5.7 min_altimetry_value `double woss::Altimetry::min_altimetry_value [protected]`

Stores the miminum altimetry value [m] i.e. the maximum positive value (if present)

Referenced by Altimetry(), getMinAltimetryValue(), insertValue(), operator=(), and sumValue().

13.3.5.8 range `double woss::Altimetry::range [protected]`

range [m]

Referenced by Altimetry(), eraseValue(), findValue(), getRange(), initialize(), woss::AltimBretschneider::initialize(), insertValue(), operator=(), setRange(), and sumValue().

13.3.5.9 range_precision `long double woss::Altimetry::range_precision [protected]`

Stores the precision of all PDouble ranges [m]

Referenced by Altimetry(), crop(), getRangePrecision(), initialize(), woss::AltimBretschneider::initialize(), and operator=().

13.3.5.10 total_range_steps `int woss::Altimetry::total_range_steps [protected]`

range total steps

Referenced by woss::AltimBretschneider::AltimBretschneider(), woss::AltimBretschneider::create(), getTotalRangeSteps(), initialize(), woss::AltimBretschneider::initialize(), woss::AltimBretschneider::isValid(), operator=(), and setTotalRangeSteps().

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

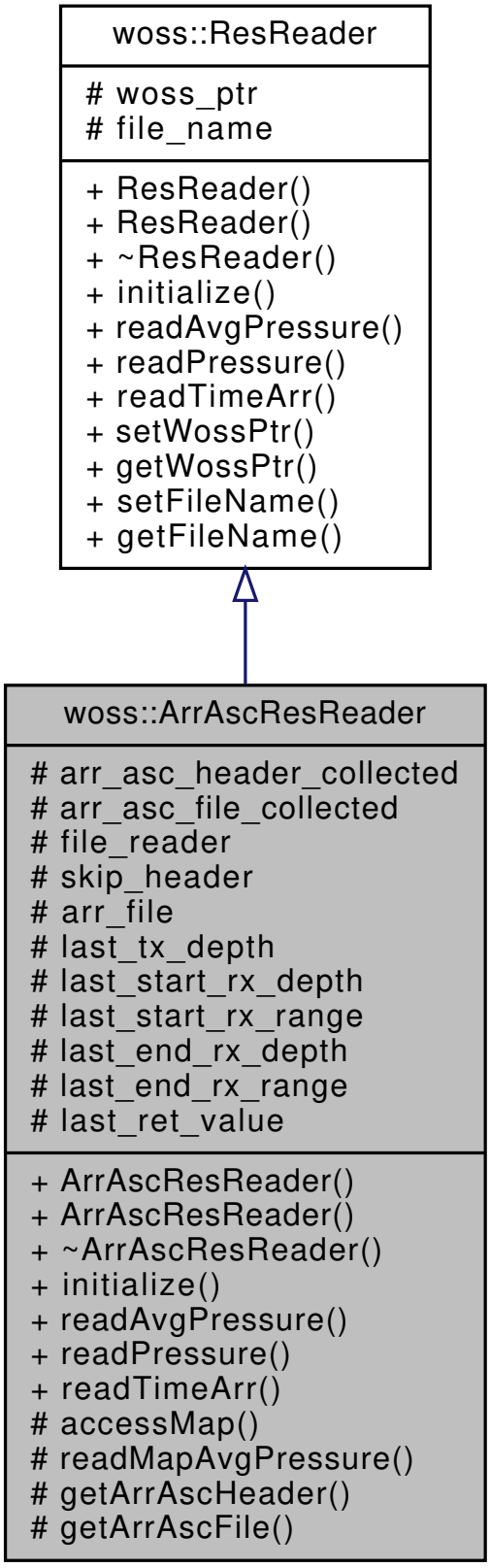
- woss/woss_def/altimetry-definitions.h
- woss/woss_def/altimetry-definitions.cpp

13.4 woss::ArrAscResReader Class Reference

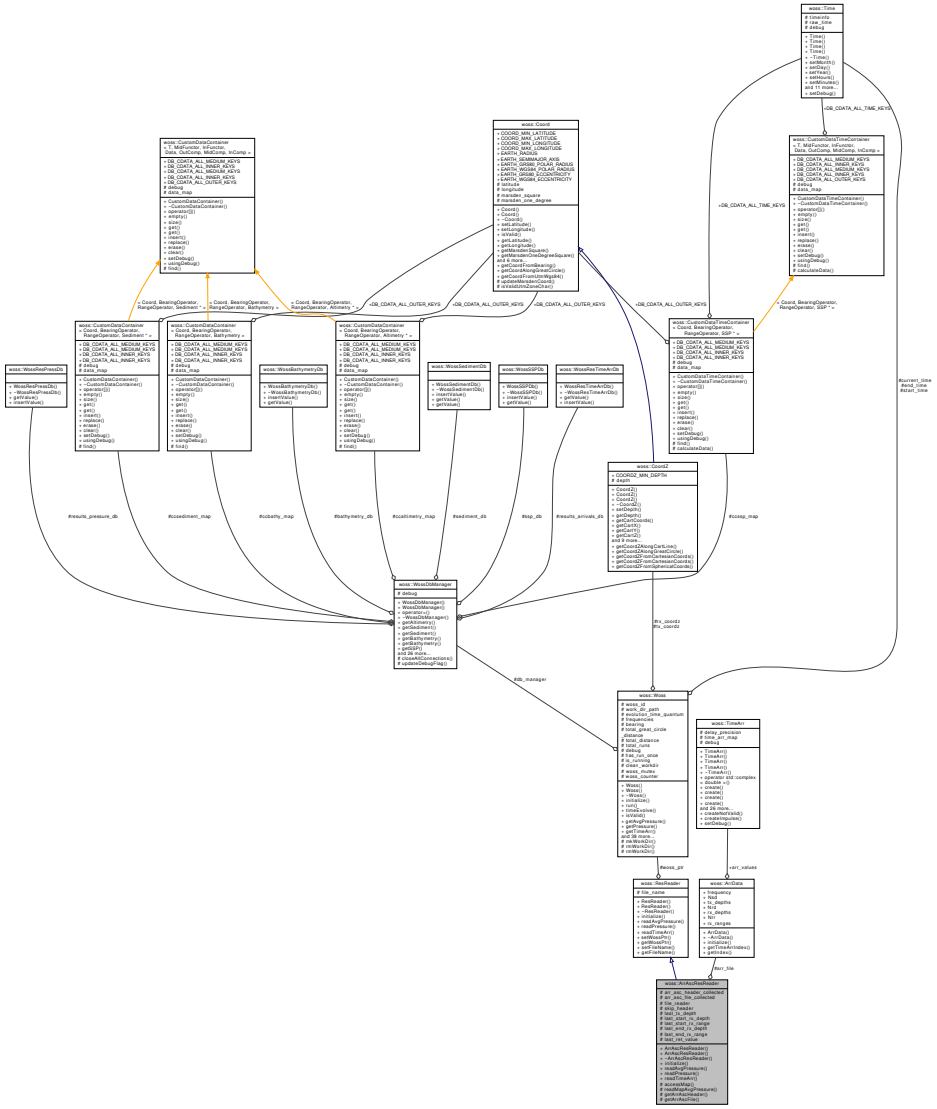
Class for reading and manipulating results provided by any acoustic toolbox textual ARR file.

`#include <ac-toolbox-arr-asc-reader.h>`

Inheritance diagram for woss::ArrAscResReader:



Collaboration diagram for woss::ArrAscResReader:



Public Member Functions

- ArrAscResReader ()
- ArrAscResReader (const Woss ∗const woss)
- virtual bool initialize ()
- virtual Pressure ∗ readAvgPressure (double tx_depth, double start_rx_depth, double start_rx_range, double end_rx_depth, double end_rx_range)
- virtual Pressure ∗ readPressure (double tx_depth, double rx_depth, double rx_range) const
- virtual TimeArr ∗ readTimeArr (double tx_depth, double rx_depth, double rx_range) const

Protected Member Functions

- TimeArr ∗ accessMap (double tx_depth, double rx_depth, double rx_range) const
- ::std::complex< double > readMapAvgPressure (double tx_depth, double start_rx_depth, double start_rx↩ _range, double end_rx_depth, double end_rx_range)
- bool getArrAscHeader ()
- bool getArrAscFile ()

Protected Attributes

- bool arr_asc_header_collected
- bool arr_asc_file_collected
- ::std::ifstream file_reader
- ::std::streampos skip_header
- ArrData arr_file
- double **last_tx_depth**
- double **last_start_rx_depth**
- double **last_start_rx_range**
- double **last_end_rx_depth**
- double **last_end_rx_range**
- ::std::complex< double > **last_ret_value**

13.4.1 Detailed Description

Class for reading and manipulating results provided by any acoustic toolbox textual ARR file.

Class ArrAscResReader stores TimeArr provided by any acoustic toolbox textual ARR file in a ArrData. It also offers TimeArr manipulation and Pressure conversion methods.

13.4.2 Constructor & Destructor Documentation

13.4.2.1 ArrAscResReader() [1/2] ArrAscResReader::ArrAscResReader ( )

ArrAscResReader default constructor

13.4.2.2 ArrAscResReader() [2/2] ArrAscResReader::ArrAscResReader (

```
const Woss *const woss )
```

ArrAscResReader constructor

Parameters

woss	const pointer to a const Woss object

13.4.3 Member Function Documentation

13.4.3.1 accessMap() TimeArr * woss::ArrAscResReader::accessMap (

```
double tx_depth,
double rx_depth,
double rx_range ) const  [inline], [protected]
```

Gets the TimeArr value from ArrData TimeArr array associated to given parameters

Parameters

tx_depth	transmitter depth [m]
rx_depth	start receiver depth [m]
rx_range	start receiver range [m]

Returns

a valid TimeArr value; a not valid TimeArr if arr_file hasn't been read yet

References arr_file, woss::ArrData::arr_values, and woss::ArrData::getTimeArrIndex().

Referenced by readTimeArr().

Here is the call graph for this function:

13.4.3.2 getArrAscFile() `bool ArrAscResReader::getArrAscFile ( ) [protected]`

Process the ARR file data

Returns

true if method was successful, *false* otherwise

References arr_asc_header_collected, woss::ResReader::file_name, file_reader, and skip_header.

Referenced by initialize().

13.4.3.3 getArrAscHeader() `bool ArrAscResReader::getArrAscHeader ( ) [protected]`

Process the ARR file header

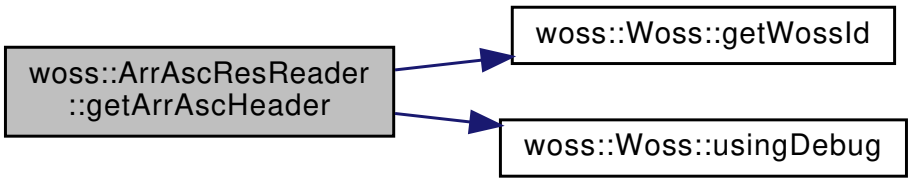
Returns

true if method was successful, *false* otherwise

References arr_asc_header_collected, woss::ResReader::file_name, file_reader, woss::Woss::getWossId(), woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by initialize().

Here is the call graph for this function:



13.4.3.4 initialize() `bool ArrAscResReader::initialize ( ) [virtual]`

Initializes the ArrAscResReader object, reads ARR file, and stores read TimeArr values

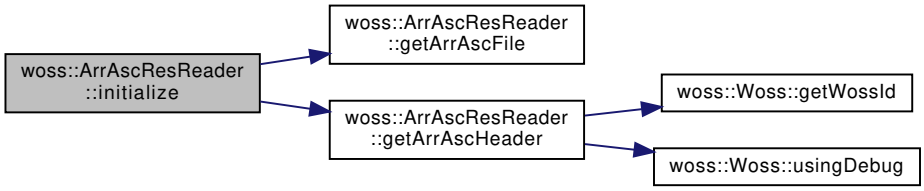
Returns

true if method was successful, *false* otherwise

Implements woss::ResReader.

References woss::ResReader::file_name, getArrAscFile(), getArrAscHeader(), and woss::ResReader::woss_ptr.

Here is the call graph for this function:



13.4.3.5 readAvgPressure() `Pressure * ArrAscResReader::readAvgPressure ( double tx_depth, double start_rx_depth, double start_rx_range, double end_rx_depth, double end_rx_range ) [virtual]`

Gets the average Pressure value in given rx range-depth box. Returned Pressure is the coherent sum of the computed average TimeArr

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

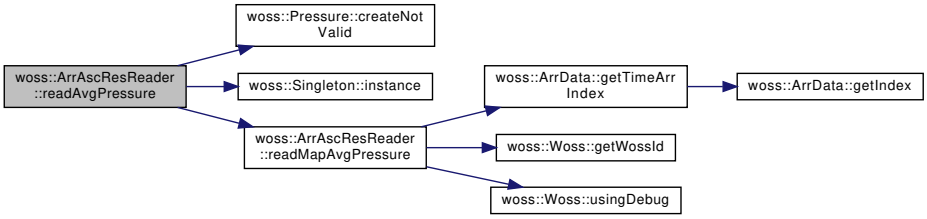
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

Implements woss::ResReader.

References arr_asc_file_collected, woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), and readMapAvgPressure().

Here is the call graph for this function:



13.4.3.6 readMapAvgPressure()

```
std::complex< double > ArrAscResReader::readMapAvgPressure (
        double tx_depth,
        double start_rx_depth,
        double start_rx_range,
        double end_rx_depth,
        double end_rx_range )  [protected]
```

Gets the average Pressure value in given rx range-depth box converted from ArrData TimeArr array

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

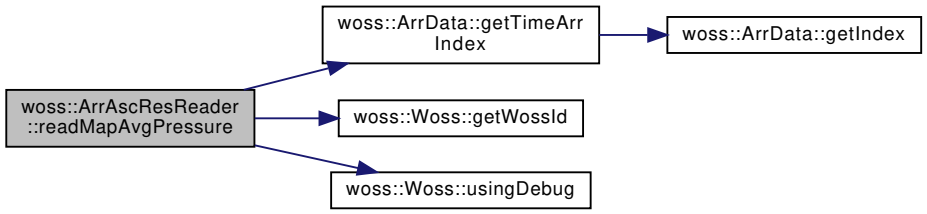
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

References arr_file, woss::ArrData::arr_values, woss::ArrData::getTimeArrIndex(), woss::Woss::getWossId(), woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by readAvgPressure().

Here is the call graph for this function:



13.4.3.7 readPressure()

```
Pressure * ArrAscResReader::readPressure (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

Gets a Pressure value for given range, depths. Returned Pressure is the coherent sum of the computed TimeArr

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

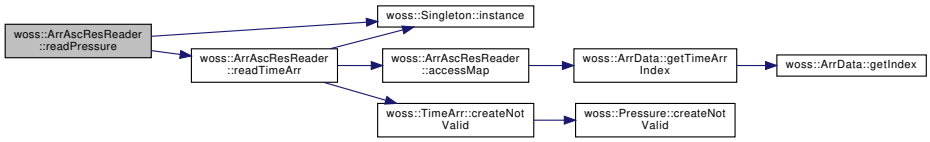
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

Implements woss::ResReader.

References woss::Singleton< T >::instance(), and readTimeArr().

Here is the call graph for this function:



13.4.3.8 readTimeArr()

```
TimeArr * ArrAscResReader::readTimeArr (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

Gets a TimeArr value for given range, depths

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

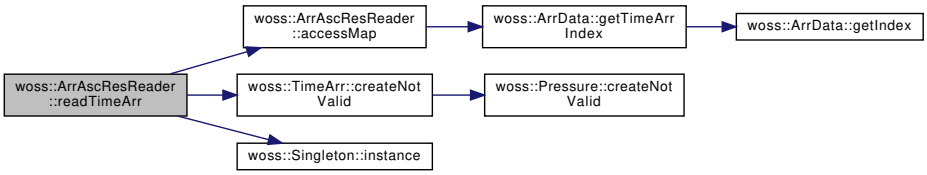
a valid TimeArr value; a not valid TimeArr if arr_file hasn't been read yet

Implements woss::ResReader.

References accessMap(), arr_asc_file_collected, woss::TimeArr::createNotValid(), and woss::Singleton< T >::instance().

Referenced by readPressure().

Here is the call graph for this function:



13.4.4 Member Data Documentation

13.4.4.1 arr_asc_file_collected `bool woss::ArrAscResReader::arr_asc_file_collected [protected]`

Boolean associated to the reading of ARR file data

Referenced by readAvgPressure(), and readTimeArr().

13.4.4.2 arr_asc_header_collected `bool woss::ArrAscResReader::arr_asc_header_collected [protected]`

Boolean associated to the reading of ARR file header

Referenced by getArrAscFile(), and getArrAscHeader().

13.4.4.3 arr_file `ArrData woss::ArrAscResReader::arr_file [protected]`

Struct that holds TimeArr data read from ARR file

Referenced by accessMap(), and readMapAvgPressure().

13.4.4.4 file_reader `::std::ifstream woss::ArrAscResReader::file_reader [protected]`

Input file stream

Referenced by getArrAscFile(), and getArrAscHeader().

13.4.4.5 skip_header `::std::streampos woss::ArrAscResReader::skip_header [protected]`

Total lines to skip if ARR file header has already been read

Referenced by getArrAscFile().

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

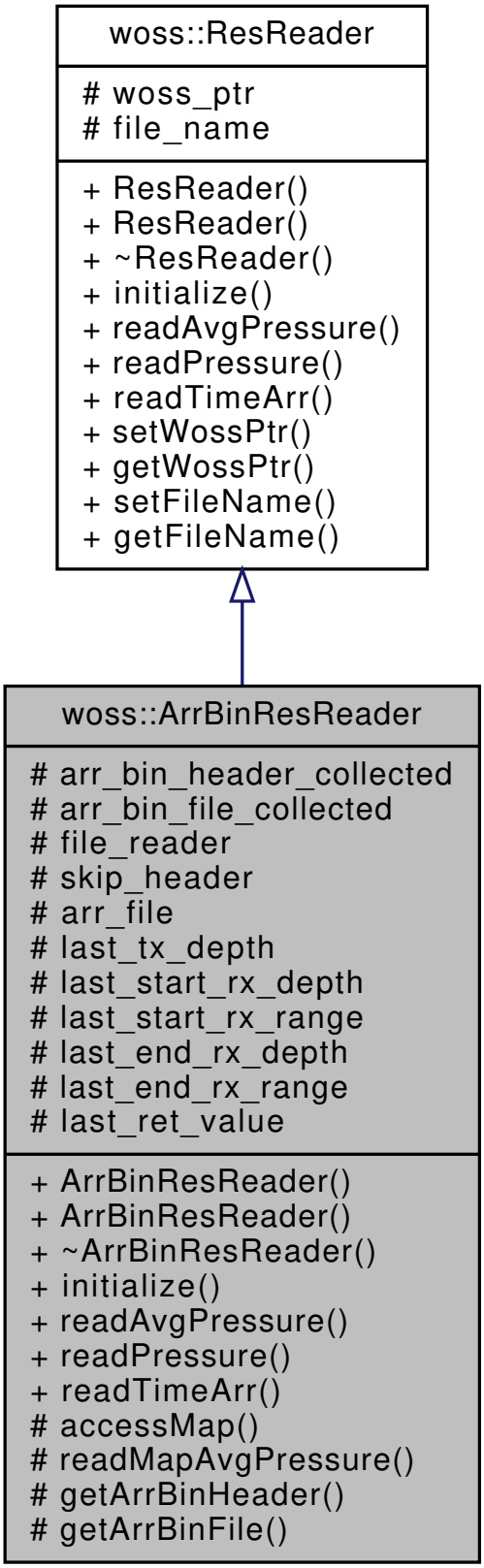
- woss/ac-toolbox-arr-asc-reader.h
- woss/ac-toolbox-arr-asc-reader.cpp

13.5 woss::ArrBinResReader Class Reference

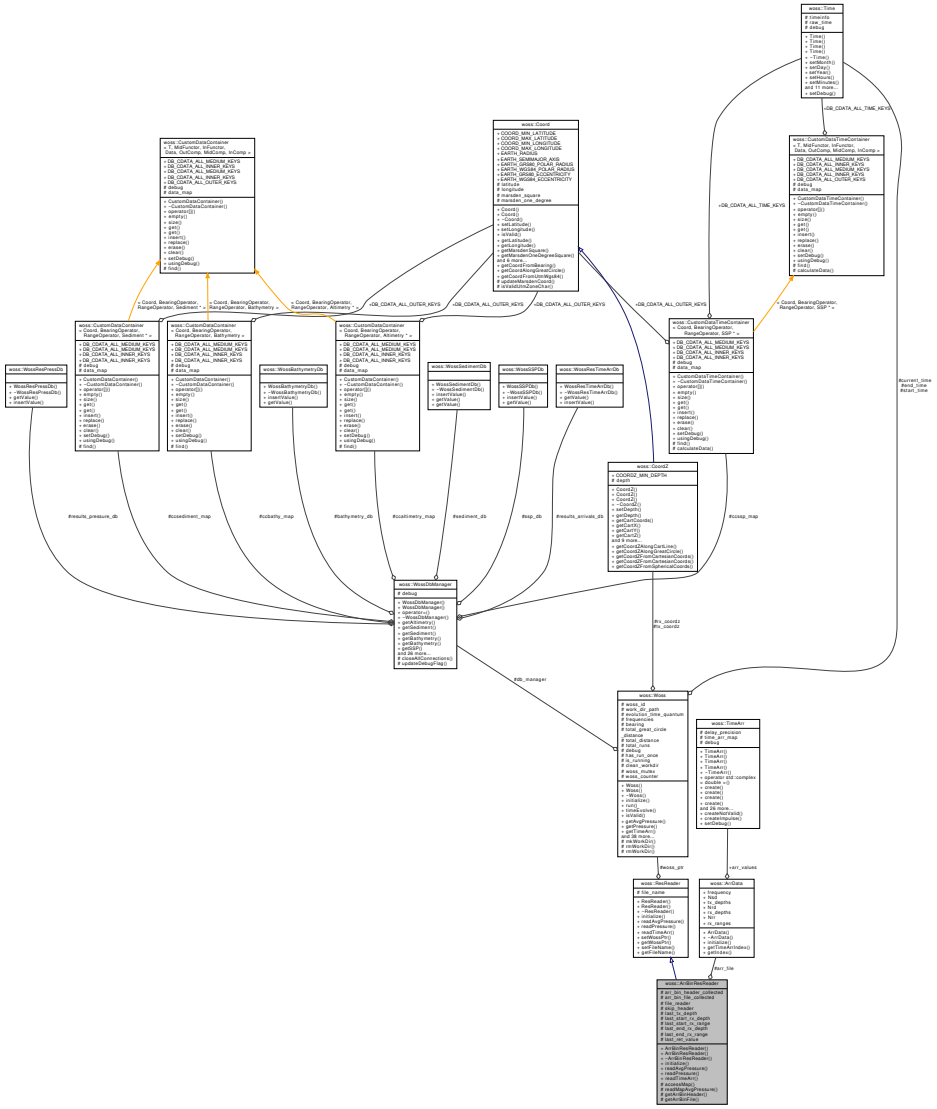
Class for reading and manipulating results provided by any acoustic toolbox binary ARR file.

`#include <ac-toolbox-arr-bin-reader.h>`

Inheritance diagram for woss::ArrBinResReader:



Collaboration diagram for woss::ArrBinResReader:



Public Member Functions

- ArrBinResReader ()
- ArrBinResReader (const Woss ∗const woss)
- virtual bool initialize ()
- virtual Pressure ∗ readAvgPressure (double tx_depth, double start_rx_depth, double start_rx_range, double end_rx_depth, double end_rx_range)
- virtual Pressure ∗ readPressure (double tx_depth, double rx_depth, double rx_range) const
- virtual TimeArr ∗ readTimeArr (double tx_depth, double rx_depth, double rx_range) const

Protected Member Functions

- TimeArr ∗ accessMap (double tx_depth, double rx_depth, double rx_range) const
- ::std::complex< double > readMapAvgPressure (double tx_depth, double start_rx_depth, double start_rx↩ _range, double end_rx_depth, double end_rx_range)
- bool getArrBinHeader ()
- bool getArrBinFile ()

Protected Attributes

- bool arr_bin_header_collected
- bool arr_bin_file_collected
- ::std::ifstream file_reader
- ::std::streampos skip_header
- ArrData arr_file
- double **last_tx_depth**
- double **last_start_rx_depth**
- double **last_start_rx_range**
- double **last_end_rx_depth**
- double **last_end_rx_range**
- ::std::complex< double > **last_ret_value**

13.5.1 Detailed Description

Class for reading and manipulating results provided by any acoustic toolbox binary ARR file.

Class ArrAscResReader stores TimeArr provided by any acoustic toolbox binary ARR file in a ArrStruct. It also offers TimeArr manipulation and Pressure conversion methods.

13.5.2 Constructor & Destructor Documentation

13.5.2.1 ArrBinResReader() [1/2] ArrBinResReader::ArrBinResReader ( )

ArrBinResReader default constructor

13.5.2.2 ArrBinResReader() [2/2] ArrBinResReader::ArrBinResReader (

```
const Woss *const woss )
```

ArrBinResReader constructor

Parameters

woss	const pointer to a const Woss object

13.5.3 Member Function Documentation

13.5.3.1 accessMap() TimeArr * woss::ArrBinResReader::accessMap (

```
double tx_depth,
double rx_depth,
double rx_range ) const  [inline], [protected]
```

Gets the TimeArr value from ArrData TimeArr array associated to given parameters

Parameters

tx_depth	transmitter depth [m]
rx_depth	start receiver depth [m]
rx_range	start receiver range [m]

Returns

a valid TimeArr value; a not valid TimeArr if arr_file hasn't been read yet

References arr_file, woss::ArrData::arr_values, and woss::ArrData::getTimeArrIndex().

Referenced by readTimeArr().

Here is the call graph for this function:

13.5.3.2 getArrBinFile() `bool ArrBinResReader::getArrBinFile ( ) [protected]`

Process the ARR file data

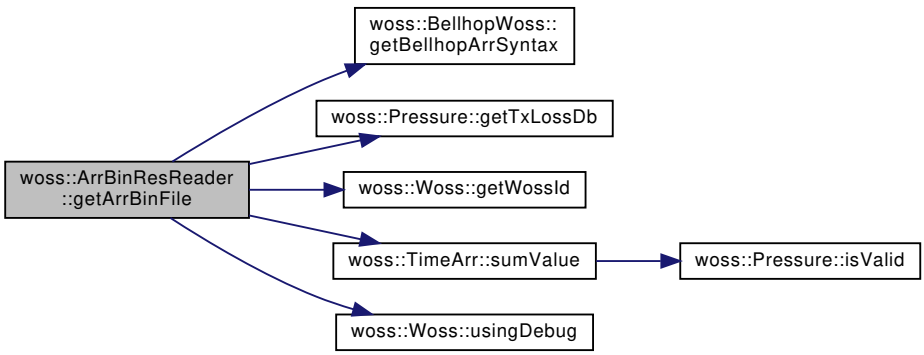
Returns

true if method was successful, *false* otherwise

References arr_bin_file_collected, arr_bin_header_collected, arr_file, woss::ArrData::arr_values, woss::BELLHOP_CREATOR_ARR_
woss::BELLHOP_CREATOR_ARR_FILE_SYNTAX_1, woss::BELLHOP_CREATOR_ARR_FILE_SYNTAX_2,
woss::ResReader::file_name, file_reader, woss::ArrData::frequency, woss::BellhopWoss::getBellhopArrSyntax(),
woss::Pressure::getTxLossDb(), woss::Woss::getWossId(), woss::ArrData::Nrd, woss::ArrData::Nrr, woss::ArrData::Nsd,
woss::ArrData::rx_depths, woss::ArrData::rx_ranges, skip_header, woss::TimeArr::sumValue(), woss::ArrData::tx_depths,
woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by initialize().

Here is the call graph for this function:



13.5.3.3 getArrBinHeader() `bool ArrBinResReader::getArrBinHeader ( ) [protected]`

Process the ARR file header

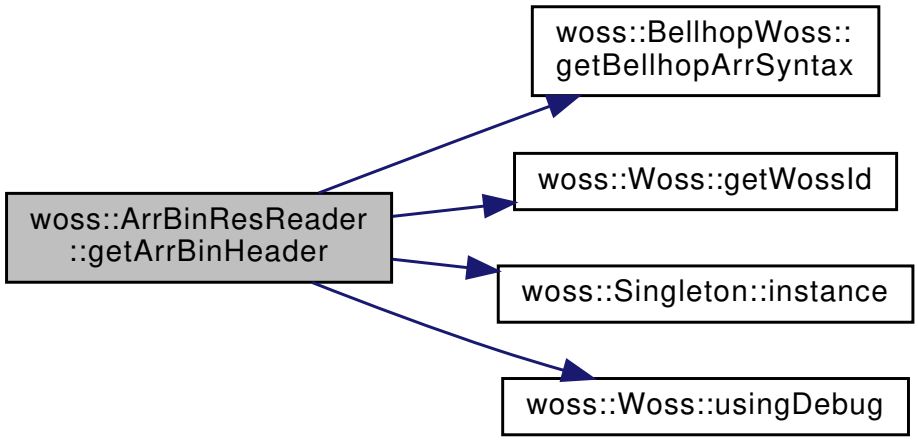
Returns

true if method was successful, *false* otherwise

References arr_bin_header_collected, arr_file, woss::ArrData::arr_values, woss::BELLHOP_CREATOR_ARR_FILE_SYNTAX_2, woss::ResReader::file_name, file_reader, woss::ArrData::frequency, woss::BellhopWoss::getBellhopArrSyntax(), woss::Woss::getWossId(), woss::Singleton< T >::instance(), woss::ArrData::Nrd, woss::ArrData::Nrr, woss::ArrData::Nsd, woss::ArrData::rx_depths, woss::ArrData::rx_ranges, skip_header, woss::ArrData::tx_depths, woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by initialize().

Here is the call graph for this function:



13.5.3.4 initialize() `bool ArrBinResReader::initialize ( ) [virtual]`

Initializes the ArrBinResReader object, reads ARR file, and stores read TimeArr values

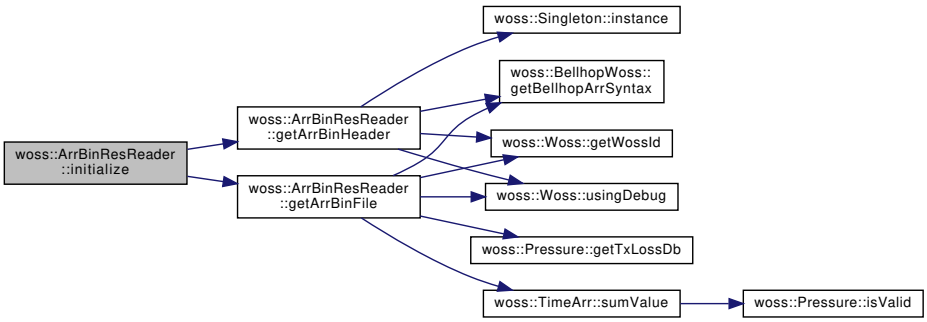
Returns

true if method was successful, *false* otherwise

Implements woss::ResReader.

References woss::ResReader::file_name, getArrBinFile(), getArrBinHeader(), and woss::ResReader::woss_ptr.

Here is the call graph for this function:



13.5.3.5 readAvgPressure()

```
Pressure * ArrBinResReader::readAvgPressure (
        double tx_depth,
        double start_rx_depth,
        double start_rx_range,
        double end_rx_depth,
        double end_rx_range ) [virtual]
```

Gets the average Pressure value in given rx range-depth box. Returned Pressure is the coherent sum of the computed average TimeArr

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

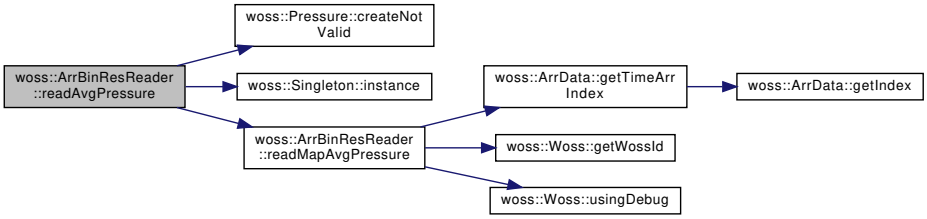
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

Implements woss::ResReader.

References arr_bin_file_collected, woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), and readMapAvgPressure().

Here is the call graph for this function:



13.5.3.6 readMapAvgPressure()

```
std::complex< double > ArrBinResReader::readMapAvgPressure (
        double tx_depth,
        double start_rx_depth,
        double start_rx_range,
        double end_rx_depth,
        double end_rx_range ) [protected]
```

Gets the average Pressure value in given rx range-depth box converted from ArrStruct TimeArr array

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

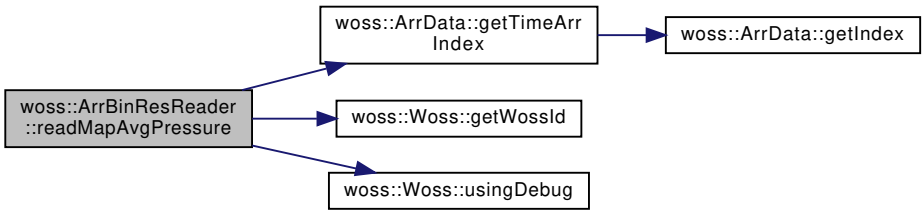
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

References arr_file, woss::ArrData::arr_values, woss::ArrData::getTimeArrIndex(), woss::Woss::getWossId(), woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by readAvgPressure().

Here is the call graph for this function:



13.5.3.7 readPressure()

```
Pressure * ArrBinResReader::readPressure (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

Gets a Pressure value for given range, depths. Returned Pressure is the coherent sum of the computed TimeArr

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

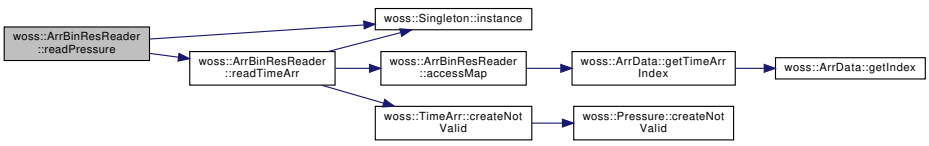
Returns

a valid Pressure value; a not valid Pressure if arr_file hasn't been read yet

Implements woss::ResReader.

References woss::Singleton< T >::instance(), and readTimeArr().

Here is the call graph for this function:



13.5.3.8 readTimeArr()

```
TimeArr * ArrBinResReader::readTimeArr (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

Gets a TimeArr value for given range, depths

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

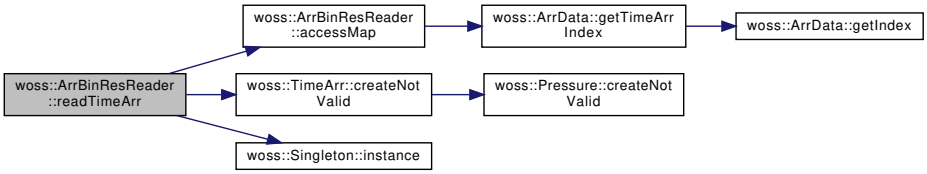
a valid TimeArr value; a not valid TimeArr if arr_file hasn't been read yet

Implements woss::ResReader.

References accessMap(), arr_bin_file_collected, woss::TimeArr::createNotValid(), and woss::Singleton< T >::instance().

Referenced by readPressure().

Here is the call graph for this function:



13.5.4 Member Data Documentation

13.5.4.1 arr_bin_file_collected

```
bool woss::ArrBinResReader::arr_bin_file_collected  [protected]
```

Boolean associated to the reading of ARR file data

Referenced by getArrBinFile(), readAvgPressure(), and readTimeArr().

13.5.4.2 arr_bin_header_collected

```
bool woss::ArrBinResReader::arr_bin_header_collected  [protected]
```

Boolean associated to the reading of ARR file header

Referenced by getArrBinFile(), and getArrBinHeader().

13.5.4.3 arr_file `ArrData woss::ArrBinResReader::arr_file [protected]`

Struct that holds TimeArr data read from ARR file

Referenced by accessMap(), getArrBinFile(), getArrBinHeader(), and readMapAvgPressure().

13.5.4.4 file_reader `::std::ifstream woss::ArrBinResReader::file_reader [protected]`

Input file stream

Referenced by getArrBinFile(), and getArrBinHeader().

13.5.4.5 skip_header `::std::streampos woss::ArrBinResReader::skip_header [protected]`

Total lines to skip if ARR file header has already been read

Referenced by getArrBinFile(), and getArrBinHeader().

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

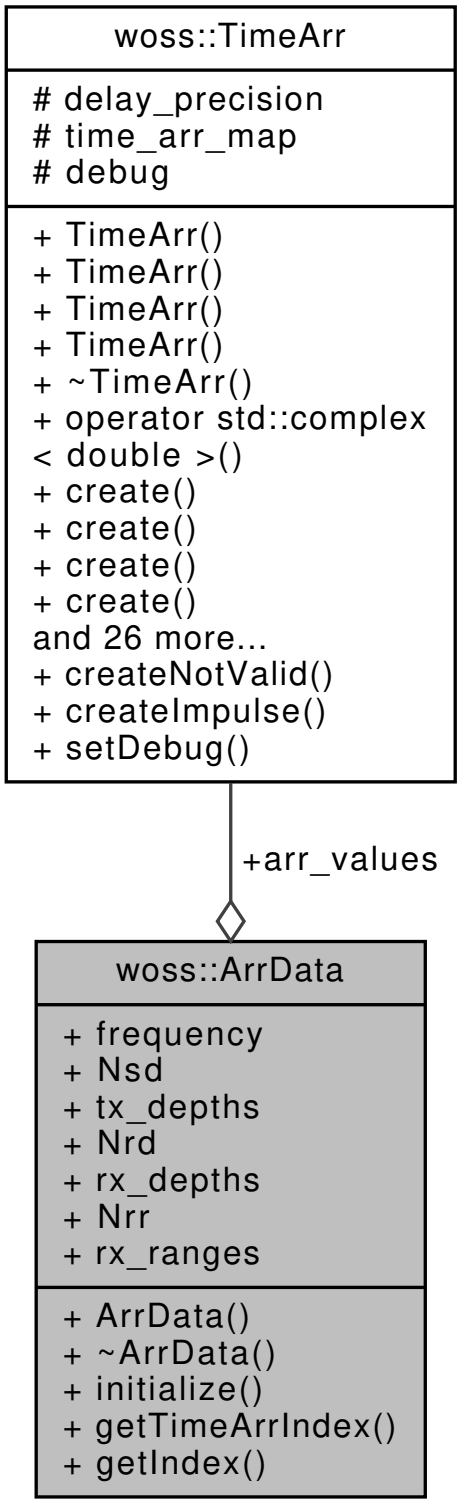
- woss/ac-toolbox-arr-bin-reader.h
- woss/ac-toolbox-arr-bin-reader.cpp

13.6 woss::ArrData Class Reference

class for storing data of any acoustic toolbox ARR file

`#include <ac-toolbox-arr-asc-reader.h>`

Collaboration diagram for woss::ArrData:



Public Member Functions

- ∼ArrData ()
- void initialize ()
- int getTimeArrIndex (double tx_depth, double rx_depth, double rx_range) const
- int getIndex (float value, float ∗array, int array_size) const

Public Attributes

- float frequency
- int32_t Nsd
- float * tx_depths
- int32_t Nrd
- float * rx_depths
- int32_t Nrr
- float * rx_ranges
- TimeArr * arr_values

13.6.1 Detailed Description

class for storing data of any acoustic toolbox ARR file

class ArrData stores TimeArr values provided by any acoustic toolbox ARR file

13.6.2 Constructor & Destructor Documentation

13.6.2.1 ~ArrData() `woss::ArrData::~ArrData ( ) [inline]`

Destructor

References arr_values, rx_depths, rx_ranges, and tx_depths.

13.6.3 Member Function Documentation

13.6.3.1 getIndex()

```
int ArrData::getIndex (
        float value,
        float * array,
        int array_size ) const
```

Returns the index of given array associated to given value

Parameters

value	test value
array	valid pointer to an array
array_size	size of passed array

Returns

valid array index value

Referenced by getTimeArrIndex().

13.6.3.2 getTimeArrIndex()

```
int ArrData::getTimeArrIndex (
        double tx_depth,
        double rx_depth,
        double rx_range ) const
```

Returns the arr_values index associated to given parameters

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

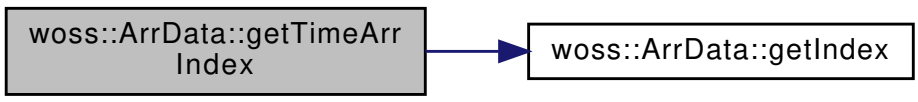
Returns

valid arr_values index value

References getIndex(), Nrd, Nrr, Nsd, rx_depths, rx_ranges, and tx_depths.

Referenced by woss::ArrAscResReader::accessMap(), woss::ArrBinResReader::accessMap(), woss::ArrAscResReader::readMapAvg and woss::ArrBinResReader::readMapAvgPressure().

Here is the call graph for this function:



13.6.3.3 initialize()

```
void woss::ArrData::initialize ( )  [inline]
```

Initializes the struct

References arr_values, frequency, Nrd, Nrr, Nsd, rx_depths, rx_ranges, and tx_depths.

13.6.4 Member Data Documentation

13.6.4.1 arr_values

```
TimeArr* woss::ArrData::arr_values
```

Pointer to an array of TimeArr values [m]

Referenced by woss::ArrAscResReader::accessMap(), woss::ArrBinResReader::accessMap(), woss::ArrBinResReader::getArrBinFile woss::ArrBinResReader::getArrBinHeader(), initialize(), woss::ArrAscResReader::readMapAvgPressure(), woss::ArrBinResReader::readMapAvgPressure(), and ~ArrData().

13.6.4.2 frequency `float woss::ArrData::frequency`

Frequency value [Hz]

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), and initialize().

13.6.4.3 Nrd `int32_t woss::ArrData::Nrd`

Total number of receiver depths

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), and initialize().

13.6.4.4 Nrr `int32_t woss::ArrData::Nrr`

Total number of receiver ranges

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), and initialize().

13.6.4.5 Nsd `int32_t woss::ArrData::Nsd`

Total number of transmitter depths

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), and initialize().

13.6.4.6 rx_depths `float* woss::ArrData::rx_depths`

Pointer to an array of receiver depths [m]

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), initialize(), and ~ArrData().

13.6.4.7 rx_ranges `float* woss::ArrData::rx_ranges`

Pointer to an array of receiver ranges [m]

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), initialize(), and ~ArrData().

13.6.4.8 tx_depths `float* woss::ArrData::tx_depths`

Pointer to an array of transmitter depths [m]

Referenced by woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getArrBinHeader(), getTimeArrIndex(), initialize(), and ~ArrData().

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

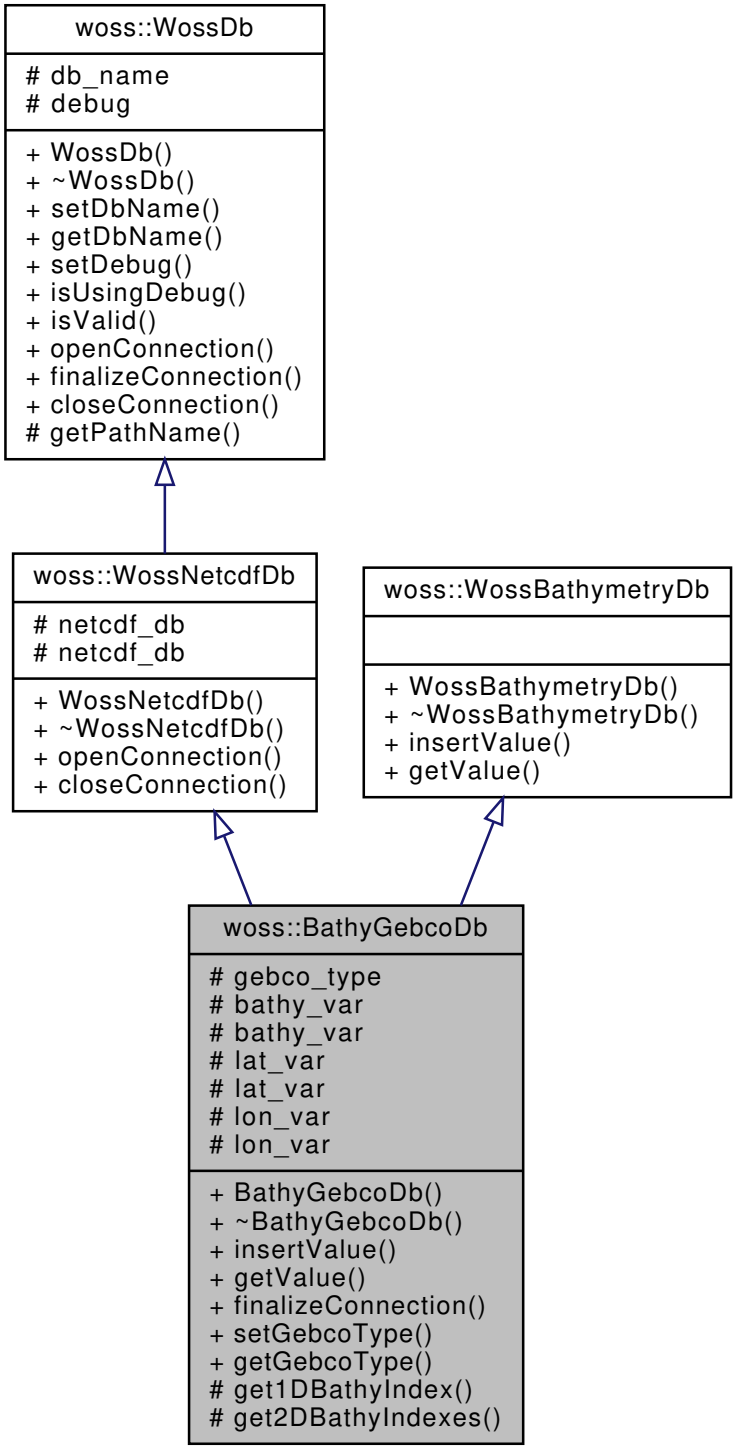
- woss/ac-toolbox-arr-asc-reader.h
- woss/ac-toolbox-arr-asc-reader.cpp

13.7 woss::BathyGebcoDb Class Reference

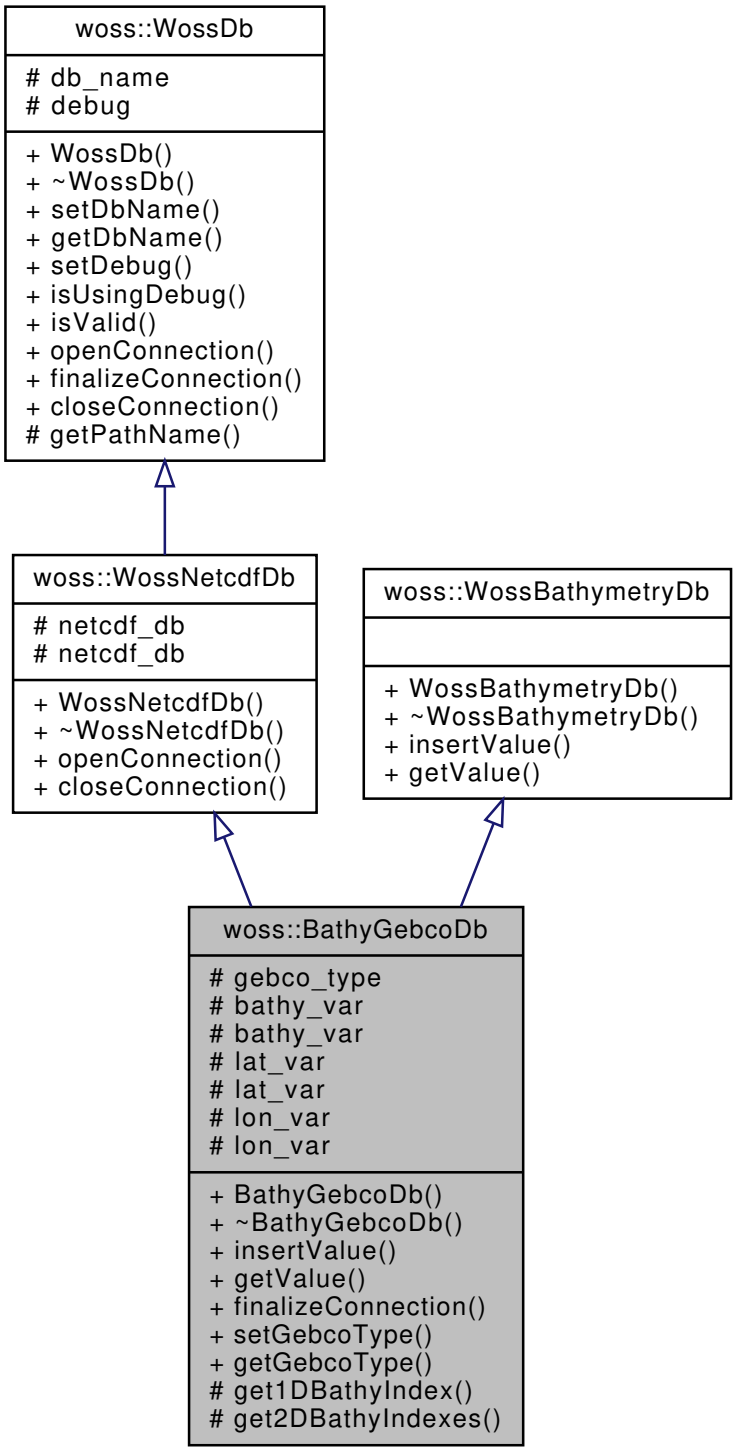
NetCDF specialization of WossNetcdfDb for GEBCO database.

`#include <bathymetry-gebco-db.h>`

Inheritance diagram for woss::BathyGebcoDb:



Collaboration diagram for woss::BathyGebcoDb:



Public Member Functions

- BathyGebcoDb (const ::std::string &name)
- virtual bool insertValue (const Coord &coordinates, const Bathymetry &bathymetry_value)
- virtual double getValue (const Coord &coords) const
- virtual bool finalizeConnection ()
- void setGebcoType (const GEBCO_BATHY_TYPE &type)
- GEBCO_BATHY_TYPE getGebcoType ()

Protected Member Functions

- long get1DBathyIndex (const Coord &coords) const
- Gebco2DIndexes get2DBathyIndexes (const Coord &coords) const

Protected Attributes

- GEBCO_BATHY_TYPE gebco_type
- netCDF::NcVar bathy_var
- NcVar * **bathy_var**
- netCDF::NcVar lat_var
- NcVar * **lat_var**
- netCDF::NcVar lon_var
- NcVar * **lon_var**

13.7.1 Detailed Description

NetCDF specialization of WossNetcdfDb for GEBCO database.

NetCDF specialization of WossNetcdfDb for GEBCO database. It creates a NetCDF variable used to get requested bathymetry values

13.7.2 Constructor & Destructor Documentation

13.7.2.1 BathyGebcoDb() `BathyGebcoDb::BathyGebcoDb ( const ::std::string & name )`

BathyGebcoDb constructor

Parameters

name	pathname of database

References woss::GEBCO_2D_30_SECONDS_BATHY_TYPE.

13.7.3 Member Function Documentation

13.7.3.1 finalizeConnection() `bool BathyGebcoDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions. It create and initializes a NetCDF variable

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References bathy_var, woss::WossDb::debug, woss::GEBCO_1D_1_MINUTE_BATHY_TYPE, woss::GEBCO_1D_30_SECONDS_B
gebco_type, lat_var, lon_var, and woss::WossNetcdfDb::netcdf_db.

13.7.3.2 get1DBathyIndex() `long BathyGebcoDb::get1DBathyIndex ( const Coord & coords ) const [protected]`

Returns the GEBCO 1D index corresponding the given coordinates. This index will be used to access the NetCDF variable and thus retrieving the bathymetry value

Parameters

coords	const reference to a valid Coord object

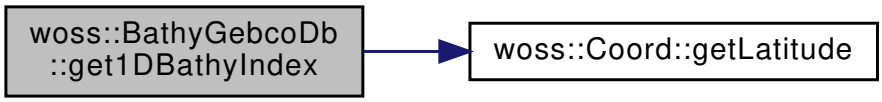
Returns

index value

References woss::GEBCO_1D_1_MINUTE_BATHY_TYPE, gebco_type, and woss::Coord::getLatitude().

Referenced by getValue().

Here is the call graph for this function:



13.7.3.3 get2DBathyIndexes() `Gebco2DIndexes BathyGebcoDb::get2DBathyIndexes ( const Coord & coords ) const [protected]`

Returns the GEBCO 2D indexes corresponding the given coordinates. These indexes will be used to access the NetCDF variable and thus retrieving the bathymetry value

Parameters

coords	const reference to a valid Coord object

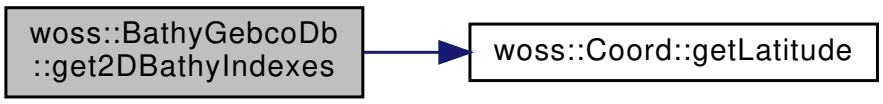
Returns

Gebco2DIndexes value

References woss::GEBCO_2D_1_MINUTE_BATHY_TYPE, gebco_type, and woss::Coord::getLatitude().

Referenced by getValue().

Here is the call graph for this function:



13.7.3.4 getGebcoType() `GEBCO_BATHY_TYPE woss::BathyGebcoDb::getGebcoType ( ) [inline]`

Returns which GEBCO version is in use #return GEBCO_BATHY_TYPE instance

References gebco_type.

13.7.3.5 getValue() `double BathyGebcoDb::getValue ( const Coord & coords ) const [virtual]`

Returns the positive depth value (bathymetry) of given coordinates, if present in the database. If given coordinates are on land (original retrieved value is positive) HUGE_VAL is returned.

Parameters

coords	const reference to a valid Coord object

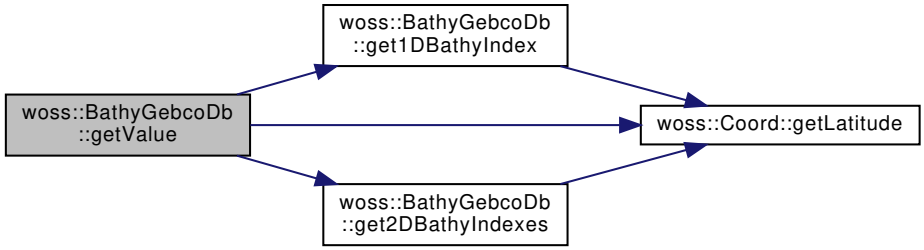
Returns

positive depth value [m] if coordinates are found, *HUGE_VAL* otherwise

Implements woss::WossBathymetryDb.

References bathy_var, woss::WossDb::debug, woss::GEBCO_1D_1_MINUTE_BATHY_TYPE, woss::GEBCO_1D_30_SECONDS_B
woss::GEBCO_2D_15_SECONDS_BATHY_TYPE, woss::GEBCO_2D_1_MINUTE_BATHY_TYPE, woss::GEBCO_2D_30_SECOND
gebco_type, get1DBathyIndex(), get2DBathyIndexes(), woss::Coord::getLatitude(), lat_var, and lon_var.

Here is the call graph for this function:



13.7.3.6 insertValue()

```
bool BathyGebcoDb::insertValue (
        const Coord & coordinates,
        const Bathymetry & bathymetry_value )  [virtual]
```

Inserts the given woss::Bathymetry value in the database for given coordinates

Parameters

coordinates	const reference to a valid Coord object
bathymetry_value	const reference to woss::Bathymetry value to be inserted

Returns

true if method was successful, *false* otherwise

Implements woss::WossBathymetryDb.

13.7.3.7 setGebcoType()

```
void woss::BathyGebcoDb::setGebcoType (
        const GEBCO_BATHY_TYPE & type )  [inline]
```

Notify the database which GEBCO version is in use

Parameters

type	GEBCO_BATHY_TYPE instance

References gebco_type.

Referenced by woss::BathyGebcoDbCreator::createWossDb().

13.7.4 Member Data Documentation

13.7.4.1 bathy_var `netCDF::NcVar woss::BathyGebcoDb::bathy_var [protected]`

NetCDF bathymetry variable

Referenced by finalizeConnection(), and getValue().

13.7.4.2 gebco_type `GEBCO_BATHY_TYPE woss::BathyGebcoDb::gebco_type [protected]`

GEBCO version in use

Referenced by finalizeConnection(), get1DBathyIndex(), get2DBathyIndexes(), getGebcoType(), getValue(), and setGebcoType().

13.7.4.3 lat_var `netCDF::NcVar woss::BathyGebcoDb::lat_var [protected]`

NetCDF latitude variable

Referenced by finalizeConnection(), and getValue().

13.7.4.4 lon_var `netCDF::NcVar woss::BathyGebcoDb::lon_var [protected]`

NetCDF longitude variable

Referenced by finalizeConnection(), and getValue().

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

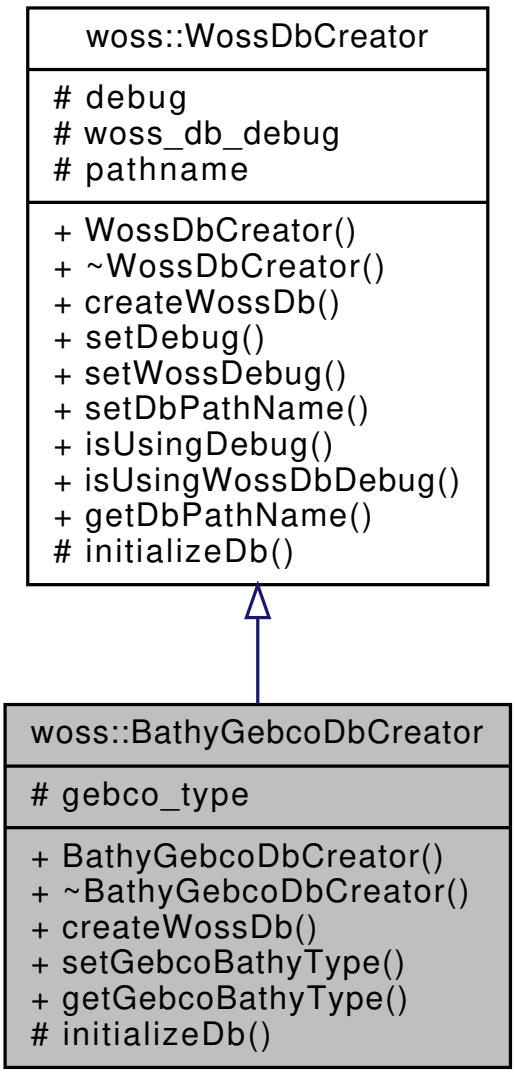
- woss/woss_db/bathymetry-gebco-db.h
- woss/woss_db/bathymetry-gebco-db.cpp

13.8 woss::BathyGebcoDbCreator Class Reference

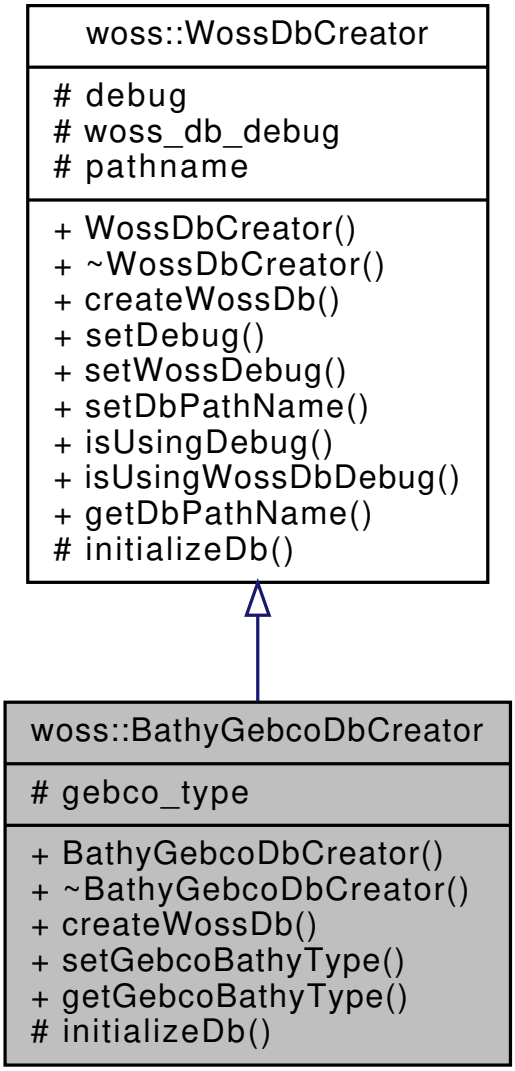
WossDbCreator for the GEBCO bathymetry database.

`#include <bathymetry-gebco-db-creator.h>`

Inheritance diagram for woss::BathyGebcoDbCreator:



Collaboration diagram for woss::BathyGebcoDbCreator:



Public Member Functions

- BathyGebcoDbCreator ()
- virtual WossDb ∗const createWossDb ()
- BathyGebcoDbCreator & setGebcoBathyType (GEBCO_BATHY_TYPE bathy_type)
- GEBCO_BATHY_TYPE getGebcoBathyType ()

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- GEBCO_BATHY_TYPE gebco_type

13.8.1 Detailed Description

WossDbCreator for the GEBCO bathymetry database.

Specialization of WossDbCreator for the GEBCO bathymetry database.

It also provides a Tcl interpreter for NS-2 implementation.

13.8.2 Constructor & Destructor Documentation

13.8.2.1 BathyGebcoDbCreator() `BathyGebcoDbCreator::BathyGebcoDbCreator ( )`

Default BathyGebcoDbCreator constructor

References woss::GEBCO_2D_30_SECONDS_BATHY_TYPE.

13.8.3 Member Function Documentation

13.8.3.1 createWossDb() `WossDb *const BathyGebcoDbCreator::createWossDb ( ) [virtual]`

Creates and initialize a BathyGebcoDb object

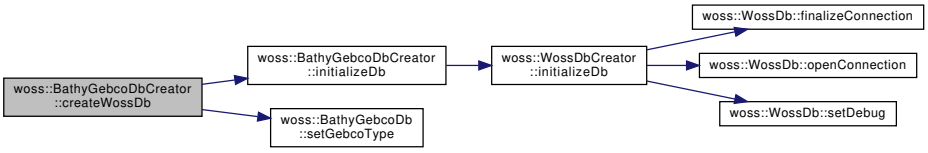
Returns

a pointer to a properly initialized BathyGebcoDb object

Implements woss::WossDbCreator.

References gebco_type, initializeDb(), woss::WossDbCreator::pathname, and woss::BathyGebcoDb::setGebcoType().

Here is the call graph for this function:



13.8.3.2 getGebcoBathyType() `GEBCO_BATHY_TYPE woss::BathyGebcoDbCreator::getGebcoBathyType ( ) [inline]`

Gets the GEBCO_BATHY_TYPE related to the netcdf db that will be opened

Returns

gebco_type

References gebco_type.

13.8.3.3 initializeDb() `bool BathyGebcoDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created BathyGebcoDb

Returns

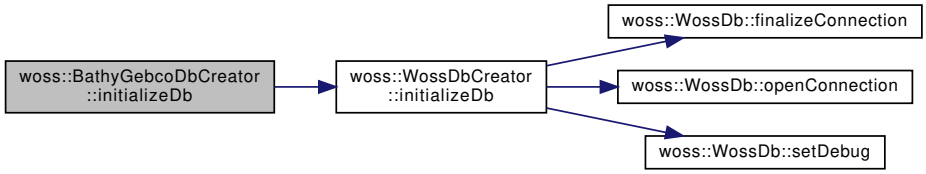
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



13.8.3.4 setGebcoBathyType()

```
BathyGebcoDbCreator & woss::BathyGebcoDbCreator::setGebcoBathyType (
            GEBCO_BATHY_TYPE bathy_type ) [inline]
```

Sets the GEBCO_BATHY_TYPE related to the netcdf db that will be opened

Parameters

bathy_type	netcdf file format

Returns

reference to ∗**this**

References gebco_type.

13.8.4 Member Data Documentation

13.8.4.1 gebco_type GEBCO_BATHY_TYPE woss::BathyGebcoDbCreator::gebco_type [protected]

GEBCO version in use

See also

GEBCO_BATHY_TYPE

Referenced by createWossDb(), getGebcoBathyType(), and setGebcoBathyType().

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

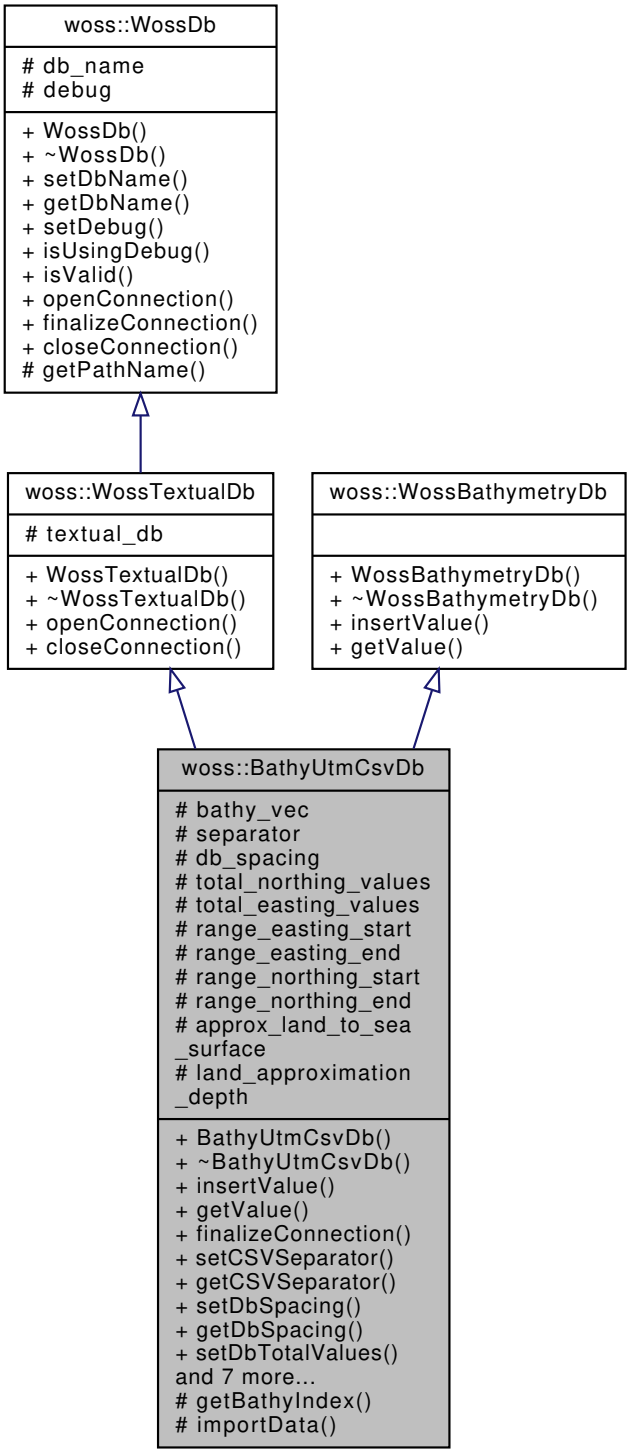
- woss/woss_db/bathymetry-gebco-db-creator.h
- woss/woss_db/bathymetry-gebco-db-creator.cpp

13.9 woss::BathyUtmCsvDb Class Reference

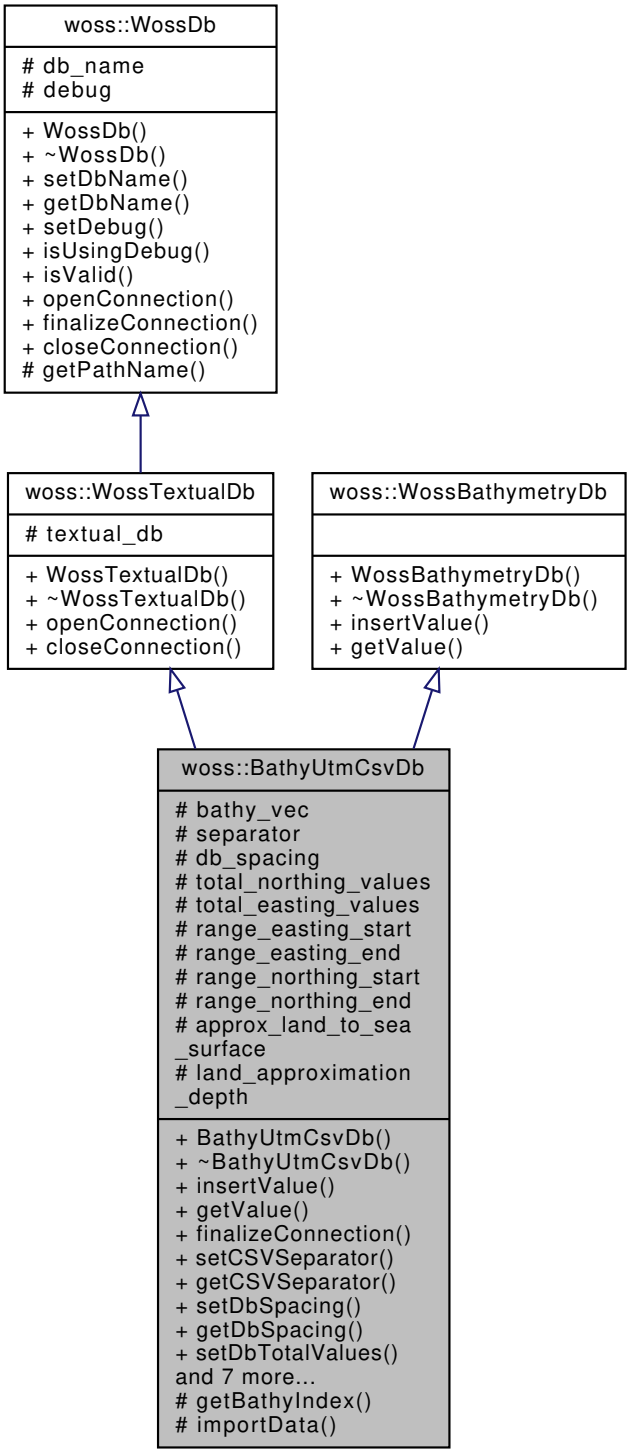
Specialization of WossTextualDb for UTM CSV database.

`#include <bathymetry-utm-csv-db.h>`

Inheritance diagram for woss::BathyUtmCsvDb:



Collaboration diagram for woss::BathyUtmCsvDb:



Public Member Functions

- BathyUtmCsvDb (const ::std::string &name)
- virtual bool insertValue (const Coord &coordinates, const Bathymetry &bathymetry_value)
- virtual double getValue (const Coord &coords) const
- virtual bool finalizeConnection ()
- void setCSVSeparator (const char new_separator)

- const char getCSVSeparator () const
- void setDbSpacing (double spacing)
- double getDbSpacing () const
- void setDbTotalValues (int nnorth, int neast)
- std::pair< int, int > getDbTotalValues () const
- void setDbRangeEasting (double start, double end)
- std::pair< double, double > getDbRangeEasting () const
- void setDbRangeNorthing (double start, double end)
- std::pair< double, double > getDbRangeNorthing () const
- void setLandApproximationFlag (bool flag)
- bool getLandApproximationFlag ()

Protected Member Functions

- int getBathyIndex (const Coord &coords) const
- virtual bool importData ()

Protected Attributes

- std::vector< double > bathy_vec
- char separator
- double db_spacing
- int total_northing_values
- int total_easting_values
- double range_easting_start
- double range_easting_end
- double range_northing_start
- double range_northing_end
- bool approx_land_to_sea_surface

Static Protected Attributes

- static const double land_approximation_depth = 0.000000001

13.9.1 Detailed Description

Specialization of WossTextualDb for UTM CSV database.

Specialization of WossTextualDb for UTM CSV database. It creates a vector used to get requested bathymetry values.

13.9.2 Constructor & Destructor Documentation

13.9.2.1 BathyUtmCsvDb() `BathyUtmCsvDb::BathyUtmCsvDb ( const ::std::string & name )`

BathyUtmCsvDb constructor

Parameters

name	pathname of database

13.9.3 Member Function Documentation

13.9.3.1 finalizeConnection() `bool BathyUtmCsvDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions. It create and initializes a NetCDF variable

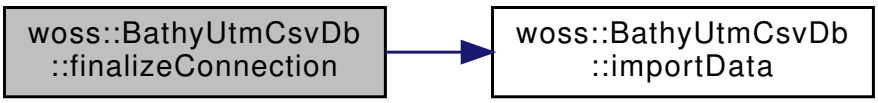
Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References importData().

Here is the call graph for this function:



13.9.3.2 getBathyIndex() `int BathyUtmCsvDb::getBathyIndex ( const Coord & coords ) const [protected]`

Returns the index corresponding the given coordinates. This index will be used to access the vector variable and thus retrieving the bathymetry value

Parameters

coords	const reference to a valid Coord object

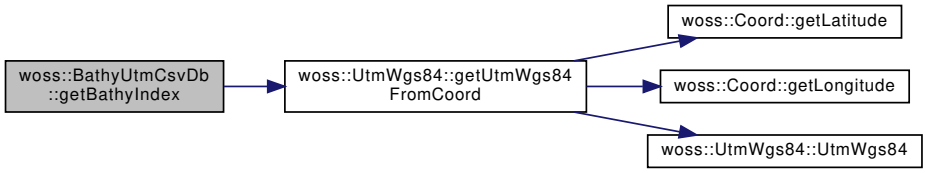
Returns

index value

References db_spacing, woss::WossDb::debug, woss::UtmWgs84::getUtmWgs84FromCoord(), range_easting_end, range_easting_start, range_northing_end, range_northing_start, and total_easting_values.

Referenced by getValue().

Here is the call graph for this function:



13.9.3.3 getCSVSeparator() `const char woss::BathyUtmCsvDb::getCSVSeparator ( ) const [inline]`

Returns the current char separator

Returns

the char separator

References separator.

13.9.3.4 getDbRangeEasting() `std::pair< double, double > woss::BathyUtmCsvDb::getDbRangeEasting ( ) const [inline]`

Returns the current db range easting

Returns

the db easting range start and end

References range_easting_end, and range_easting_start.

13.9.3.5 getDbRangeNorthing() `std::pair< double, double > woss::BathyUtmCsvDb::getDbRangeNorthing ( ) const [inline]`

Returns the current db range northing

Returns

the db easting range start and end

References range_northing_end, and range_northing_start.

13.9.3.6 getDbSpacing() `double woss::BathyUtmCsvDb::getDbSpacing ( ) const [inline]`

Returns the current db spatial resolution

Returns

the db spatial resolution in meters

References db_spacing.

13.9.3.7 getDbTotalValues() `std::pair< int, int > woss::BathyUtmCsvDb::getDbTotalValues ( ) const [inline]`

Returns the total northing and easting values

Returns

pair of total northing and easting values

References total_easting_values, and total_northing_values.

13.9.3.8 getLandApproximationFlag() `bool woss::BathyUtmCsvDb::getLandApproximationFlag ( ) [inline]`

Returns the current land approximation flag

Returns

the land approximation flag

References approx_land_to_sea_surface.

13.9.3.9 getValue() `double BathyUtmCsvDb::getValue ( const Coord & coords ) const [virtual]`

Returns the positive depth value (bathymetry) of given coordinates, if present in the database. If given coordinates are on land (original retrieved value is positive) HUGE_VAL is returned.

Parameters

coords	const reference to a valid Coord object

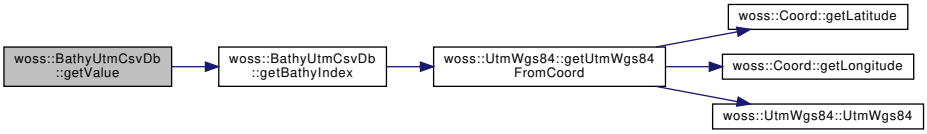
Returns

positive depth value [m] if coordinates are found, *HUGE_VAL* otherwise

Implements woss::WossBathymetryDb.

References approx_land_to_sea_surface, bathy_vec, woss::WossDb::debug, getBathyIndex(), and land_approximation_depth.

Here is the call graph for this function:



13.9.3.10 importData() `bool BathyUtmCsvDb::importData ( ) [protected], [virtual]`

Imports data from .csv file and store depth values in the bathy_vec

Returns

true if operation succeeds, *false* otherwise

References bathy_vec, woss::WossDb::db_name, woss::WossDb::debug, separator, and woss::WossTextualDb::textual_db.

Referenced by finalizeConnection().

13.9.3.11 insertValue() `bool BathyUtmCsvDb::insertValue ( const Coord & coordinates, const Bathymetry & bathymetry_value ) [virtual]`

Inserts the given woss::Bathymetry value in the database for given coordinates

Parameters

coordinates	const reference to a valid Coord object
bathymetry_value	const reference to woss::Bathymetry value to be inserted

Returns

true if method was successful, *false* otherwise

Implements woss::WossBathymetryDb.

13.9.3.12 setCSVSeparator() `void woss::BathyUtmCsvDb::setCSVSeparator ( const char new_separator ) [inline]`

Sets the char separator used in CSV textual db

Parameters

new_separator	separator char

References separator.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.3.13 setDbRangeEasting() `void woss::BathyUtmCsvDb::setDbRangeEasting ( double start, double end ) [inline]`

Sets db range easting start and end

Parameters

start	easting range start
end	easting range end

References range_easting_end, and range_easting_start.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.3.14 setDbRangeNorthing() `void woss::BathyUtmCsvDb::setDbRangeNorthing ( double start, double end ) [inline]`

Sets db northing range start and end

Parameters

start	northing range start
end	northing range end

References range_northing_end, and range_northing_start.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.3.15 setDbSpacing() void woss::BathyUtmCsvDb::setDbSpacing (double *spacing*) [inline]

Sets db resolution, in meters

Parameters

spacing	space resolution in meters

References db_spacing.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.3.16 setDbTotalValues() `void woss::BathyUtmCsvDb::setDbTotalValues ( int nnorth, int neast ) [inline]`

Sets the db total northing and easting values

Parameters

nnorth	db total northing points. should be >= 0
neast	db total easting points. should be >= 0

References total_easting_values, and total_northing_values.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.3.17 setLandApproximationFlag() `void woss::BathyUtmCsvDb::setLandApproximationFlag ( bool flag ) [inline]`

Sets land approximation flag

Parameters

flag	land approximation flag

References approx_land_to_sea_surface.

Referenced by woss::BathyUtmCsvDbCreator::createWossDb().

13.9.4 Member Data Documentation

13.9.4.1 approx_land_to_sea_surface `bool woss::BathyUtmCsvDb::approx_land_to_sea_surface [protected]`

Approximate land db points to sea surface

Referenced by getLandApproximationFlag(), getValue(), and setLandApproximationFlag().

13.9.4.2 bathy_vec `std::vector<double> woss::BathyUtmCsvDb::bathy_vec [protected]`

Vector with bathymetry values

Referenced by getValue(), and importData().

13.9.4.3 db_spacing `double woss::BathyUtmCsvDb::db_spacing [protected]`

spatial spacing of the db, in meters

Referenced by getBathyIndex(), getDbSpacing(), and setDbSpacing().

13.9.4.4 land_approximation_depth `const double BathyUtmCsvDb::land_approximation_depth = 0.↩ 000000001 [static], [protected]`

Depth value for land approximation

Referenced by getValue().

13.9.4.5 range_easting_end `double woss::BathyUtmCsvDb::range_easting_end [protected]`

Separator used in the csv file

Referenced by getBathyIndex(), getDbRangeEasting(), and setDbRangeEasting().

13.9.4.6 range_easting_start `double woss::BathyUtmCsvDb::range_easting_start [protected]`

Separator used in the csv file

Referenced by getBathyIndex(), getDbRangeEasting(), and setDbRangeEasting().

13.9.4.7 range_northing_end `double woss::BathyUtmCsvDb::range_northing_end [protected]`

Separator used in the csv file

Referenced by getBathyIndex(), getDbRangeNorthing(), and setDbRangeNorthing().

13.9.4.8 range_northing_start `double woss::BathyUtmCsvDb::range_northing_start [protected]`

Separator used in the csv file

Referenced by getBathyIndex(), getDbRangeNorthing(), and setDbRangeNorthing().

13.9.4.9 separator `char woss::BathyUtmCsvDb::separator [protected]`

Separator used in the csv file

Referenced by getCSVSeparator(), importData(), and setCSVSeparator().

13.9.4.10 total_easting_values `int woss::BathyUtmCsvDb::total_easting_values [protected]`

Separator used in the csv file

Referenced by getBathyIndex(), getDbTotalValues(), and setDbTotalValues().

13.9.4.11 total_northing_values `int woss::BathyUtmCsvDb::total_northing_values [protected]`

Separator used in the csv file

Referenced by getDbTotalValues(), and setDbTotalValues().

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

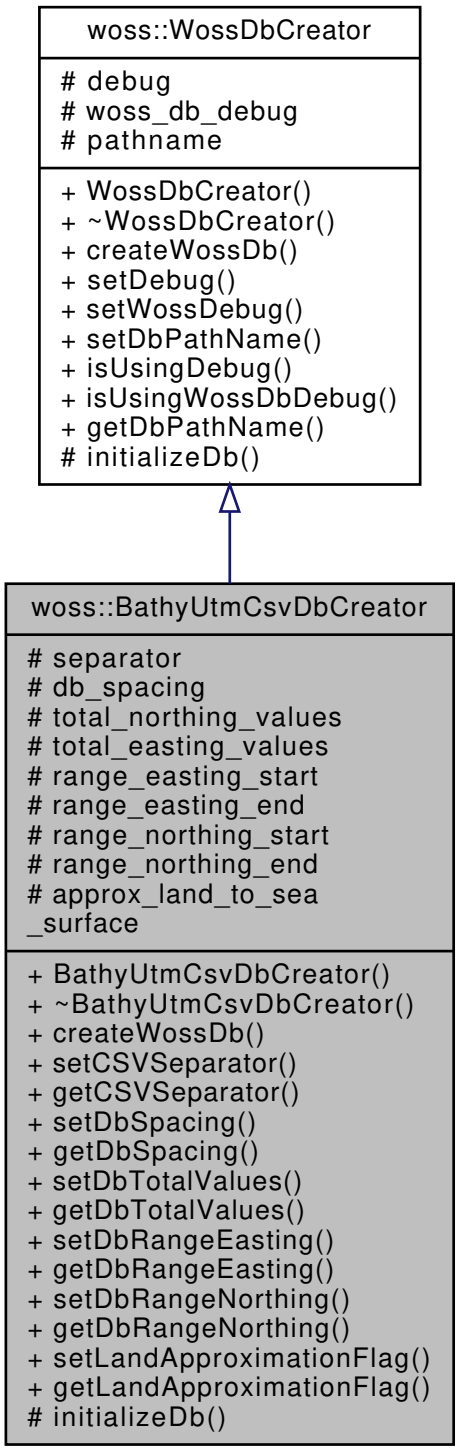
- woss/woss_db/bathymetry-utm-csv-db.h
- woss/woss_db/bathymetry-utm-csv-db.cpp

13.10 woss::BathyUtmCsvDbCreator Class Reference

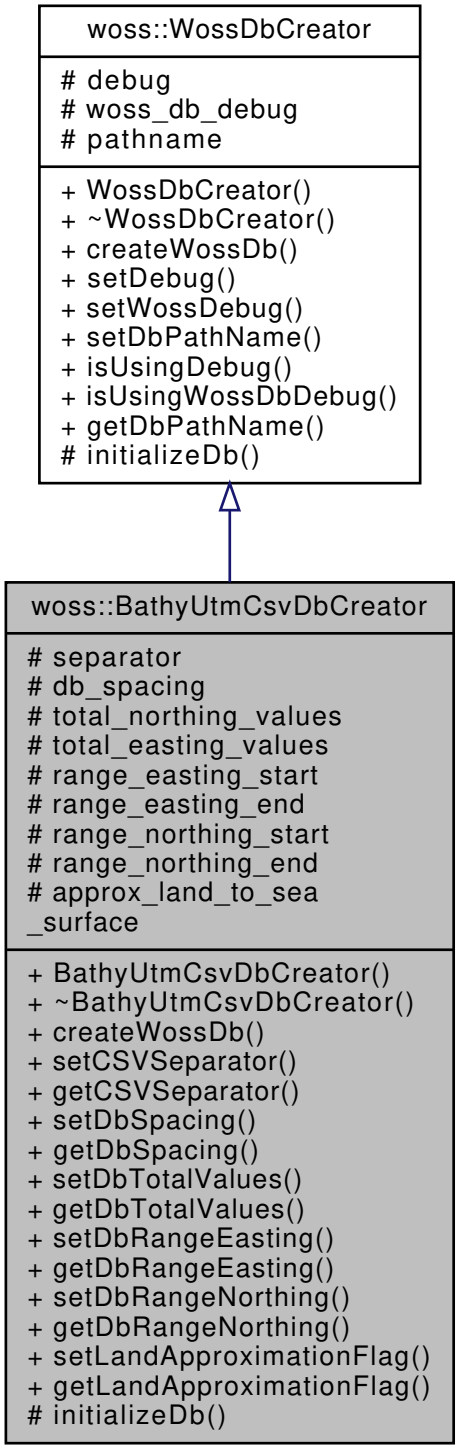
WossDbCreator for the UMT CSV bathymetry database.

`#include <bathymetry-utm-csv-db-creator.h>`

Inheritance diagram for woss::BathyUtmCsvDbCreator:



Collaboration diagram for woss::BathyUtmCsvDbCreator:



Public Member Functions

- BathyUtmCsvDbCreator ()
- virtual WossDb ∗const createWossDb ()
- BathyUtmCsvDbCreator & setCSVSeparator (const char new_separator)
- const char getCSVSeparator () const
- BathyUtmCsvDbCreator & setDbSpacing (double spacing)

- double getDbSpacing () const
- BathyUtmCsvDbCreator & setDbTotalValues (int nnorth, int neast)
- std::pair< int, int > getDbTotalValues () const
- BathyUtmCsvDbCreator & setDbRangeEasting (double start, double end)
- std::pair< double, double > getDbRangeEasting () const
- BathyUtmCsvDbCreator & setDbRangeNorthing (double start, double end)
- std::pair< double, double > getDbRangeNorthing () const
- BathyUtmCsvDbCreator & setLandApproximationFlag (bool flag)
- bool getLandApproximationFlag ()

Protected Member Functions

- virtual bool initializeDb (WossDb *const woss_db)

Protected Attributes

- char separator
- double db_spacing
- int total_northing_values
- int total_easting_values
- double range_easting_start
- double range_easting_end
- double range_northing_start
- double range_northing_end
- bool approx_land_to_sea_surface

13.10.1 Detailed Description

WossDbCreator for the UMT CSV bathymetry database.

Specialization of WossDbCreator for the UMT CSV bathymetry database.

13.10.2 Constructor & Destructor Documentation

13.10.2.1 BathyUtmCsvDbCreator() `BathyUtmCsvDbCreator::BathyUtmCsvDbCreator ( )`

Default BathyUtmCsvDbCreator constructor

13.10.3 Member Function Documentation

13.10.3.1 createWossDb() `WossDb *const BathyUtmCsvDbCreator::createWossDb ( ) [virtual]`

Creates and initialize a BathyUtmCsvDb object

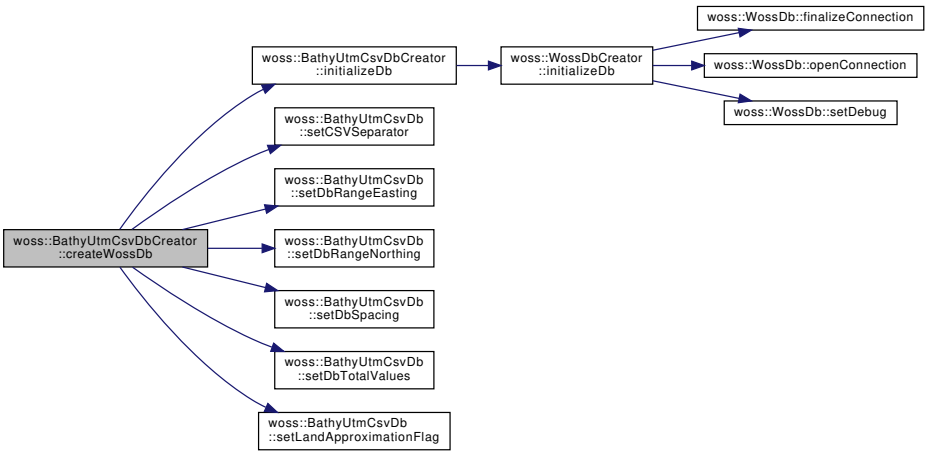
Returns

a pointer to a properly initialized BathyUtmCsvDb object

Implements woss::WossDbCreator.

References approx_land_to_sea_surface, db_spacing, initializeDb(), woss::WossDbCreator::pathname, range_easting_end, range_easting_start, range_northing_end, range_northing_start, separator, woss::BathyUtmCsvDb::setCSVSeparator(), woss::BathyUtmCsvDb::setDbRangeEasting(), woss::BathyUtmCsvDb::setDbRangeNorthing(), woss::BathyUtmCsvDb::setDbSpacin woss::BathyUtmCsvDb::setDbTotalValues(), woss::BathyUtmCsvDb::setLandApproximationFlag(), total_easting_values, and total_northing_values.

Here is the call graph for this function:



13.10.3.2 getCSVSeparator() `const char woss::BathyUtmCsvDbCreator::getCSVSeparator ( ) const [inline]`

Returns the current char separator

Returns

the char separator

References separator.

13.10.3.3 getDbRangeEasting() `std::pair< double, double > woss::BathyUtmCsvDbCreator::getDbRangeEasting ( ) const [inline]`

Returns the current db range easting

Returns

the db easting range start and end

References range_easting_end, and range_easting_start.

13.10.3.4 getDbRangeNorthing() `std::pair< double, double > woss::BathyUtmCsvDbCreator::getDbRangeNorthing ( ) const [inline]`

Returns the current db range northing

Returns

the db easting range start and end

References range_northing_end, and range_northing_start.

13.10.3.5 getDbSpacing() `double woss::BathyUtmCsvDbCreator::getDbSpacing ( ) const [inline]`

Returns the current db spatial resolution

Returns

the db spatial resolution in meters

References db_spacing.

13.10.3.6 getDbTotalValues() `std::pair< int, int > woss::BathyUtmCsvDbCreator::getDbTotalValues ( ) const [inline]`

Returns the total northing and easting values

Returns

pair of total northing and easting values

References total_easting_values, and total_northing_values.

13.10.3.7 getLandApproximationFlag() `bool woss::BathyUtmCsvDbCreator::getLandApproximationFlag ( ) [inline]`

Returns the current land approximation flag

Returns

the land approximation flag

References approx_land_to_sea_surface.

13.10.3.8 initializeDb() `bool BathyUtmCsvDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created BathyUtmCsvDb

Returns

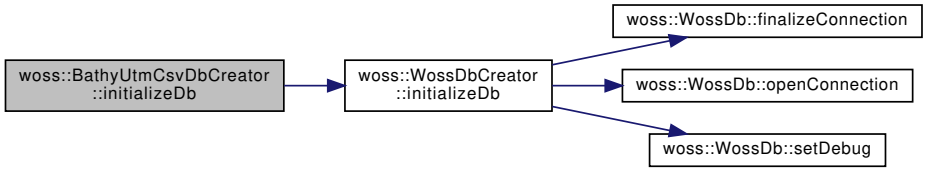
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



13.10.3.9 setCSVSeparator()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::setCSVSeparator (
            const char new_separator )  [inline]
```

Sets the char separator used in CSV textual db

Parameters

new_separator	separator char

Returns

reference to ∗**this**

References separator.

13.10.3.10 setDbRangeEasting()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::setDbRangeEasting (
            double start,
            double end )  [inline]
```

Sets db range easting start and end

Parameters

start	easting range start
end	easting range end

Returns

reference to ∗**this**

References range_easting_end, and range_easting_start.

13.10.3.11 setDbRangeNorthing()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::setDbRangeNorthing (
            double start,
            double end ) [inline]
```

Sets db northing range start and end

Parameters

start	northing range start
end	northing range end

Returns

reference to ∗**this**

References range_northing_end, and range_northing_start.

13.10.3.12 setDbSpacing()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::setDbSpacing (
            double spacing ) [inline]
```

Sets db resolution, in meters

Parameters

spacing	space resolution in meters

Returns

reference to ∗**this**

References db_spacing.

13.10.3.13 setDbTotalValues()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::setDbTotal↩
Values (
            int nnorth,
            int neast )  [inline]
```

Sets the db total northing and easting values

Parameters

nnorth	db total northing points. should be >= 0
neast	db total easting points. should be >= 0

Returns

reference to ∗**this**

References total_easting_values, and total_northing_values.

13.10.3.14 setLandApproximationFlag()

```
BathyUtmCsvDbCreator & woss::BathyUtmCsvDbCreator::set↩
LandApproximationFlag (
            bool flag )  [inline]
```

Sets land approximation flag

Parameters

flag	land approximation flag

Returns

reference to ∗**this**

References approx_land_to_sea_surface.

13.10.4 Member Data Documentation

13.10.4.1 approx_land_to_sea_surface

```
bool woss::BathyUtmCsvDbCreator::approx_land_to_sea_↩
surface  [protected]
```

Approximate land db points to sea surface

Referenced by createWossDb(), getLandApproximationFlag(), and setLandApproximationFlag().

13.10.4.2 db_spacing `double woss::BathyUtmCsvDbCreator::db_spacing [protected]`

spatial spacing of the db, in meters

Referenced by createWossDb(), getDbSpacing(), and setDbSpacing().

13.10.4.3 range_easting_end `double woss::BathyUtmCsvDbCreator::range_easting_end [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbRangeEasting(), and setDbRangeEasting().

13.10.4.4 range_easting_start `double woss::BathyUtmCsvDbCreator::range_easting_start [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbRangeEasting(), and setDbRangeEasting().

13.10.4.5 range_northing_end `double woss::BathyUtmCsvDbCreator::range_northing_end [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbRangeNorthing(), and setDbRangeNorthing().

13.10.4.6 range_northing_start `double woss::BathyUtmCsvDbCreator::range_northing_start [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbRangeNorthing(), and setDbRangeNorthing().

13.10.4.7 separator `char woss::BathyUtmCsvDbCreator::separator [protected]`

Separator used in the csv file

Referenced by createWossDb(), getCSVSeparator(), and setCSVSeparator().

13.10.4.8 total_easting_values `int woss::BathyUtmCsvDbCreator::total_easting_values [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbTotalValues(), and setDbTotalValues().

13.10.4.9 total_northing_values `int woss::BathyUtmCsvDbCreator::total_northing_values [protected]`

Separator used in the csv file

Referenced by createWossDb(), getDbTotalValues(), and setDbTotalValues().

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

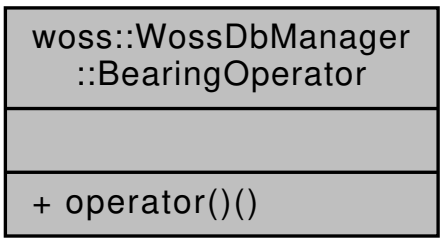
- woss/woss_db/bathymetry-utm-csv-db-creator.h
- woss/woss_db/bathymetry-utm-csv-db-creator.cpp

13.11 woss::WossDbManager::BearingOperator Class Reference

Bearing operator function object.

`#include <woss-db-manager.h>`

Collaboration diagram for woss::WossDbManager::BearingOperator:



Public Member Functions

- double operator() (const Coord &x, const Coord &y) const

13.11.1 Detailed Description

Bearing operator function object.

Function object that returns the bearing between two valid woss::Coord

13.11.2 Member Function Documentation

13.11.2.1 operator()() `double woss::WossDbManager::BearingOperator::operator() ( const Coord & x, const Coord & y ) const [inline]`

Function that compares to woss::Coord instances. If WossDbManager::cust_bathymetry_coord_resolution is valid (>=0) two valid CoordZ are considered equivalent if their great circle distance is less or equal to the space sampling value

Parameters

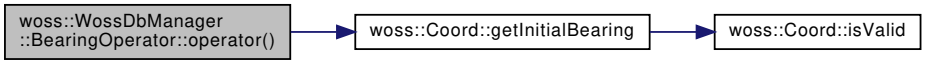
tx	const reference to a valid Coord object
rx	const reference to a valid Coord object

Returns

true if x less than y, *false* otherwise

References woss::Coord::getInitialBearing().

Here is the call graph for this function:



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

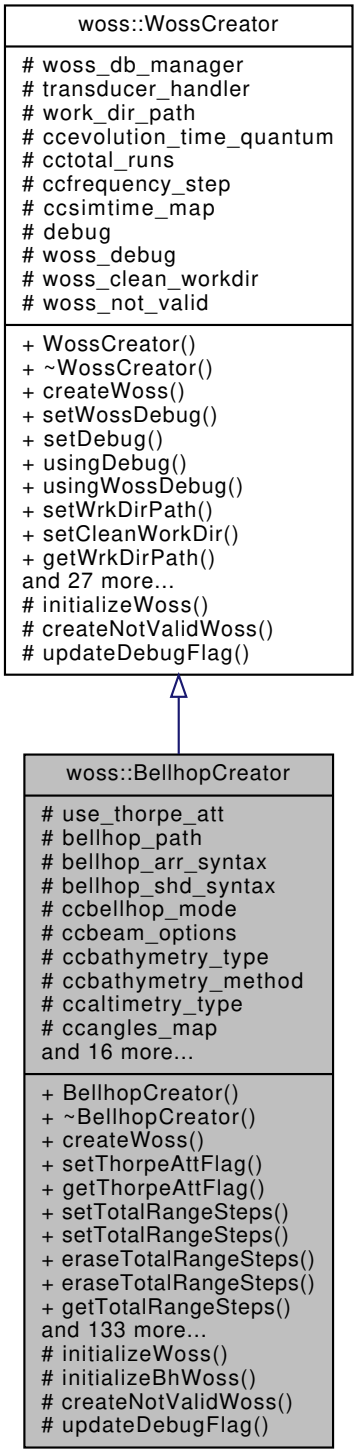
- woss/woss_db/woss-db-manager.h

13.12 woss::BellhopCreator Class Reference

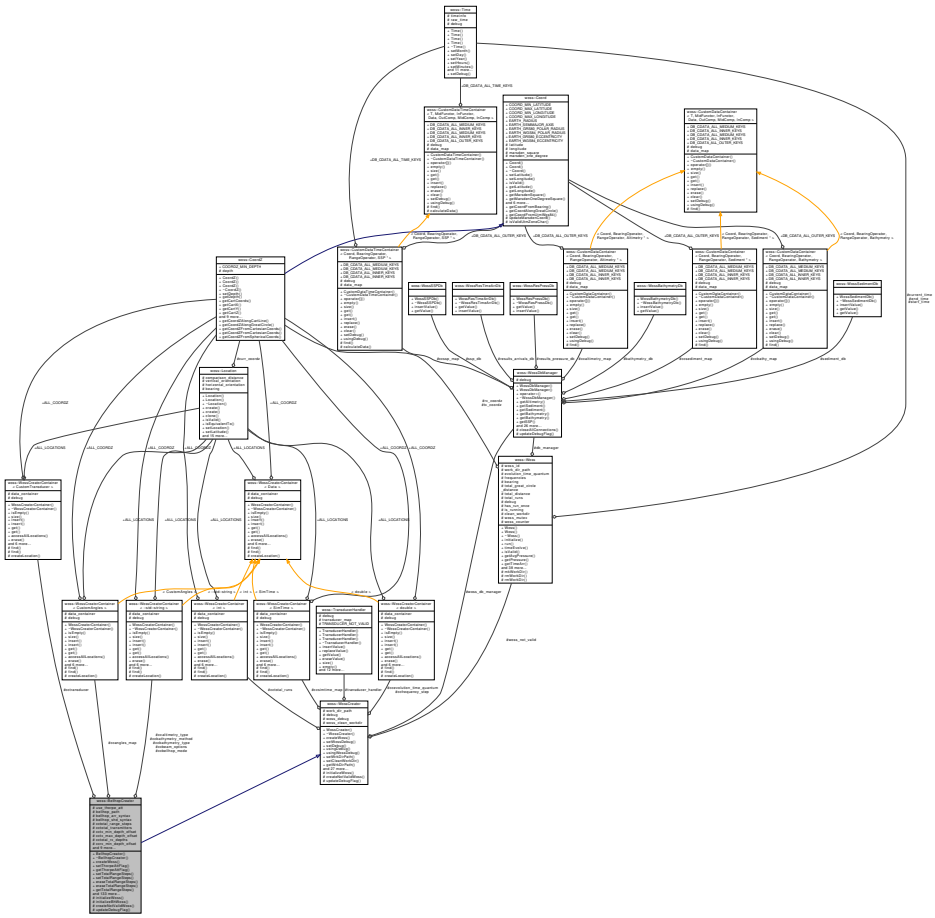
class that provides correctly initialized BellhopWoss objects

`#include <bellhop-creator.h>`

Inheritance diagram for woss::BellhopCreator:



Collaboration diagram for woss::BellhopCreator:



Public Types

- typedef WossCreatorContainer< CustomAngles > CCAngles
- typedef WossCreatorContainer< CustomTransducer > **CCTransducer**
- typedef WossCreatorContainer< ::std::string > **CCString**

Public Member Functions

- BellhopCreator ()
- virtual BellhopWoss *const createWoss (const CoordZ &tx, const CoordZ &rx, double start_frequency, double end_frequency) const
- BellhopCreator & setThorpeAttFlag (bool flag)
- bool getThorpeAttFlag ()
- BellhopCreator & setTotalRangeSteps (int steps, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setTotalRangeSteps (int steps, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseTotalRangeSteps (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseTotalRangeSteps (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- double getTotalRangeSteps (const CoordZ &tx, const CoordZ &rx)
- double getTotalRangeSteps (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setTxMinDepthOffset (double offset, const CoordZ &tx, const CoordZ &rx)

- BellhopCreator & setTxMinDepthOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseTxMinDepthOffset (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseTxMinDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getTxMinDepthOffset (const CoordZ &tx, const CoordZ &rx)
- double getTxMinDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setTxMaxDepthOffset (double offset, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setTxMaxDepthOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseTxMaxDepthOffset (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseTxMaxDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getTxMaxDepthOffset (const CoordZ &tx, const CoordZ &rx)
- double getTxMaxDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setTotalTransmitters (int sources, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setTotalTransmitters (int sources, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseTotalTransmitters (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseTotalTransmitters (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getTotalTransmitters (const CoordZ &tx, const CoordZ &rx)
- int getTotalTransmitters (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setRxMinDepthOffset (double offset, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxMinDepthOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseRxMinDepthOffset (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRxMinDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getRxMinDepthOffset (const CoordZ &tx, const CoordZ &rx)
- double getRxMinDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setRxMaxDepthOffset (double offset, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxMaxDepthOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseRxMaxDepthOffset (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRxMaxDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getRxMaxDepthOffset (const CoordZ &tx, const CoordZ &rx)
- double getRxMaxDepthOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setRxMinRangeOffset (double offset, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxMinRangeOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseRxMinRangeOffset (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRxMinRangeOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getRxMinRangeOffset (const CoordZ &tx, const CoordZ &rx)
- double getRxMinRangeOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setRxMaxRangeOffset (double offset, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxMaxRangeOffset (double offset, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseRxMaxRangeOffset (const CoordZ &tx, const CoordZ &rx)

- BellhopCreator & eraseRxMaxRangeOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getRxMaxRangeOffset (const CoordZ &tx, const CoordZ &rx)
- double getRxMaxRangeOffset (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setRxTotalDepths (int number, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxTotalDepths (int number, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseRxTotalDepths (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRxTotalDepths (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getRxTotalDepths (const CoordZ &tx, const CoordZ &rx)
- int getRxTotalDepths (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setRxTotalRanges (int number, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRxTotalRanges (int number, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseRxTotalRanges (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRxTotalRanges (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getRxTotalRanges (const CoordZ &tx, const CoordZ &rx)
- int getRxTotalRanges (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setRaysNumber (int number, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setRaysNumber (int number, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseRaysNumber (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseRaysNumber (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getRaysNumber (const CoordZ &tx, const CoordZ &rx)
- int getRaysNumber (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setBellhopPath (const ::std::string &path)
- ::std::string getBellhopPath ()
- BellhopCreator & setBellhopArrSyntax (BellhopArrSyntax syntax)
- BellhopArrSyntax getBellhopArrSyntax ()
- BellhopCreator & setBellhopShdSyntax (BellhopShdSyntax syntax)
- BellhopShdSyntax getBellhopShdSyntax ()
- BellhopCreator & setBeamOptions (const ::std::string &options, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBeamOptions (const ::std::string &options, Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & eraseBeamOptions (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseBeamOptions (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- ::std::string getBeamOptions (const CoordZ &tx, const CoordZ &rx)
- ::std::string getBeamOptions (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS
- BellhopCreator & setBhMode (const ::std::string &options, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBhMode (const ::std::string &options, Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & eraseBhMode (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseBhMode (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- ::std::string getBhMode (const CoordZ &tx, const CoordZ &rx)
- ::std::string getBhMode (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & setBathymetryType (const ::std::string &options, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBathymetryType (const ::std::string &options, Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & eraseBathymetryType (const CoordZ &tx, const CoordZ &rx)

- BellhopCreator & eraseBathymetryType (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- ::std::string getBathymetryType (const CoordZ &tx, const CoordZ &rx)
- ::std::string getBathymetryType (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & setBathymetryMethod (const ::std::string &options, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBathymetryMethod (const ::std::string &options, Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & eraseBathymetryMethod (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseBathymetryMethod (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- ::std::string getBathymetryMethod (const CoordZ &tx, const CoordZ &rx)
- ::std::string getBathymetryMethod (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & setAltimetryType (const ::std::string &options, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setAltimetryType (const ::std::string &options, Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- BellhopCreator & eraseAltimetryType (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseAltimetryType (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS)
- ::std::string getAltimetryType (const CoordZ &tx, const CoordZ &rx)
- ::std::string getAltimetryType (Location *const tx=CCString::ALL_LOCATIONS, Location *const rx=CCString::ALL_LOCATIONS
- BellhopCreator & setAngles (const CustomAngles &angles, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setAngles (const CustomAngles &angles, Location *const tx=CCAngles::ALL_LOCATIONS, Location *const rx=CCAngles::ALL_LOCATIONS)
- CustomAngles getAngles (const CoordZ &tx, const CoordZ &rx) const
- CustomAngles getAngles (Location *const tx=CCAngles::ALL_LOCATIONS, Location *const rx=CCAngles::ALL_LOCATIONS) const
- BellhopCreator & eraseAngles (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseAngles (Location *const tx=CCAngles::ALL_LOCATIONS, Location *const rx=CCAngles::ALL_LOCATIONS)
- BellhopCreator & setBoxDepth (double box_depth, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBoxDepth (double box_depth, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseBoxDepth (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseBoxDepth (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getBoxDepth (const CoordZ &tx, const CoordZ &rx)
- double getBoxDepth (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setBoxRange (double box_range, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setBoxRange (double box_range, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseBoxRange (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseBoxRange (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getBoxRange (const CoordZ &tx, const CoordZ &rx)
- double getBoxRange (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & setSspDepthPrecision (double ssp_precision, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setSspDepthPrecision (double ssp_precision, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- BellhopCreator & eraseSspDepthPrecision (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseSspDepthPrecision (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getSspDepthPrecision (const CoordZ &tx, const CoordZ &rx)
- double getSspDepthPrecision (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)

- BellhopCreator & setSspDepthSteps (int ssp_depth_steps, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setSspDepthSteps (int ssp_depth_steps, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & eraseSspDepthSteps (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseSspDepthSteps (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getSspDepthSteps (const CoordZ &tx, const CoordZ &rx)
- int getSspDepthSteps (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- BellhopCreator & setCustomTransducer (const CustomTransducer &type, const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & setCustomTransducer (const CustomTransducer &type, Location *const tx=CCTransducer::ALL_LOCATIONS, Location *const rx=CCTransducer::ALL_LOCATIONS)
- CustomTransducer getCustomTransducer (const CoordZ &tx, const CoordZ &rx) const
- CustomTransducer getCustomTransducer (Location *const tx=CCTransducer::ALL_LOCATIONS, Location *const rx=CCTransducer::ALL_LOCATIONS) const
- BellhopCreator & eraseCustomTransducer (const CoordZ &tx, const CoordZ &rx)
- BellhopCreator & eraseCustomTransducer (Location *const tx=CCTransducer::ALL_LOCATIONS, Location *const rx=CCTransducer::ALL_LOCATIONS)

Protected Member Functions

- virtual bool initializeWoss (Woss *const woss_ptr) const
- bool initializeBhWoss (BellhopWoss *const woss_ptr) const
- virtual const BellhopWoss * createNotValidWoss () const
- virtual void updateDebugFlag ()

Protected Attributes

- bool use_thorpe_att
- ::std::string bellhop_path
- BellhopArrSyntax bellhop_arr_syntax
- BellhopShdSyntax bellhop_shd_syntax
- CCString ccbellhop_mode
- CCString ccbeam_options
- CCString ccbathymetry_type
- CCString ccbathymetry_method
- CCString ccaltimetry_type
- CCAngles ccangles_map
- CCInt cctotal_range_steps
- CCInt cctotal_transmitters
- CCDouble cctx_min_depth_offset
- CCDouble cctx_max_depth_offset
- CCInt cctotal_rx_depths
- CCDouble ccrx_min_depth_offset
- CCDouble ccrx_max_depth_offset
- CCInt cctotal_rx_ranges
- CCDouble ccrx_min_range_offset
- CCDouble ccrx_max_range_offset
- CCInt cctotal_rays
- CCDouble ccssp_depth_precision
- CCInt ccnormalized_ssp_depth_steps
- CCTransducer cctransducer
- CCDouble ccbox_depth
- CCDouble ccbox_range

Additional Inherited Members

13.12.1 Detailed Description

class that provides correctly initialized BellhopWoss objects

BellhopCreator implements WossCreator and provides interface for creation and initializiation of BellhopWoss object

13.12.2 Member Typedef Documentation

13.12.2.1 CCAngles `typedef WossCreatorContainer< CustomAngles > woss::BellhopCreator::CCAngles`

CustomAngles container. A map that links a valid CoordZ (transmitter) with its CustomAngles

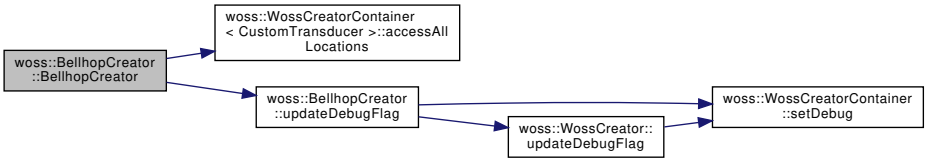
13.12.3 Constructor & Destructor Documentation

13.12.3.1 BellhopCreator() `BellhopCreator::BellhopCreator ( )`

BellhopCreator default constructor

References woss::WossCreatorContainer< CustomTransducer >::accessAllLocations(), woss::BELLHOP_CREATOR_ARR_FILE_IN
woss::BELLHOP_CREATOR_SHD_FILE_INVALID, cctransducer, and updateDebugFlag().

Here is the call graph for this function:



13.12.4 Member Function Documentation

13.12.4.1 createNotValidWoss() `const BellhopWoss * BellhopCreator::createNotValidWoss ( ) const [protected], [virtual]`

Implements woss::WossCreator.

13.12.4.2 createWoss()

```
BellhopWoss *const BellhopCreator::createWoss (
        const CoordZ & tx,
        const CoordZ & rx,
        double start_frequency,
        double end_frequency ) const  [virtual]
```

Returns a pointer to valid BellhopWoss for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

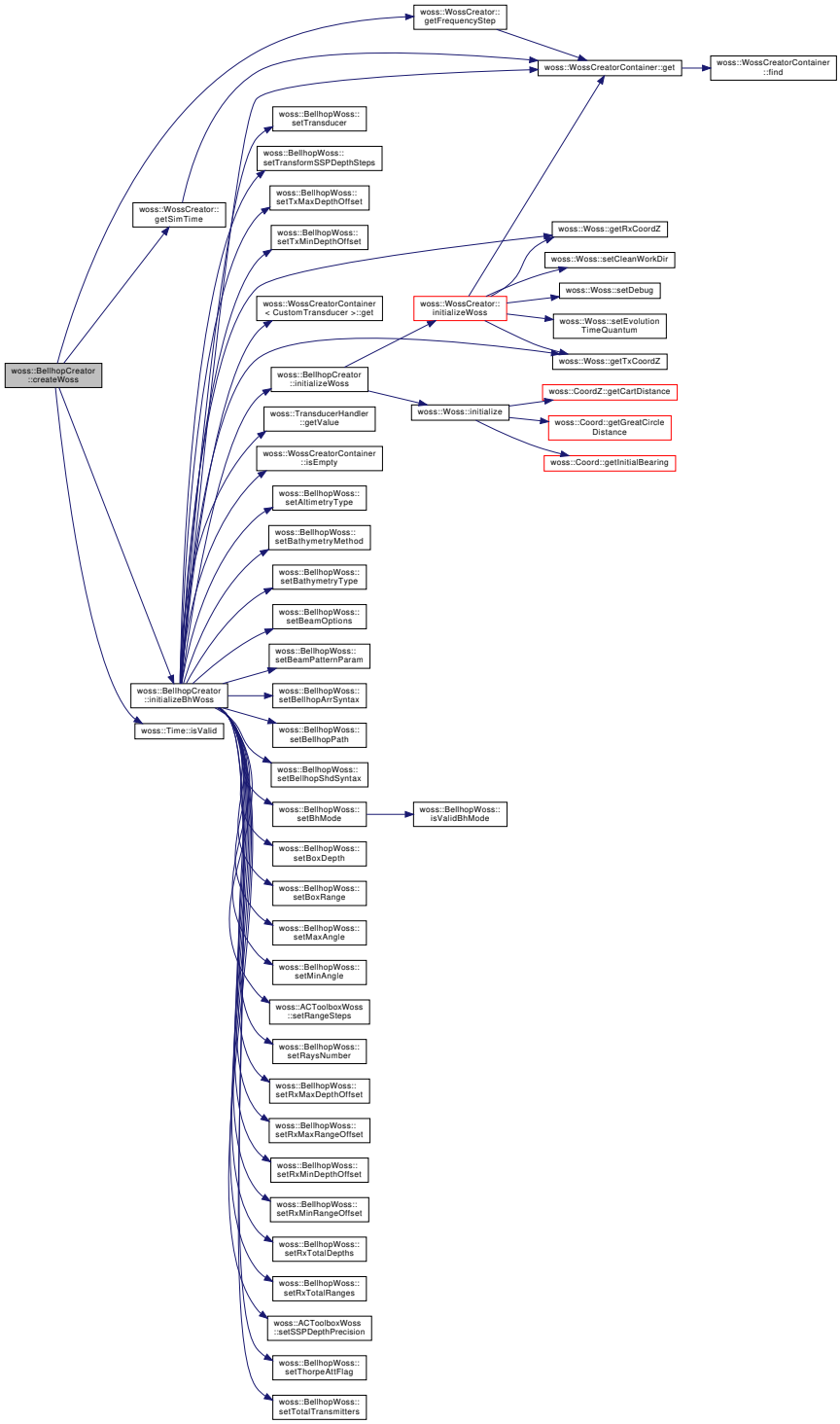
Returns

pointer to properly initialized BellhopWoss object

Implements woss::WossCreator.

References woss::WossCreator::getFrequencyStep(), woss::WossCreator::getSimTime(), initializeBhWoss(), and woss::Time::isValid().

Here is the call graph for this function:



13.12.4.3 eraseAltimetryType() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseAltimetryType (
        const CoordZ & tx,
        const CoordZ & rx )  [inline]
```

Erases the altimetry type (L, C) for given transmitter, receiver woss::CoordZ

Parameters

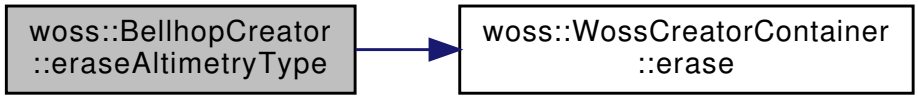
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.4 eraseAltimetryType() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseAltimetryType (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Erases the altimetry type (L, C) for given transmitter, receiver woss::Location

Parameters

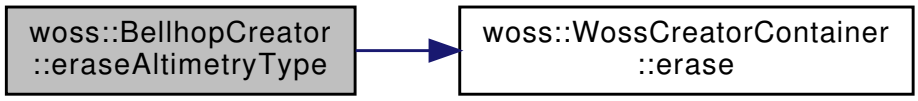
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.5 eraseAngles() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseAngles (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the CustomAngles for given transmitter CoordZ, receiver woss::CoordZ

Parameters

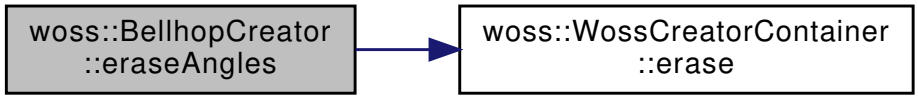
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccangles_map, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.6 eraseAngles() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseAngles (
        Location *const tx = CCAngles::ALL_LOCATIONS,
        Location *const rx = CCAngles::ALL_LOCATIONS )  [inline]
```

Erases the CustomAngles for given transmitter CoordZ, receiver woss::Location

Parameters

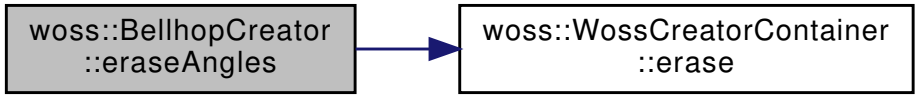
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccangles_map, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.7 eraseBathymetryMethod() [1/2] BellhopCreator & woss::BellhopCreator::eraseBathymetryMethod (

```
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the bathymetry write method (S, D) for given transmitter, receiver woss::CoordZ

Parameters

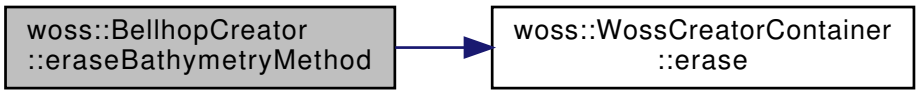
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.8 eraseBathymetryMethod() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBathymetryMethod (
            Location *const tx = CCString::ALL_LOCATIONS,
            Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Erases the bathymetry write method (S, D) for given transmitter, receiver woss::Location

Parameters

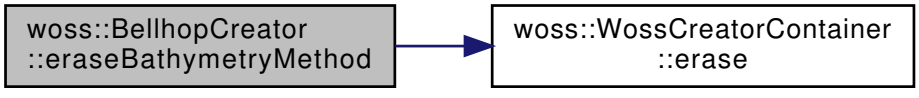
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.9 eraseBathymetryType() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseBathymetryType (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the bathymetry type (L, C) for given transmitter, receiver woss::CoordZ

Parameters

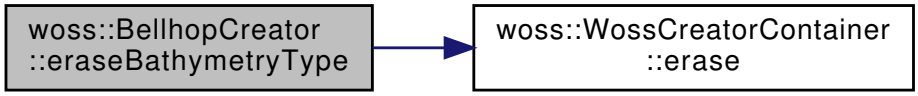
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.10 eraseBathymetryType() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBathymetryType (
            Location *const tx = CCString::ALL_LOCATIONS,
            Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Erases the bathymetry type (L, C) for given transmitter, receiver woss::Location

Parameters

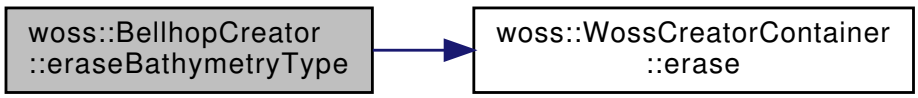
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.11 eraseBeamOptions() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseBeamOptions (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the beam option string for given transmitter, receiver woss::CoordZ

Parameters

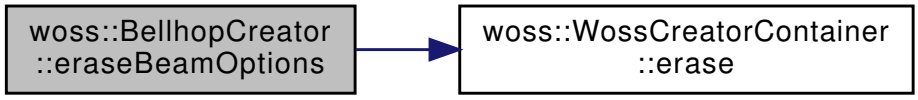
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbeam_options, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.12 eraseBeamOptions() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBeamOptions (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Erases the beam option string for given transmitter, receiver woss::Location

Parameters

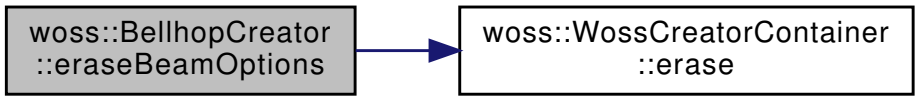
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbeam_options, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.13 eraseBhMode() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseBhMode (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the Bellhop run mode string for given transmitter, receiver woss::CoordZ

Parameters

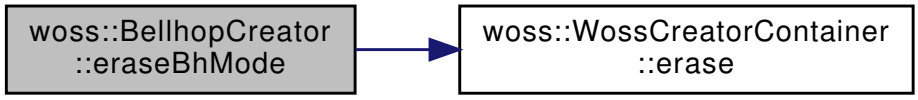
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.14 eraseBhMode() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBhMode (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Erases the Bellhop run mode string for given transmitter, receiver woss::Location

Parameters

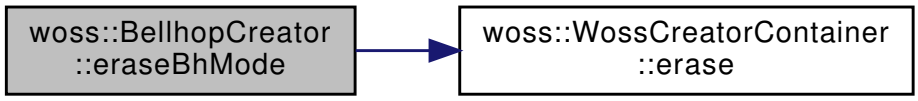
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.15 eraseBoxDepth() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseBoxDepth (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the maximum box depth [m] for given transmitter, receiver woss::CoordZ

Parameters

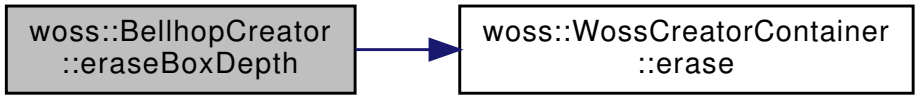
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbox_depth, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.16 eraseBoxDepth() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBoxDepth (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the receiver box depth [m] for given transmitter, receiver woss::Location

Parameters

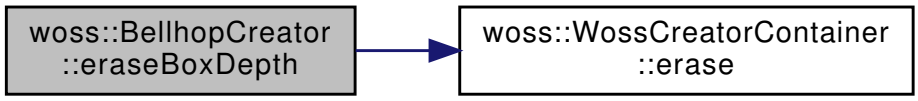
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbox_depth, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.17 eraseBoxRange() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseBoxRange (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the maximum box depth [m] for given transmitter, receiver woss::CoordZ

Parameters

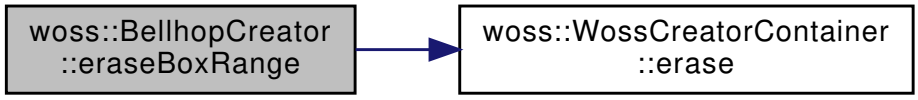
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbox_range, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.18 eraseBoxRange() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseBoxRange (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the receiver box depth [m] for given transmitter, receiver woss::Location

Parameters

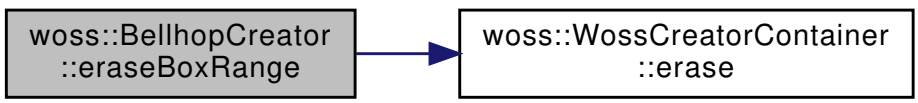
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbox_range, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.19 eraseCustomTransducer() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseCustom↩
Transducer (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the woss::Transducer for given transmitter woss::CoordZ, receiver woss::CoordZ

Parameters

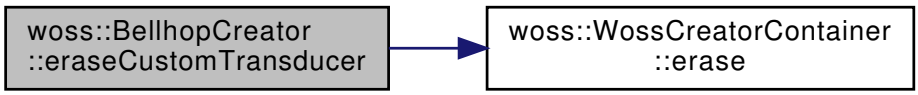
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccangles_map, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.20 eraseCustomTransducer() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseCustomTransducer (
            Location *const tx = CCTransducer::ALL_LOCATIONS,
            Location *const rx = CCTransducer::ALL_LOCATIONS )  [inline]
```

Erases the woss::Transducer for given transmitter woss::CoordZ, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctransducer.

13.12.4.21 eraseRaysNumber() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRaysNumber (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the number of launched rays for given transmitter, receiver woss::CoordZ

Parameters

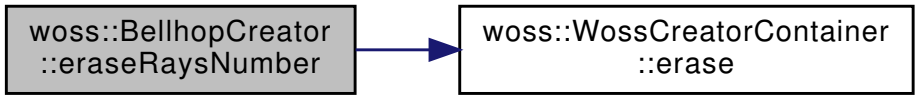
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rays, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.22 eraseRaysNumber() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRaysNumber (
        Location *const tx = CCInt::ALL_LOCATIONS,
        Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the number of launched rays for given transmitter, receiver woss::Location

Parameters

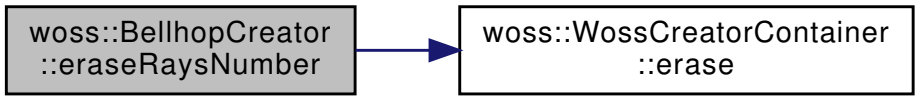
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rays, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.23 eraseRxMaxDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMax↩
DepthOffset (
        const CoordZ & tx,
        const CoordZ & rx )  [inline]
```

Erases the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

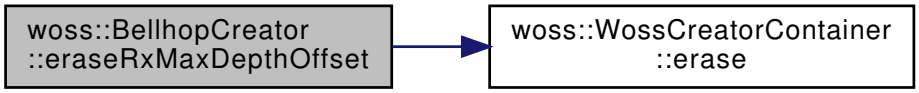
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.24 eraseRxMaxDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMax↩
DepthOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

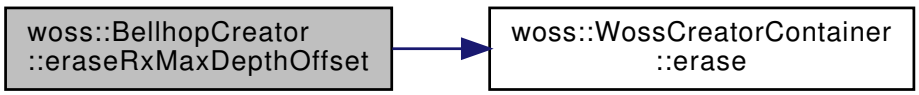
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.25 eraseRxMaxRangeOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMaxRangeOffset (
            const CoordZ & tx,
            const CoordZ & rx ) [inline]
```

Erases the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

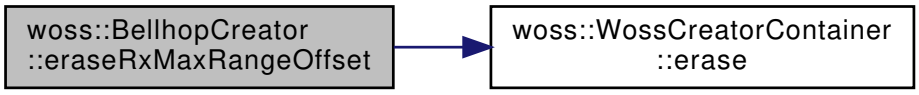
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.26 eraseRxMaxRangeOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMaxRangeOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS ) [inline]
```

Erases the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

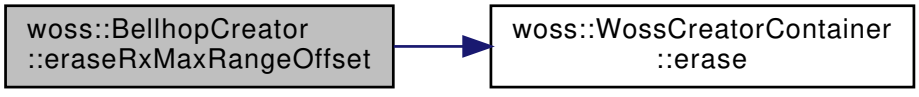
tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.27 eraseRxMinDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMinDepthOffset (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the receiver mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

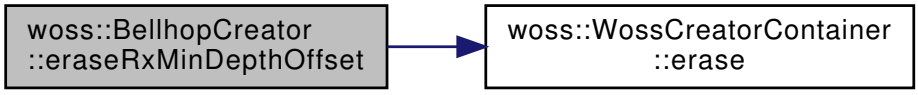
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.28 eraseRxMinDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMinDepthOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the receiver mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

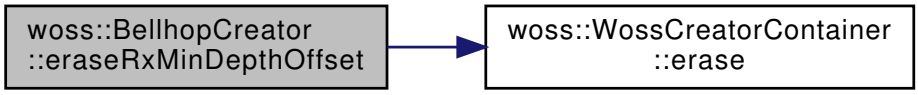
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.29 eraseRxMinRangeOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMinRangeOffset (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the receiver mimimum range offset [m] for given transmitter, receiver woss::CoordZ

Parameters

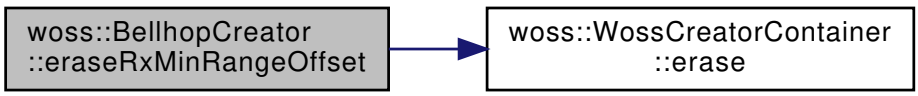
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.30 eraseRxMinRangeOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxMinRangeOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the receiver mimimum range offset [m] for given transmitter, receiver woss::Location

Parameters

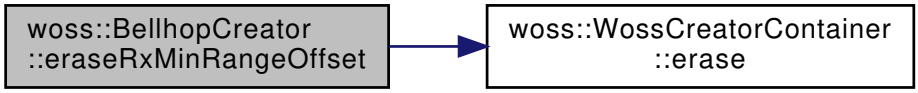
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.31 eraseRxTotalDepths() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxTotalDepths
(
        const CoordZ & tx,
        const CoordZ & rx )  [inline]
```

Erases the number of receiver depths for given transmitter, receiver woss::CoordZ

Parameters

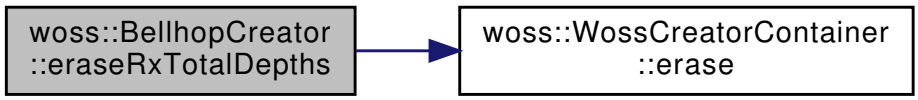
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.32 eraseRxTotalDepths() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxTotalDepths
(
        Location *const tx = CCInt::ALL_LOCATIONS,
        Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the number of receiver depths for given transmitter, receiver woss::Location

Parameters

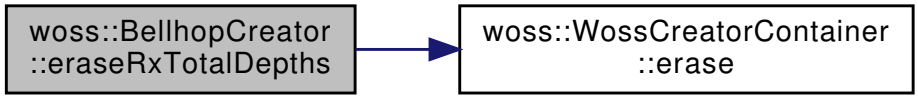
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.33 eraseRxTotalRanges() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseRxTotalRanges
(
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the number of receiver ranges for given transmitter, receiver woss::CoordZ

Parameters

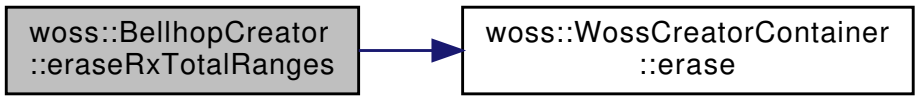
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.34 eraseRxTotalRanges() [2/2] BellhopCreator & woss::BellhopCreator::eraseRxTotalRanges (

```
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the number of receiver ranges for given transmitter, receiver woss::Location

Parameters

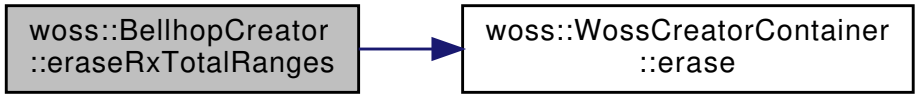
tx	const reference to a valid woss::Location instance
rx	const reference to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.35 eraseSspDepthPrecision() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseSspDepthPrecision (
            const CoordZ & tx,
            const CoordZ & rx ) [inline]
```

Erases the SSP depth precision [m] for given transmitter, receiver woss::CoordZ

Parameters

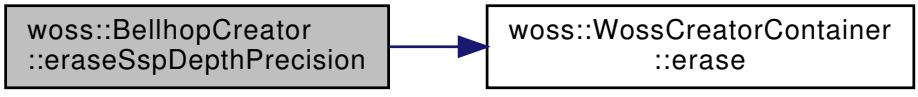
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.36 eraseSspDepthPrecision() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseSspDepthPrecision (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the SSP depth precision [m] for given transmitter, receiver woss::Location

Parameters

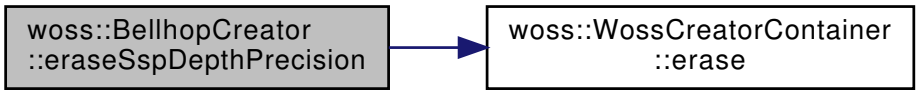
tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.37 eraseSspDepthSteps() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseSspDepthSteps (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases depth steps for given transmitter, receiver woss::CoordZ

Parameters

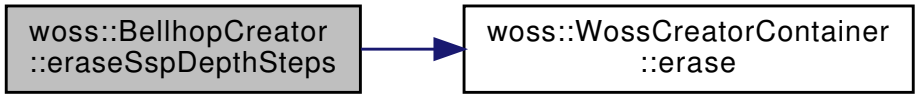
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.38 eraseSspDepthSteps() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseSspDepthSteps (
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases depth steps for given transmitter, receiver woss::Location

Parameters

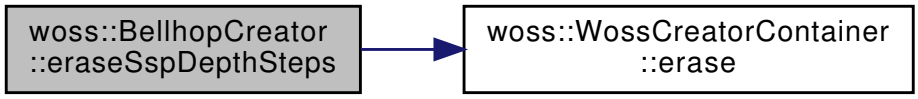
tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.39 eraseTotalRangeSteps() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseTotalRangeSteps (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the total range steps of bellhop simulation for given transmitter, receiver woss::CoordZ

Parameters

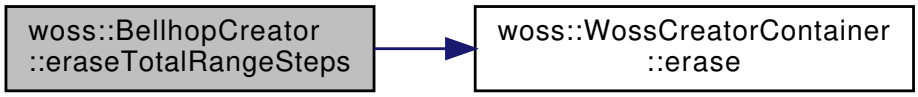
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.40 eraseTotalRangeSteps() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseTotalRangeSteps (
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the total range steps of bellhop simulation for given transmitter, receiver woss::Location

Parameters

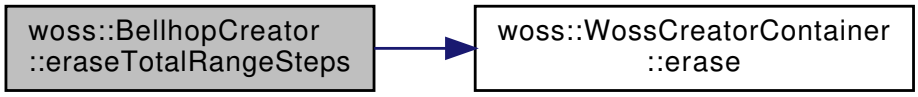
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.41 eraseTotalTransmitters() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseTotalTransmitters (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the number of transmitters for given transmitter, receiver woss::CoordZ

Parameters

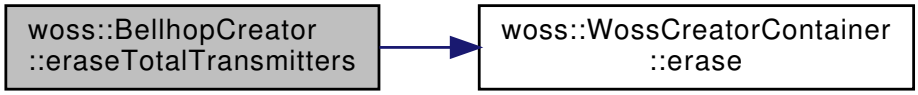
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.42 eraseTotalTransmitters() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseTotal↩
Transmitters (
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the number of transmitters for given transmitter, receiver woss::Location

Parameters

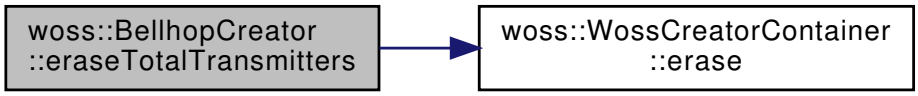
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.43 eraseTxMaxDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseTxMax↩
DepthOffset (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the transmitter maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

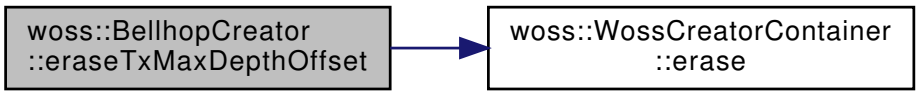
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.44 eraseTxMaxDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseTxMax↩
DepthOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the transmitter maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

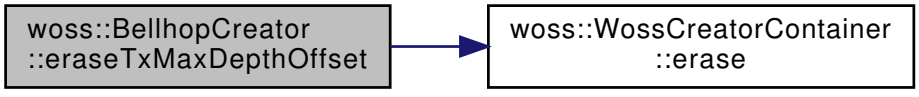
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.45 eraseTxMinDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::eraseTxMinDepthOffset (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

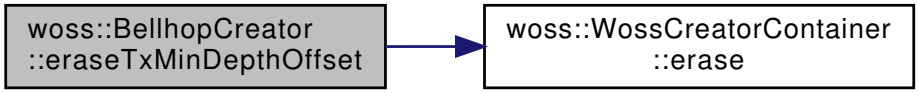
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.46 eraseTxMinDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::eraseTxMinDepthOffset (
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

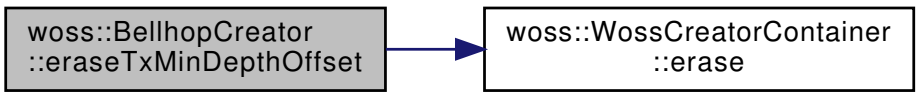
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.12.4.47 getAltimetryType() [1/2]

```
::std::string woss::BellhopCreator::getAltimetryType (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Returns the altimetry type (L, C) for given transmitter, receiver woss::CoordZ

Parameters

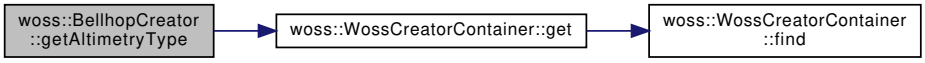
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

altimetry type

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.48 getAltimetryType() [2/2]

```
::std::string woss::BellhopCreator::getAltimetryType (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the altimetry type (L, C) for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

altimetry type

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.49 getAngles() [1/2]

```
CustomAngles woss::BellhopCreator::getAngles (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the CustomAngles for given transmitter CoordZ, receiver woss::CoordZ

Parameters

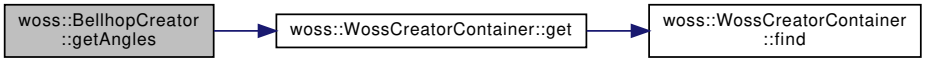
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

a valid CustomAngles

References ccangles_map, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.50 getAngles() [2/2]

```
CustomAngles woss::BellhopCreator::getAngles (
    Location *const tx = CCAngles::ALL_LOCATIONS,
    Location *const rx = CCAngles::ALL_LOCATIONS ) const  [inline]
```

Returns the CustomAngles for given transmitter CoordZ, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

a valid CustomAngles

References ccangles_map, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.51 getBathymetryMethod() [1/2]

```
::std::string woss::BellhopCreator::getBathymetryMethod
(
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the bathymetry write method (S, D) for given transmitter, receiver woss::CoordZ

Parameters

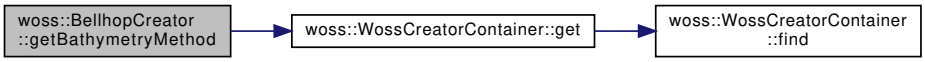
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

bathymetry type

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.52 getBathymetryMethod() [2/2]

```
::std::string woss::BellhopCreator::getBathymetryMethod
(
        Location *const tx = CCString::ALL_LOCATIONS,
        Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the bathymetry write method (S, D) for given transmitter, receiver woss::Location

Parameters

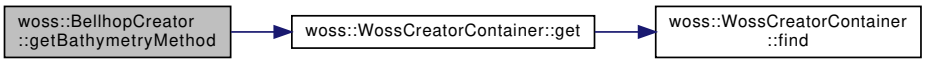
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

bathymetry type

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.53 getBathymetryType() [1/2]

```
::std::string woss::BellhopCreator::getBathymetryType (
        const CoordZ & tx,
        const CoordZ & rx )  [inline]
```

Returns the bathymetry type (L, C) for given transmitter, receiver woss::CoordZ

Parameters

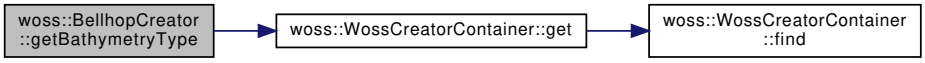
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

bathymetry type

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.54 getBathymetryType() [2/2]

```
::std::string woss::BellhopCreator::getBathymetryType (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the bathymetry type (L, C) for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

bathymetry type

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.55 getBeamOptions() [1/2]

```
::std::string woss::BellhopCreator::getBeamOptions (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the beam option string for given transmitter, receiver woss::CoordZ

Parameters

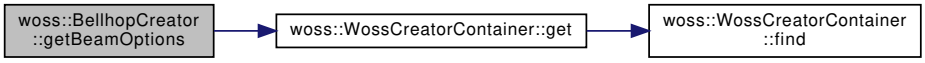
rx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

beam options

References ccbeam_options, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.56 getBeamOptions() [2/2]

```
::std::string woss::BellhopCreator::getBeamOptions (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS ) [inline]
```

Returns the beam option string for given transmitter, receiver woss::Location

Parameters

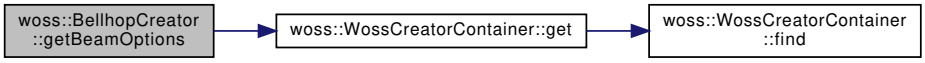
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

beam options

References ccbeam_options, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.57 getBellhopArrSyntax()

```
BellhopArrSyntax woss::BellhopCreator::getBellhopArrSyntax ( ) [inline]
```

Gets the .arr file syntax

Returns

bellhop_arr_syntax

References bellhop_arr_syntax.

13.12.4.58 getBellhopPath() ::std::string woss::BellhopCreator::getBellhopPath ( ) [inline]

Gets the path of bellhop program

Returns

path string

References bellhop_path.

13.12.4.59 getBellhopShdSyntax() BellhopShdSyntax woss::BellhopCreator::getBellhopShdSyntax ( ) [inline]

Gets the .shd file syntax

Returns

bellhop_shd_syntax

References bellhop_shd_syntax.

13.12.4.60 getBhMode() [1/2] ::std::string woss::BellhopCreator::getBhMode (const CoordZ & *tx*, const CoordZ & *rx*) [inline]

Returns the Bellhop run mode string for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

Bellhop run mode

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.61 getBhMode() [2/2]

```
::std::string woss::BellhopCreator::getBhMode (
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Returns the Bellhop run mode string for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

Bellhop run mode

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.62 getBoxDepth() [1/2]

```
double woss::BellhopCreator::getBoxDepth (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the maximum box depth [m] for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccbox_depth, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.63 getBoxDepth() [2/2]

```
double woss::BellhopCreator::getBoxDepth (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the maximum box depth [m] for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccbox_depth, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.64 getBoxRange() [1/2]

```
double woss::BellhopCreator::getBoxRange (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the maximum box depth [m] for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccbox_range, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.65 getBoxRange() [2/2]

```
double woss::BellhopCreator::getBoxRange (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the maximum box depth [m] for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccbox_range, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.66 getCustomTransducer() [1/2]

```
CustomTransducer woss::BellhopCreator::getCustomTransducer (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the woss::Transducer for given transmitter CoordZ, receiver woss::CoordZ

Parameters

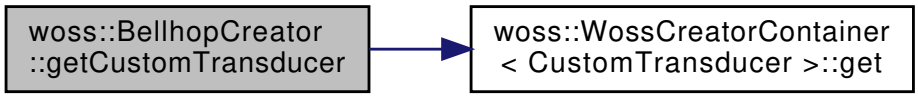
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

a valid woss::Transducer

References cctransducer, and woss::WossCreatorContainer< CustomTransducer >::get().

Here is the call graph for this function:



13.12.4.67 getCustomTransducer() [2/2]

```
CustomTransducer woss::BellhopCreator::getCustomTransducer (
            Location *const tx = CCTransducer::ALL_LOCATIONS,
            Location *const rx = CCTransducer::ALL_LOCATIONS ) const  [inline]
```

Returns the woss::Transducer for given transmitter woss::CoordZ, receiver woss::Location

Parameters

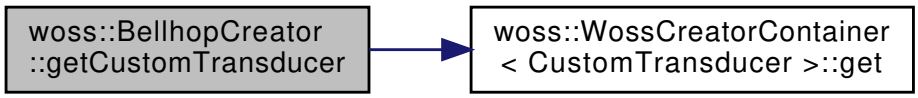
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

a valid woss::Transducer

References cctransducer, and woss::WossCreatorContainer< CustomTransducer >::get().

Here is the call graph for this function:



13.12.4.68 getRaysNumber() [1/2]

```
int woss::BellhopCreator::getRaysNumber (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Returns the number of launched rays for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

number of launched rays

References cctotal_rays, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.69 getRaysNumber() [2/2]

```
int woss::BellhopCreator::getRaysNumber (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Returns the number of launched rays for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

number of launched rays

References cctotal_rays, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.70 getRxMaxDepthOffset() [1/2]

```
double woss::BellhopCreator::getRxMaxDepthOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

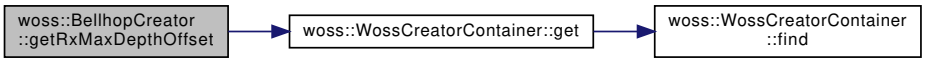
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.71 getRxMaxDepthOffset() [2/2]

```
double woss::BellhopCreator::getRxMaxDepthOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

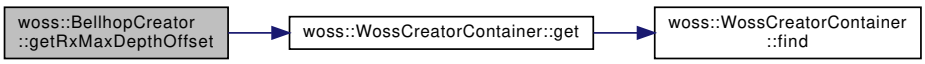
offset	0 <= depth offset <= 0 [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.72 getRxMaxRangeOffset() [1/2]

```
double woss::BellhopCreator::getRxMaxRangeOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

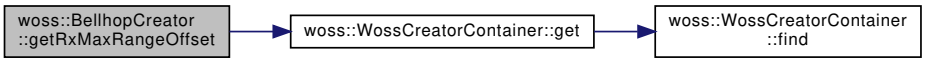
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.73 getRxMaxRangeOffset() [2/2]

```
double woss::BellhopCreator::getRxMaxRangeOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

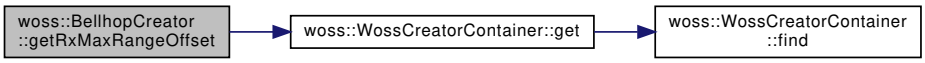
tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.74 getRxMinDepthOffset() [1/2]

```
double woss::BellhopCreator::getRxMinDepthOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the receiver mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

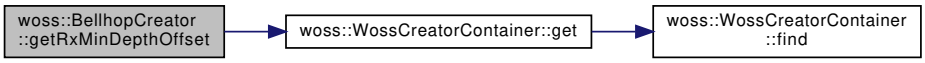
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.75 getRxMinDepthOffset() [2/2]

```
double woss::BellhopCreator::getRxMinDepthOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the receiver mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

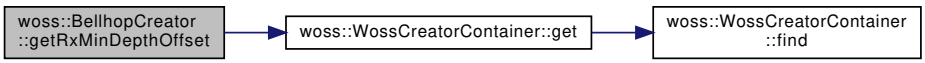
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.76 getRxMinRangeOffset() [1/2]

```
double woss::BellhopCreator::getRxMinRangeOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the receiver mimimum range offset [m] for given transmitter, receiver woss::CoordZ

Parameters

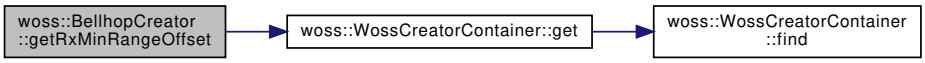
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.77 getRxMinRangeOffset() [2/2]

```
double woss::BellhopCreator::getRxMinRangeOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the receiver mimimum range offset [m] for given transmitter, receiver woss::Location

Parameters

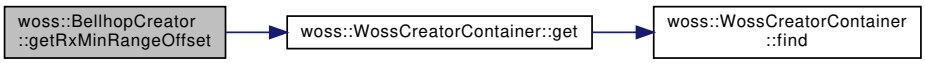
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.78 getRxTotalDepths() [1/2]

```
int woss::BellhopCreator::getRxTotalDepths (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Gets the number of receiver depths for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

number of receiver depths

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.79 getRxTotalDepths() [2/2]

```
int woss::BellhopCreator::getRxTotalDepths (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the number of receiver depths for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

number of receiver depths

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.80 getRxTotalRanges() [1/2]

```
int woss::BellhopCreator::getRxTotalRanges (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the number of receiver ranges for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

number of receiver ranges

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.81 getRxTotalRanges() [2/2]

```
int woss::BellhopCreator::getRxTotalRanges (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Returns the number of receiver ranges for given transmitter, receiver woss::Location

Parameters

tx	const reference to a valid woss::Location instance
rx	const reference to a valid woss::Location instance

Returns

number of receiver ranges

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.82 getSspDepthPrecision() [1/2]

```
double woss::BellhopCreator::getSspDepthPrecision (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the SSP depth precision [m] for given transmitter, receiver woss::CoordZ

Parameters

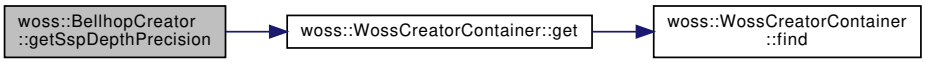
tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.83 getSspDepthPrecision() [2/2]

```
double woss::BellhopCreator::getSspDepthPrecision (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns the SSP depth precision [m] for given transmitter, receiver woss::Location

Parameters

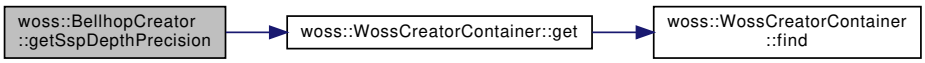
tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.84 getSspDepthSteps() [1/2]

```
int woss::BellhopCreator::getSspDepthSteps (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns depth steps for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
tx	const reference to a valid woss::CoordZ instance

Returns

integer value

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.85 getSspDepthSteps() [2/2]

```
int woss::BellhopCreator::getSspDepthSteps (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Returns depth steps for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
tx	const pointer to a valid woss::Location instance

Returns

integer value

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.86 getThorpeAttFlag()

```
bool woss::BellhopCreator::getThorpeAttFlag ( )  [inline]
```

Gets the Thorpe attenuation flag for all bellhop instances

Returns

boolean flag

References use_thorpe_att.

13.12.4.87 getTotalRangeSteps() [1/2]

```
double woss::BellhopCreator::getTotalRangeSteps (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the total range steps of bellhop simulation for given transmitter, receiver woss::CoordZ

Parameters

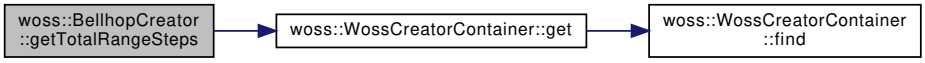
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

int value

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.88 getTotalRangeSteps() [2/2]

```
double woss::BellhopCreator::getTotalRangeSteps (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Returns the total range steps of bellhop simulation for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

int value

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.89 getTotalTransmitters() [1/2]

```
int woss::BellhopCreator::getTotalTransmitters (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns the number of transmitters for given transmitter, receiver woss::CoordZ

Parameters

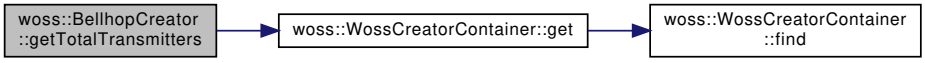
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

total number of transmitter sources

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.90 getTotalTransmitters() [2/2]

```
int woss::BellhopCreator::getTotalTransmitters (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Returns the number of transmitters for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

total number of transmitter sources

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.12.4.91 getTxMaxDepthOffset() [1/2]

```
double woss::BellhopCreator::getTxMaxDepthOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Gets the transmitter maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

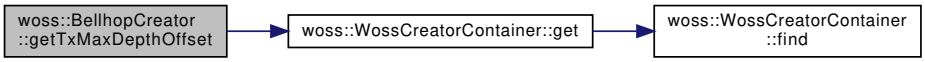
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.92 getTxMaxDepthOffset() [2/2]

```
double woss::BellhopCreator::getTxMaxDepthOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Gets the transmitter maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

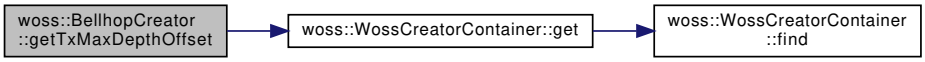
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.93 getTxMinDepthOffset() [1/2]

```
double woss::BellhopCreator::getTxMinDepthOffset (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Returns mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

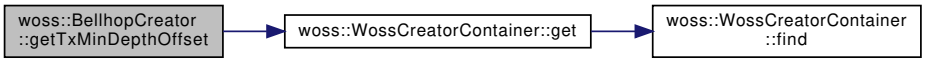
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.94 getTxMinDepthOffset() [2/2]

```
double woss::BellhopCreator::getTxMinDepthOffset (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Returns mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

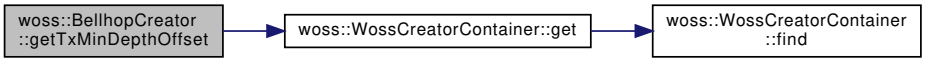
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.12.4.95 initializeBhWoss()

```
bool BellhopCreator::initializeBhWoss (
    BellhopWoss *const woss_ptr ) const  [protected]
```

Initializes given BellhopWoss object

Parameters

woss_ptr	const pointer to an unitialized BellhopWoss

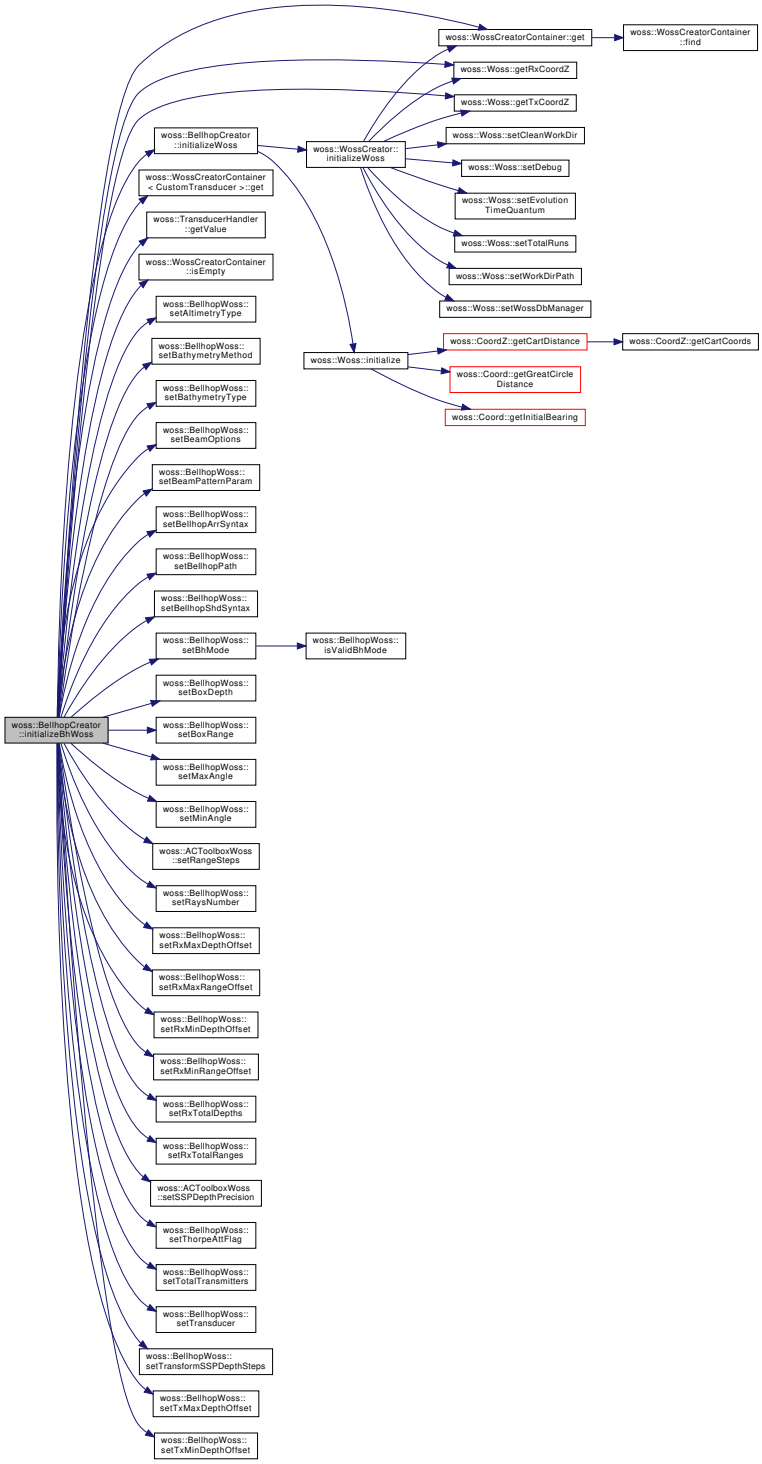
Returns

true if method succeeded, *false* otherwise

References woss::CustomTransducer::add_costant, bellhop_arr_syntax, bellhop_path, bellhop_shd_syntax, ccaltimetry_type, ccangles_map, ccbathymetry_method, ccbathymetry_type, ccbeam_options, ccbellhop_mode, ccbox_depth, ccbox_range, ccnormalized_ssp_depth_steps, ccrx_max_depth_offset, ccrx_max_range_offset, ccrx_min_depth_offset, ccrx_min_range_offset, ccssp_depth_precision, cctotal_range_steps, cctotal_rays, cctotal_rx_depths, cctotal_rx_ranges, cctotal_transmitters, cctransducer, cctx_max_depth_offset, cctx_min_depth_offset, woss::WossCreatorContainer< Data >::get(), woss::WossCreatorContainer< CustomTransducer >::get(), woss::Woss::getRxCoordZ(), woss::Woss::getTxCoordZ(), woss::TransducerHandler::getValue(), woss::CustomTransducer::initial_be woss::CustomTransducer::initial_horiz_rotation, woss::CustomTransducer::initial_vert_rotation, initializeWoss(), woss::WossCreatorContainer< Data >::isEmpty(), woss::CustomAngles::max_angle, woss::CustomAngles::min_angle, woss::CustomTransducer::multiply_costant, woss::BellhopWoss::setAltimetryType(), woss::BellhopWoss::setBathymetryMethod(), woss::BellhopWoss::setBathymetryType(), woss::BellhopWoss::setBeamOptions(), woss::BellhopWoss::setBeamPatternParam(), woss::BellhopWoss::setBellhopArrSyntax(), woss::BellhopWoss::setBellhopPath(), woss::BellhopWoss::setBellhopShdSyntax(), woss::BellhopWoss::setBhMode(), woss::BellhopWoss::setBoxDepth(), woss::BellhopWoss::setBoxRange(), woss::BellhopWoss::setMaxAngle(), woss::BellhopWoss::setMinAngle(), woss::ACToolboxWoss::setRangeSteps(), woss::BellhopWoss::setRaysNumber(), woss::BellhopWoss::setRxMaxDepthOffset(), woss::BellhopWoss::setRxMaxRangeOffset(), woss::BellhopWoss::setRxMinDepthOffset(), woss::BellhopWoss::setRxMinRangeOffset(), woss::BellhopWoss::setRxTotalDepths(), woss::BellhopWoss::setRxTotalRanges(), woss::ACToolboxWoss::setSSPDepthPrecision(), woss::BellhopWoss::setThorpeAttFlag(), woss::BellhopWoss::setTotalTransmitters(), woss::BellhopWoss::setTransducer(), woss::BellhopWoss::setTransformSSPDepthSteps(woss::BellhopWoss::setTxMaxDepthOffset(), woss::BellhopWoss::setTxMinDepthOffset(), woss::CustomTransducer::type, and use_thorpe_att.

Referenced by createWoss().

Here is the call graph for this function:



13.12.4.96 initializeWoss() `bool BellhopCreator::initializeWoss ( Woss *const woss_ptr ) const [protected], [virtual]`

Initializes given BellhopWoss object

Parameters

woss_ptr	const pointer to an unitialized BellhopWoss

Returns

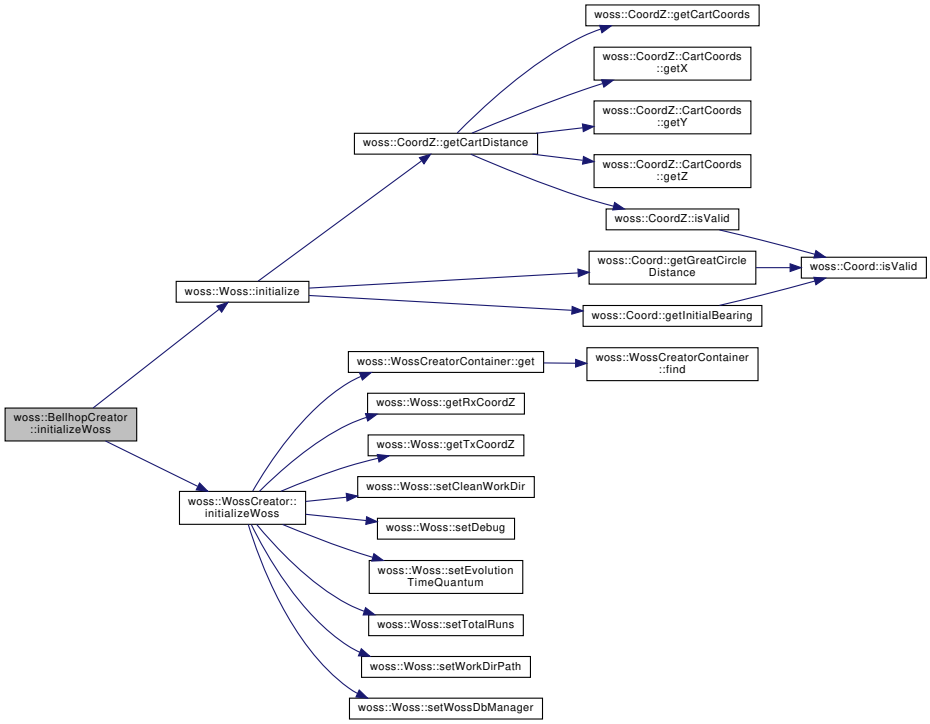
true if method succeeded, *false* otherwise

Implements woss::WossCreator.

References woss::Woss::initialize(), and woss::WossCreator::initializeWoss().

Referenced by initializeBhWoss().

Here is the call graph for this function:



13.12.4.97 setAltimetryType() [1/2]

```
BellhopCreator & woss::BellhopCreator::setAltimetryType (
    const ::std::string & options,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the altimetry type (L, C) for given transmitter, receiver woss::CoordZ. See Bellhop documentation for more info

Parameters

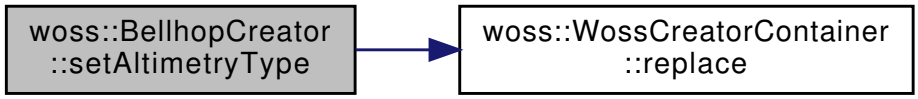
type	altimetry type
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.98 setAltimetryType() [2/2]

```
BellhopCreator & woss::BellhopCreator::setAltimetryType (
    const ::std::string & options,
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the altimetry type (L, C) for given transmitter, receiver woss::Location. See Bellhop documentation for more info

Parameters

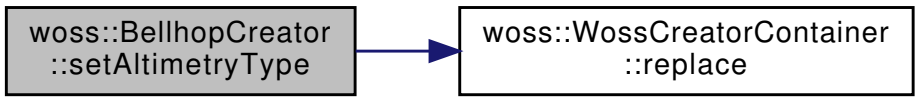
type	altimetry types
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccaltimetry_type, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.99 setAngles() [1/2]

```
BellhopCreator & woss::BellhopCreator::setAngles (
    const CustomAngles & angles,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the CustomAngles for given transmitter CoordZ, receiver woss::CoordZ

Parameters

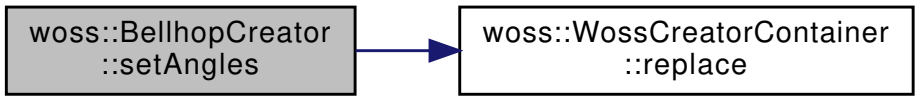
angles	const reference to a valid CustomAngles
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccangles_map, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.100 setAngles() [2/2]

```
BellhopCreator & woss::BellhopCreator::setAngles (
    const CustomAngles & angles,
    Location *const tx = CCAngles::ALL_LOCATIONS,
    Location *const rx = CCAngles::ALL_LOCATIONS )  [inline]
```

Sets the CustomAngles for given transmitter CoordZ, receiver woss::Location

Parameters

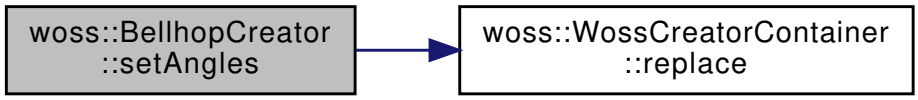
angles	const reference to a valid CustomAngles
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccangles_map, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.101 setBathymetryMethod() [1/2]

```
BellhopCreator & woss::BellhopCreator::setBathymetryMethod (
            const ::std::string & options,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the bathymetry write method (S, D) for given transmitter, receiver woss::CoordZ. See Bellhop documentation for more info

Parameters

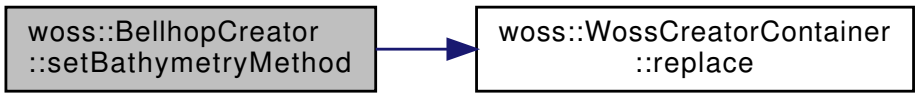
type	bathymetry write method
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.102 setBathymetryMethod() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBathymetryMethod (
            const ::std::string & options,
            Location *const tx = CCString::ALL_LOCATIONS,
            Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the bathymetry write method (S, D) for given transmitter, receiver woss::Location. See Bellhop documentation for more info

Parameters

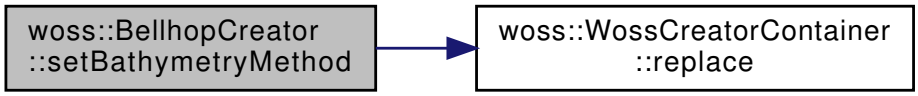
type	bathymetry types
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbathymetry_method, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.103 setBathymetryType() [1/2]

```
BellhopCreator & woss::BellhopCreator::setBathymetryType
(
        const ::std::string & options,
        const CoordZ & tx,
        const CoordZ & rx )  [inline]
```

Sets the bathymetry type (L, C) for given transmitter, receiver woss::CoordZ. See Bellhop documentation for more info

Parameters

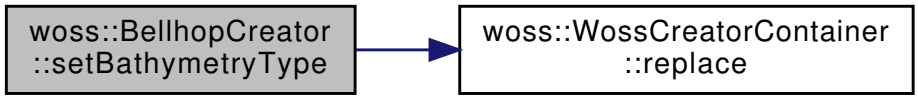
type	bathymetry type
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.104 setBathymetryType() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBathymetryType
(
        const ::std::string & options,
        Location *const tx = CCString::ALL_LOCATIONS,
        Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the bathymetry type (L, C) for given transmitter, receiver woss::Location. See Bellhop documentation for more info

Parameters

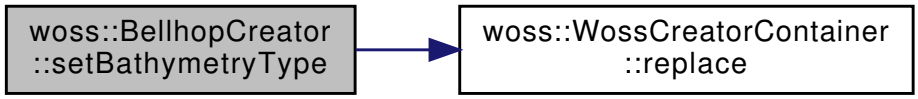
type	bathymetry types
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbathymetry_type, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.105 setBeamOptions() [1/2]

```
BellhopCreator & woss::BellhopCreator::setBeamOptions (
    const ::std::string & options,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the beam option string for given transmitter, receiver woss::CoordZ. See Bellhop documentation for more info

Parameters

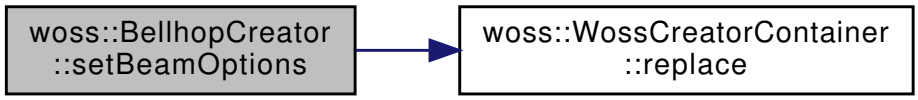
options	beam options (G, C, R , B)
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbeam_options, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.106 setBeamOptions() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBeamOptions (
    const ::std::string & options,
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the beam option string for given transmitter, receiver woss::Location. See Bellhop documentation for more info

Parameters

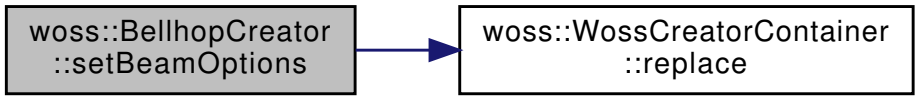
options	beam options (G, C, R , B)
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbeam_options, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.107 setBellhopArrSyntax()

```
BellhopCreator & woss::BellhopCreator::setBellhopArrSyntax (
    BellhopArrSyntax syntax ) [inline]
```

Sets the .arr file syntax to be used during file parsing

Parameters

syntax	.arr file syntax

Returns

reference to ∗**this**

References bellhop_arr_syntax.

13.12.4.108 setBellhopPath()

```
BellhopCreator & woss::BellhopCreator::setBellhopPath (
    const ::std::string & path ) [inline]
```

Sets the path of bellhop program

Parameters

path	filesystem path

Returns

reference to ∗**this**

References bellhop_path.

13.12.4.109 setBellhopShdSyntax() `BellhopCreator & woss::BellhopCreator::setBellhopShdSyntax ( BellhopShdSyntax syntax ) [inline]`

Sets the .shd file syntax to be used during file parsing

Parameters

syntax	.arr file syntax

Returns

reference to ∗**this**

References bellhop_shd_syntax.

13.12.4.110 setBhMode() [1/2] `BellhopCreator & woss::BellhopCreator::setBhMode ( const ::std::string & options, const CoordZ & tx, const CoordZ & rx ) [inline]`

Sets the Bellhop run mode string for given transmitter, receiver woss::CoordZ. See Bellhop documentation for more info

Parameters

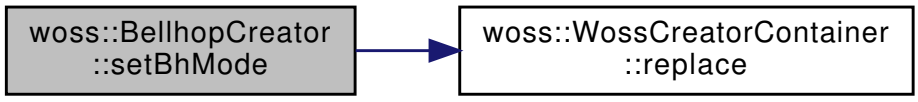
mode	Bellhop run mode (A, a , C, I , S)
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.111 setBhMode() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBhMode (
    const ::std::string & options,
    Location *const tx = CCString::ALL_LOCATIONS,
    Location *const rx = CCString::ALL_LOCATIONS )  [inline]
```

Sets the Bellhop run mode string for given transmitter, receiver woss::Location. See Bellhop documentation for more info

Parameters

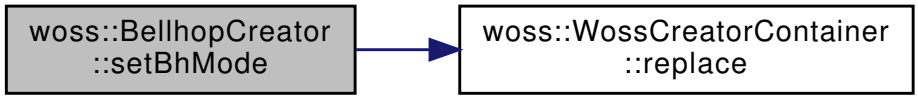
mode	Bellhop run mode (A, a , C, I , S)
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccbellhop_mode, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.112 setBoxDepth() [1/2]

```
BellhopCreator & woss::BellhopCreator::setBoxDepth (
    double box_depth,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the maximum Bellhop ray depth [m] for given transmitter, receiver woss::CoordZ

Parameters

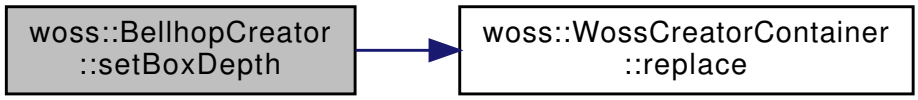
box_depth	[m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbox_depth, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.113 setBoxDepth() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBoxDepth (
    double box_depth,
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the maximum Bellhop ray depth [m] for given transmitter, receiver woss::Location

Parameters

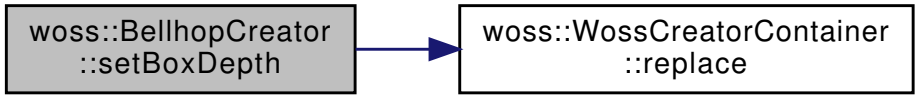
box_depth	[m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccbox_depth, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.114 setBoxRange() [1/2]

```
BellhopCreator & woss::BellhopCreator::setBoxRange (
    double box_range,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the maximum Bellhop ray range [m] for given transmitter, receiver woss::CoordZ

Parameters

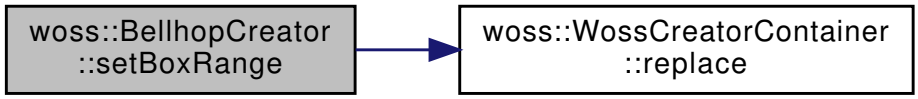
box_depth	[m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccbox_range, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.115 setBoxRange() [2/2]

```
BellhopCreator & woss::BellhopCreator::setBoxRange (
    double box_range,
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the maximum Bellhop ray depth [m] for given transmitter, receiver woss::Location

Parameters

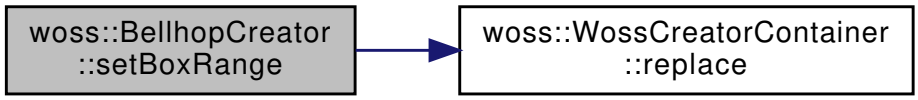
box_depth	[m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

value in meters

References ccbox_range, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.116 setCustomTransducer() [1/2]

```
BellhopCreator & woss::BellhopCreator::setCustom↩
Transducer (
    const CustomTransducer & type,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the woss::Transducer for given transmitter woss::CoordZ, receiver woss::CoordZ

Parameters

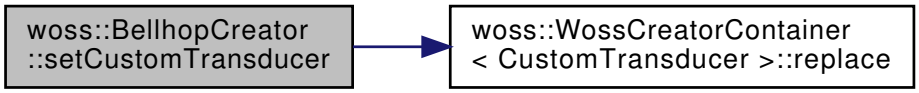
angles	const reference to a valid woss::Transducer
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctransducer, and woss::WossCreatorContainer< CustomTransducer >::replace().

Here is the call graph for this function:



13.12.4.117 setCustomTransducer() [2/2]

```
BellhopCreator & woss::BellhopCreator::setCustomTransducer (
            const CustomTransducer & type,
            Location *const tx = CCTransducer::ALL_LOCATIONS,
            Location *const rx = CCTransducer::ALL_LOCATIONS )  [inline]
```

Sets the woss::Transducer for given transmitter woss::CoordZ, receiver woss::Location

Parameters

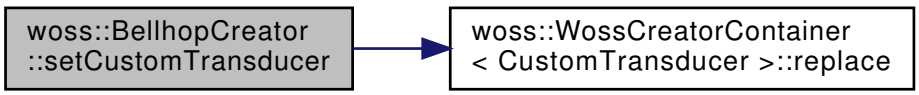
angles	const reference to a valid woss::Transducer
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctransducer, and woss::WossCreatorContainer< CustomTransducer >::replace().

Here is the call graph for this function:



13.12.4.118 setRaysNumber() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRaysNumber (
    int number,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the number of launched rays for given transmitter, receiver woss::CoordZ

Parameters

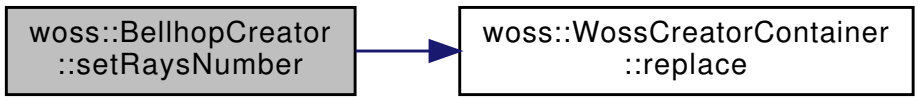
number	number of launched rays (>= 0)
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rays, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.119 setRaysNumber() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRaysNumber (
    int number,
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the number of launched rays for given transmitter, receiver woss::Location

Parameters

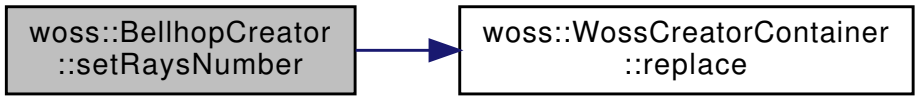
number	number of launched rays (>= 0)
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rays, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.120 setRxMaxDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxMaxDepthOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx ) [inline]
```

Sets the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

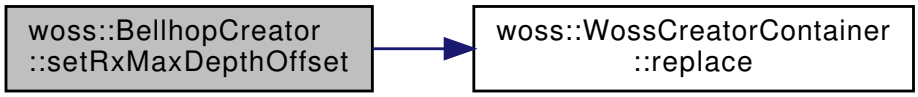
offset	0 <= depth offset <= 0 [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.121 setRxMaxDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxMaxDepthOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS ) [inline]
```

Sets the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

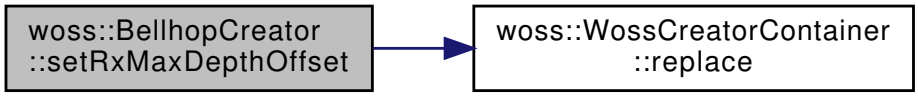
offset	0 <= depth offset <= 0 [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_max_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.122 setRxMaxRangeOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxMaxRangeOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the receiver maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

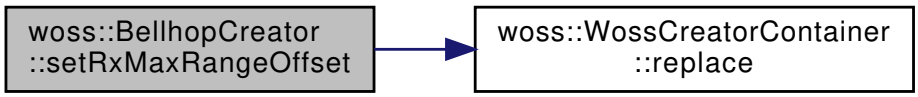
offset	-total_distance <= range offset <= total_distance [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.123 setRxMaxRangeOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxMaxRangeOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the receiver maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

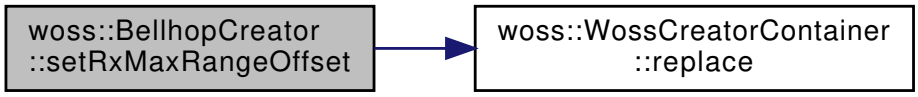
offset	-total_distance <= range offset <= total_distance [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_max_range_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.124 setRxMinDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxMinDepthOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the receiver mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

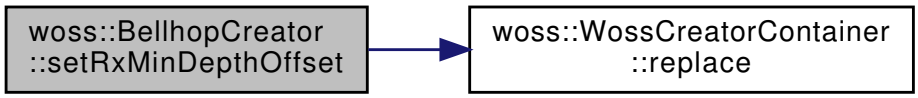
offset	0 <= depth offset <= 0 [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.125 setRxMinDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxMinDepthOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the receiver mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

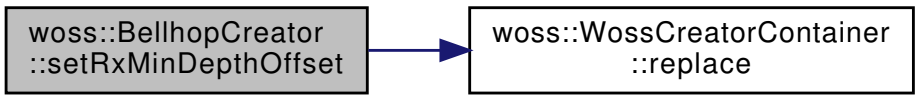
offset	0 <= depth offset <= 0 [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_min_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.126 setRxMinRangeOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxMinRangeOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the receiver mimimum range offset [m] for given transmitter, receiver woss::CoordZ

Parameters

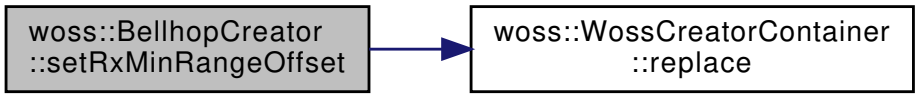
offset	-total_distance <= range offset <= total_distance [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.127 setRxMinRangeOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxMinRangeOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the receiver mimimum range offset [m] for given transmitter, receiver woss::Location

Parameters

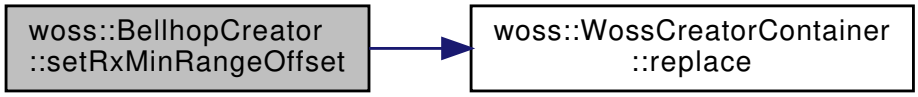
offset	-total_distance <= range offset <= total_distance [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccrx_min_range_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.128 setRxTotalDepths() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxTotalDepths (
            int number,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the number of receiver depths for given transmitter, receiver woss::CoordZ

Parameters

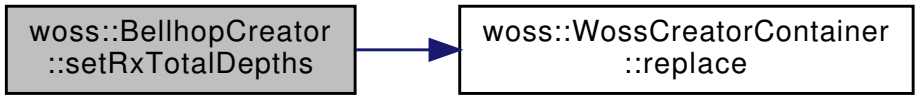
number	number of receiver depths
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.129 setRxTotalDepths() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxTotalDepths (
    int number,
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the number of receiver depths for given transmitter, receiver woss::Location

Parameters

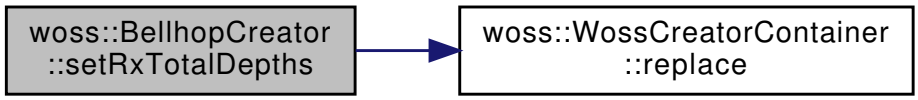
number	number of receiver depths
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rx_depths, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.130 setRxTotalRanges() [1/2]

```
BellhopCreator & woss::BellhopCreator::setRxTotalRanges (
    int number,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the number of receiver ranges for given transmitter, receiver woss::CoordZ

Parameters

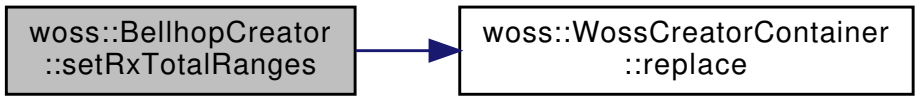
number	number of receiver ranges
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.131 setRxTotalRanges() [2/2]

```
BellhopCreator & woss::BellhopCreator::setRxTotalRanges (
    int number,
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the number of receiver ranges for given transmitter, receiver woss::Location

Parameters

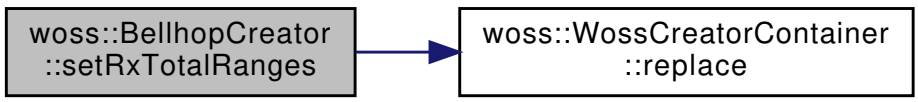
number	number of receiver ranges
tx	const reference to a valid woss::Location instance
rx	const reference to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_rx_ranges, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.132 setSspDepthPrecision() [1/2]

```
BellhopCreator & woss::BellhopCreator::setSspDepthPrecision (
            double ssp_precision,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the SSP depth precision [m] for given transmitter, receiver woss::CoordZ

Parameters

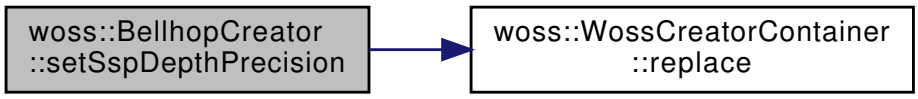
ssp_precision	depth_precision [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.133 setSspDepthPrecision() [2/2]

```
BellhopCreator & woss::BellhopCreator::setSspDepthPrecision (
            double ssp_precision,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the SSP depth precision [m] for given transmitter, receiver woss::Location

Parameters

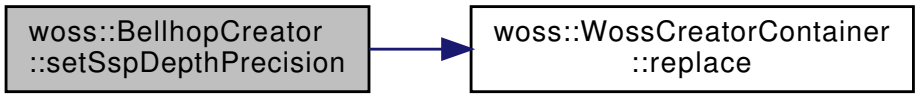
ssp_precision	depth_precision [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccssp_depth_precision, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.134 setSspDepthSteps() [1/2]

```
BellhopCreator & woss::BellhopCreator::setSspDepthSteps (
    int ssp_depth_steps,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the SSP depth steps for given transmitter, receiver woss::CoordZ

Parameters

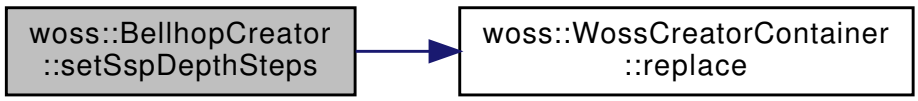
ssp_depth_steps	depth steps
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

value in meters

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.135 setSspDepthSteps() [2/2]

```
BellhopCreator & woss::BellhopCreator::setSspDepthSteps (
    int ssp_depth_steps,
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the SSP depth steps for given transmitter, receiver woss::Location

Parameters

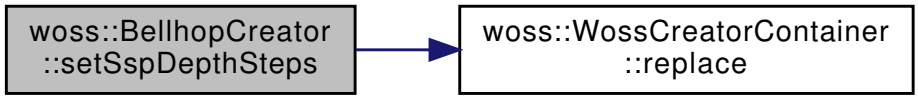
ssp_depth_steps	depth_precision [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccnormalized_ssp_depth_steps, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.136 setThorpeAttFlag()

```
BellhopCreator & woss::BellhopCreator::setThorpeAttFlag (
            bool flag )  [inline]
```

Sets the Thorpe attenuation flag for all bellhop instances

Parameters

flag	boolean flag

Returns

reference to ∗**this**

References use_thorpe_att.

13.12.4.137 setTotalRangeSteps() [1/2]

```
BellhopCreator & woss::BellhopCreator::setTotalRangeSteps (
            int steps,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the total range steps of bellhop simulation for given transmitter, receiver woss::CoordZ

Parameters

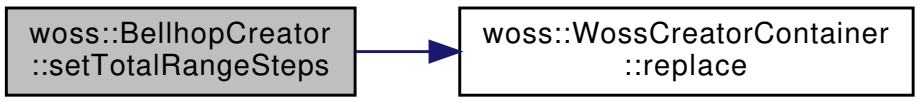
steps	total range steps
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.138 setTotalRangeSteps() [2/2]

```
BellhopCreator & woss::BellhopCreator::setTotalRange↩
Steps (
            int steps,
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the total range steps of bellhop simulation for given transmitter, receiver woss::Location

Parameters

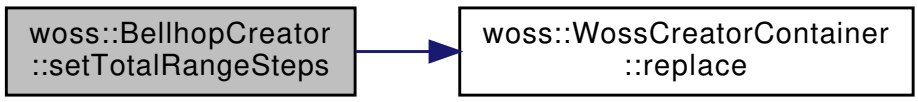
steps	total range steps
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_range_steps, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.139 setTotalTransmitters() [1/2]

```
BellhopCreator & woss::BellhopCreator::setTotalTransmitters
(
            int sources,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the number of transmitters for given transmitter, receiver woss::CoordZ

Parameters

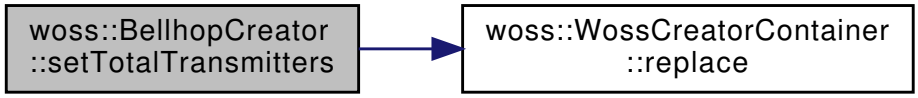
sources	number of transmitters
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.140 setTotalTransmitters() [2/2]

```
BellhopCreator & woss::BellhopCreator::setTotalTransmitters
(
            int sources,
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the number of transmitters for given transmitter, receiver woss::Location

Parameters

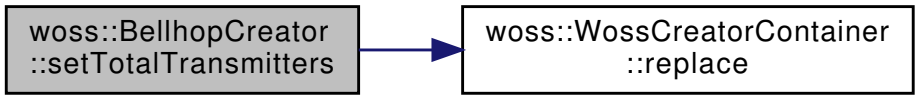
sources	number of transmitters
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_transmitters, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.141 setTxMaxDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setTxMaxDepthOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the transmitter maximum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

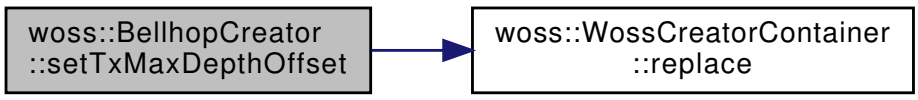
offset	0 <= depth offset <= 0 [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.142 setTxMaxDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setTxMaxDepthOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the transmitter maximum depth offset [m] for given transmitter, receiver woss::Location

Parameters

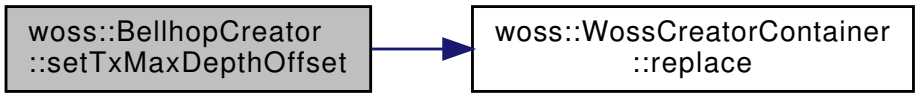
offset	0 <= depth offset <= 0 [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctx_max_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.143 setTxMinDepthOffset() [1/2]

```
BellhopCreator & woss::BellhopCreator::setTxMinDepthOffset (
            double offset,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets mimimum depth offset [m] for given transmitter, receiver woss::CoordZ

Parameters

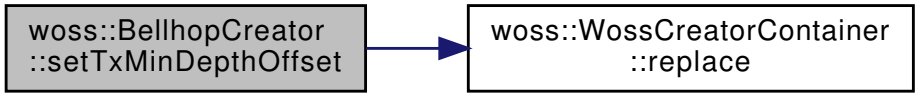
offset	0 <= depth offset <= 0 [m]
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.144 setTxMinDepthOffset() [2/2]

```
BellhopCreator & woss::BellhopCreator::setTxMinDepthOffset (
            double offset,
            Location *const tx = CCDouble::ALL_LOCATIONS,
            Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets mimimum depth offset [m] for given transmitter, receiver woss::Location

Parameters

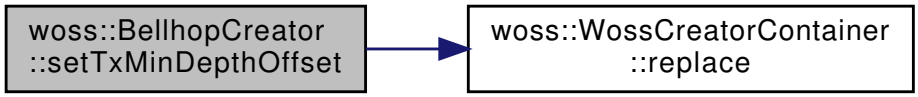
offset	0 <= depth offset <= 0 [m]
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctx_min_depth_offset, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.12.4.145 updateDebugFlag() `void BellhopCreator::updateDebugFlag ( ) [protected], [virtual]`

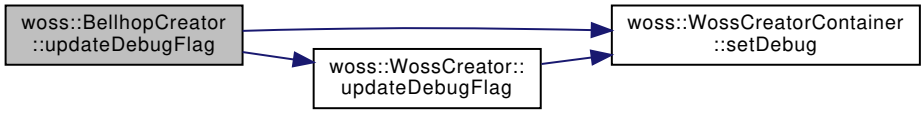
Propagates the debug flag

Reimplemented from woss::WossCreator.

References ccaltimetry_type, ccangles_map, ccbathymetry_method, ccbathymetry_type, ccbeam_options, ccbellhop_mode, ccnormalized_ssp_depth_steps, ccrx_max_depth_offset, ccrx_max_range_offset, ccrx_min_depth_offset, ccrx_min_range_offset, ccssp_depth_precision, cctotal_range_steps, cctotal_rays, cctotal_rx_depths, cctotal_rx_ranges, cctotal_transmitters, cctransducer, cctx_max_depth_offset, cctx_min_depth_offset, woss::WossCreator::debug, woss::WossCreatorContainer< Data >::setDebug(), and woss::WossCreator::updateDebugFlag().

Referenced by BellhopCreator().

Here is the call graph for this function:



13.12.5 Member Data Documentation

13.12.5.1 bellhop_arr_syntax `BellhopArrSyntax woss::BellhopCreator::bellhop_arr_syntax [protected]`

Bellhop .arr file syntax to be used during parsing, factory value = invalid

Referenced by getBellhopArrSyntax(), initializeBhWoss(), and setBellhopArrSyntax().

13.12.5.2 bellhop_path ::std::string woss::BellhopCreator::bellhop_path [protected]

Pathname of Bellhop program

Referenced by getBellhopPath(), initializeBhWoss(), and setBellhopPath().

13.12.5.3 bellhop_shd_syntax BellhopShdSyntax woss::BellhopCreator::bellhop_shd_syntax [protected]

Bellhop .shd file syntax to be used during parsing, factory value = invalid

Referenced by getBellhopShdSyntax(), initializeBhWoss(), and setBellhopShdSyntax().

13.12.5.4 ccaltimetry_type CCString woss::BellhopCreator::ccaltimetry_type [protected]

Altimetry type (L, C)

Referenced by eraseAltimetryType(), getAltimetryType(), initializeBhWoss(), setAltimetryType(), and updateDebugFlag().

13.12.5.5 ccangles_map CCAngles woss::BellhopCreator::ccangles_map [protected]

CustomAngles containter for user-given transmitter CoordZ

Referenced by eraseAngles(), eraseCustomTransducer(), getAngles(), initializeBhWoss(), setAngles(), and updateDebugFlag().

13.12.5.6 ccbathymetry_method CCString woss::BellhopCreator::ccbathymetry_method [protected]

Bathymetry write method (S (slope), D (discrete)

Referenced by eraseBathymetryMethod(), getBathymetryMethod(), initializeBhWoss(), setBathymetryMethod(), and updateDebugFlag().

13.12.5.7 ccbathymetry_type CCString woss::BellhopCreator::ccbathymetry_type [protected]

Bathymetry type (L, C)

Referenced by eraseBathymetryType(), getBathymetryType(), initializeBhWoss(), setBathymetryType(), and updateDebugFlag().

13.12.5.8 ccbeam_options

`CCString woss::BellhopCreator::ccbeam_options [protected]`

Bellhop beam options (G, C, R, B)

Referenced by eraseBeamOptions(), getBeamOptions(), initializeBhWoss(), setBeamOptions(), and updateDebugFlag().

13.12.5.9 ccbellhop_mode

`CCString woss::BellhopCreator::ccbellhop_mode [protected]`

Bellhop run mode (A, a, C, I, S)

Referenced by eraseBhMode(), getBhMode(), initializeBhWoss(), setBhMode(), and updateDebugFlag().

13.12.5.10 ccbox_depth

`CCDouble woss::BellhopCreator::ccbox_depth [protected]`

Maximum Bellhop ray depth [m]

Referenced by eraseBoxDepth(), getBoxDepth(), initializeBhWoss(), and setBoxDepth().

13.12.5.11 ccbox_range

`CCDouble woss::BellhopCreator::ccbox_range [protected]`

Maximum Bellhop ray range [m]

Referenced by eraseBoxRange(), getBoxRange(), initializeBhWoss(), and setBoxRange().

13.12.5.12 ccnormalized_ssp_depth_steps

`CCInt woss::BellhopCreator::ccnormalized_ssp_depth_steps [protected]`

Depth steps of all SSP involved

Referenced by eraseSspDepthSteps(), getSspDepthSteps(), initializeBhWoss(), setSspDepthSteps(), and updateDebugFlag().

13.12.5.13 ccrx_max_depth_offset

`CCDouble woss::BellhopCreator::ccrx_max_depth_offset [protected]`

Receiver maximum depth offset [m]

Referenced by eraseRxMaxDepthOffset(), getRxMaxDepthOffset(), initializeBhWoss(), setRxMaxDepthOffset(), and updateDebugFlag().

13.12.5.14 ccrx_max_range_offset

`CCDouble woss::BellhopCreator::ccrx_max_range_offset [protected]`

Receiver maximum range offset [m]

Referenced by eraseRxMaxRangeOffset(), getRxMaxRangeOffset(), initializeBhWoss(), setRxMaxRangeOffset(), and updateDebugFlag().

13.12.5.15 ccrx_min_depth_offset

`CCDouble woss::BellhopCreator::ccrx_min_depth_offset [protected]`

Receiver minimum depth offset [m]

Referenced by eraseRxMinDepthOffset(), getRxMinDepthOffset(), initializeBhWoss(), setRxMinDepthOffset(), and updateDebugFlag().

13.12.5.16 ccrx_min_range_offset

`CCDouble woss::BellhopCreator::ccrx_min_range_offset [protected]`

Receiver minimum range offset [m]

Referenced by eraseRxMinRangeOffset(), getRxMinRangeOffset(), initializeBhWoss(), setRxMinRangeOffset(), and updateDebugFlag().

13.12.5.17 ccssp_depth_precision

`CCDouble woss::BellhopCreator::ccssp_depth_precision [protected]`

SSP depth precision [m]

Referenced by eraseSspDepthPrecision(), getSspDepthPrecision(), initializeBhWoss(), setSspDepthPrecision(), and updateDebugFlag().

13.12.5.18 cctotal_range_steps

`CCInt woss::BellhopCreator::cctotal_range_steps [protected]`

Number ot range steps

Referenced by eraseTotalRangeSteps(), getTotalRangeSteps(), initializeBhWoss(), setTotalRangeSteps(), and updateDebugFlag().

13.12.5.19 cctotal_rays

`CCInt woss::BellhopCreator::cctotal_rays [protected]`

Number of launched rays

Referenced by eraseRaysNumber(), getRaysNumber(), initializeBhWoss(), setRaysNumber(), and updateDebugFlag().

13.12.5.20 cctotal_rx_depths `CCInt woss::BellhopCreator::cctotal_rx_depths [protected]`

Number of receiver depths

Referenced by eraseRxTotalDepths(), getRxTotalDepths(), initializeBhWoss(), setRxTotalDepths(), and updateDebugFlag().

13.12.5.21 cctotal_rx_ranges `CCInt woss::BellhopCreator::cctotal_rx_ranges [protected]`

Number of receiver ranges.

On some configuration (linux distribution / cpu) bellhop will output an empty file with a value of *total_rx_ranges* = 1.

Referenced by eraseRxTotalRanges(), getRxTotalRanges(), initializeBhWoss(), setRxTotalRanges(), and updateDebugFlag().

13.12.5.22 cctotal_transmitters `CCInt woss::BellhopCreator::cctotal_transmitters [protected]`

Number of transmitter

Referenced by eraseTotalTransmitters(), getTotalTransmitters(), initializeBhWoss(), setTotalTransmitters(), and updateDebugFlag().

13.12.5.23 cctransducer `CCTransducer woss::BellhopCreator::cctransducer [protected]`

Transmitter beam pattern

Referenced by BellhopCreator(), eraseCustomTransducer(), getCustomTransducer(), initializeBhWoss(), setCustomTransducer(), and updateDebugFlag().

13.12.5.24 cctx_max_depth_offset `CCDouble woss::BellhopCreator::cctx_max_depth_offset [protected]`

Transmitter maximum depth offset [m]

Referenced by eraseTxMaxDepthOffset(), getTxMaxDepthOffset(), initializeBhWoss(), setTxMaxDepthOffset(), and updateDebugFlag().

13.12.5.25 cctx_min_depth_offset `CCDouble woss::BellhopCreator::cctx_min_depth_offset [protected]`

Transmitter minimum depth offset [m]

Referenced by eraseTxMinDepthOffset(), getTxMinDepthOffset(), initializeBhWoss(), setTxMinDepthOffset(), and updateDebugFlag().

13.12.5.26 use_thorpe_att `bool woss::BellhopCreator::use_thorpe_att [protected]`

flag that control bellhop's thorpe calculation

Referenced by getThorpeAttFlag(), initializeBhWoss(), and setThorpeAttFlag().

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

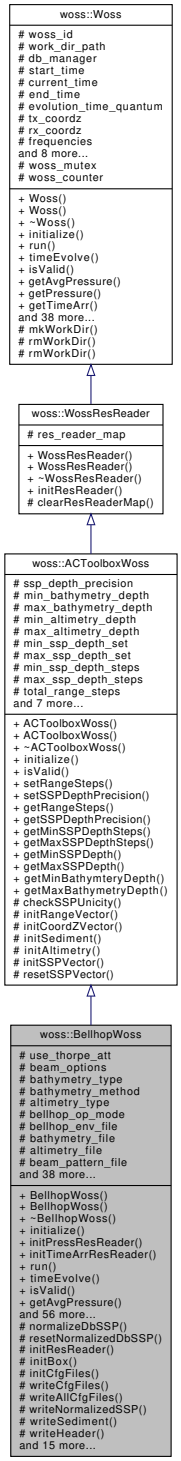
- woss/bellhop-creator.h
- woss/bellhop-creator.cpp

13.13 woss::BellhopWoss Class Reference

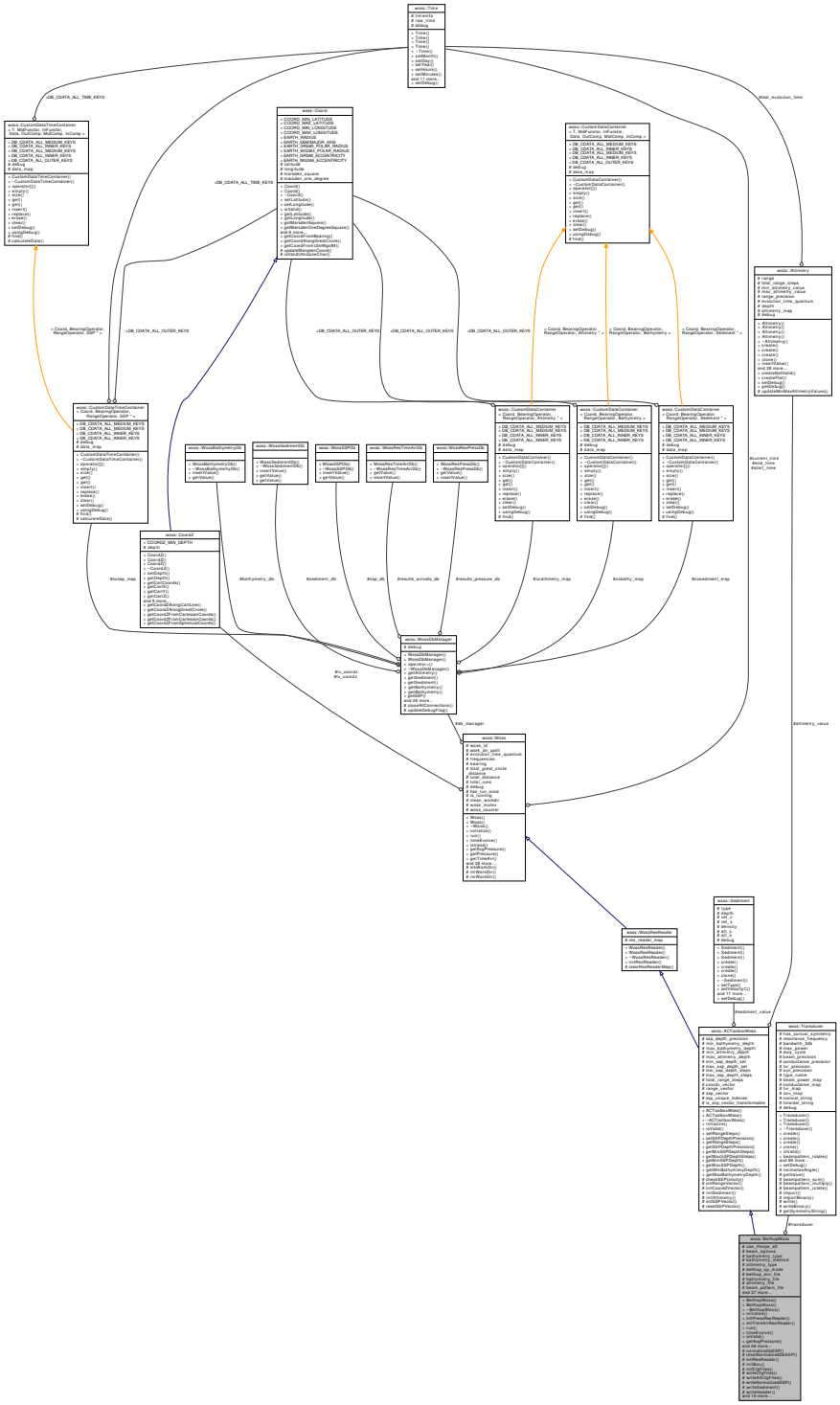
Implempentation of ACToolboxWoss for Bellhop raytracing program.

`#include <bellhop-woss.h>`

Inheritance diagram for woss::BellhopWoss:



Collaboration diagram for woss::BellhopWoss:



Public Member Functions

- BellhopWoss ()
- BellhopWoss (const CoordZ &tx, const CoordZ &rx, const Time &start_t, const Time &end_t, double start_↩ freq, double end_freq, double freq_step)
- virtual bool initialize ()
- virtual bool initPressResReader (double curr_frequency)

- virtual bool initTimeArrResReader (double curr_frequency)
- virtual bool run ()
- virtual bool timeEvolve (const Time &time_value)
- virtual bool isValid () const
- virtual Pressure * getAvgPressure (double frequency, double tx_depth, double start_rx_depth=WOSS_MIN↩ _DEPTH, double start_rx_range=WOSS_MIN_RANGE, double end_rx_depth=WOSS_MAX_DEPTH, double end_rx_range=WOSS_MAX_RANGE) const
- virtual Pressure * getPressure (double frequency, double tx_depth, double rx_depth, double rx_range) const
- virtual TimeArr * getTimeArr (double frequency, double tx_depth, double rx_depth, double rx_range) const
- BellhopWoss & setThorpeAttFlag (bool flag)
- BellhopWoss & setTxMinDepthOffset (double offset)
- BellhopWoss & setTxMaxDepthOffset (double offset)
- BellhopWoss & setTotalTransmitters (int sources)
- BellhopWoss & setRxMinDepthOffset (double offset)
- BellhopWoss & setRxMaxDepthOffset (double offset)
- BellhopWoss & setRxMinRangeOffset (double offset)
- BellhopWoss & setRxMaxRangeOffset (double offset)
- BellhopWoss & setRxTotalDepths (int number)
- BellhopWoss & setRxTotalRanges (int number)
- BellhopWoss & setRaysNumber (int number)
- BellhopWoss & setMinAngle (double angle)
- BellhopWoss & setMaxAngle (double angle)
- BellhopWoss & setBoxDepth (double depth)
- BellhopWoss & setBoxRange (double range)
- BellhopWoss & setBellhopPath (const ::std::string &path)
- BellhopWoss & setBellhopArrSyntax (BellhopArrSyntax syntax)
- BellhopWoss & setBellhopShdSyntax (BellhopShdSyntax syntax)
- BellhopWoss & setBeamOptions (const ::std::string &options)
- BellhopWoss & setBhMode (const ::std::string &mode)
- BellhopWoss & setBathymetryType (const ::std::string &type)
- BellhopWoss & setBathymetryMethod (const ::std::string &type)
- BellhopWoss & setAltimetryType (const ::std::string &type)
- BellhopWoss & setTransducer (const Transducer *const ptr)
- BellhopWoss & setTransformSSPDepthSteps (int depth_steps)
- BellhopWoss & setBeamPatternParam (double init_bearing, double vert_rot=0.0, double horiz_rot=0.0, double mult=1.0, double add=0.0)
- bool getThorpeAttFlag () const
- double getTxMinDepthOffset () const
- double getTxMaxDepthOffset () const
- int getTotalTransmitters () const
- double getRxMinDepthOffset () const
- double getRxMaxDepthOffset () const
- double getRxMinRangeOffset () const
- double getRxMaxRangeOffset () const
- int getRxTotalDepths () const
- int getRxTotalRanges () const
- int getRaysNumber () const
- double getMinAngle () const
- double getMaxAngle () const
- double getBoxDepth () const
- double getBoxRange () const
- int getTransformSSPDepthSteps () const
- ::std::string getBellhopPath () const
- BellhopArrSyntax getBellhopArrSyntax () const
- BellhopShdSyntax getBellhopShdSyntax () const

- ::std::string getBeamOptions () const
- ::std::string getBathymetryType () const
- ::std::string getBathymetryMethod () const
- ::std::string getAltimetryType () const
- const Transducer *const getTransducer () const
- bool isValidBhMode (const ::std::string &) const
- bool usingPressMode () const
- bool usingTimeArrMode () const
- bool usingSSPFile () const

Protected Member Functions

- virtual void normalizeDbSSP ()
- virtual void resetNormalizedDbSSP ()
- bool initResReader (double curr_frequency)
- void initBox (double depth, double range)
- void initCfgFiles (double curr_frequency, int curr_run)
- void writeCfgFiles (double curr_frequency, int curr_run)
- void writeAllCfgFiles ()
- void writeNormalizedSSP (int curr_run)
- void writeSediment ()
- void writeHeader (double curr_frequency, int curr_run)
- void writeTransmitter ()
- void writeReceiver ()
- void writeRayOptions ()
- void writeBox ()
- void **writeBeamBlock** ()
- void writeBathymetryFile ()
- void writeBeamPatternFile ()
- void writeAltimetryFile (int curr_run)
- void removeAllCfgFiles ()
- void removeCfgFiles (double curr_frequency, int curr_run)
- void checkBoundaries (double &frequency, double &tx_depth, double &rx_start_depth, double &rx_start_range, double &rx_end_depth, double &rx_end_range) const
- void checkDepthOffsets ()
- void checkDepthOffsets (const CoordZ &coords, double &min_offset, double &max_offset, double min_depth_value, double max_depth_value)
- void checkAngles ()
- void checkRangeOffsets ()

Protected Attributes

- bool use_thorpe_att
- ::std::string beam_options
- ::std::string bathymetry_type
- ::std::string bathymetry_method
- ::std::string altimetry_type
- ::std::string bellhop_op_mode
- ::std::string bellhop_env_file
- ::std::string bathymetry_file
- ::std::string altimetry_file
- ::std::string beam_pattern_file
- ::std::string ssp_file

- ::std::string shd_file
- ::std::string arr_file
- ::std::string bellhop_path
- BellhopArrSyntax bellhop_arr_syntax
- BellhopShdSyntax bellhop_shd_syntax
- ::std::string curr_path
- ::std::string **transducer_type**
- double tx_min_depth_offset
- double tx_max_depth_offset
- int total_transmitters
- int total_rx_depths
- double rx_min_depth_offset
- double rx_max_depth_offset
- int total_rx_ranges
- double rx_min_range_offset
- double rx_max_range_offset
- int total_rays
- double min_angle
- double max_angle
- double min_normalized_ssp_depth
- double max_normalized_ssp_depth
- int curr_norm_ssp_depth_steps
- int transform_ssp_depth_steps
- const Transducer * transducer
- double **bp_initial_bearing**
- double **bp_vertical_rotation**
- double **bp_horizontal_rotation**
- double **bp_mult_costant**
- double **bp_add_costant**
- bool **using_ssp_file**
- bool **using_press_mode**
- bool **using_time_arrival_mode**
- NormSSPMap normalized_ssp_map
- NormSSPMap randomized_ssp_map
- double box_depth
- double box_range
- ::std::ofstream f_out

Additional Inherited Members

13.13.1 Detailed Description

Implempentation of ACToolboxWoss for Bellhop raytracing program.

BellhopWoss class has the task to create enviromental file, run Bellhop and read its results.

13.13.2 Constructor & Destructor Documentation

13.13.2.1 BellhopWoss() [1/2] BellhopWoss::BellhopWoss ( )

BellhopWoss default constructor. Default constructed objects are not valid

13.13.2.2 BellhopWoss() [2/2] BellhopWoss::BellhopWoss (

```
const CoordZ & tx,
const CoordZ & rx,
const Time & start_t,
const Time & end_t,
double start_freq,
double end_freq,
double freq_step )
```

BellhopWoss constructor

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_t	const reference to a valid Time object for SSP's averaging purposes (start date time)
end_t	const reference to a valid Time object for SSP's averaging purposes (end date time)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
freq_step	frequency step [Hz]

13.13.3 Member Function Documentation

13.13.3.1 checkAngles() void BellhopWoss::checkAngles ( ) [protected]

Checks the consistency of given launching angle and modifies them if needed

References woss::Woss::debug, max_angle, min_angle, woss::Woss::total_distance, woss::Woss::total_great_circle_distance, and woss::Woss::woss_id.

Referenced by initialize().

13.13.3.2 checkBoundaries() void BellhopWoss::checkBoundaries (

```
double & frequency,
double & tx_depth,
double & rx_start_depth,
double & rx_start_range,
double & rx_end_depth,
double & rx_end_range ) const [protected]
```

Checks parameters passed by a WossManager request and modifies them in place if they are not consistent with data

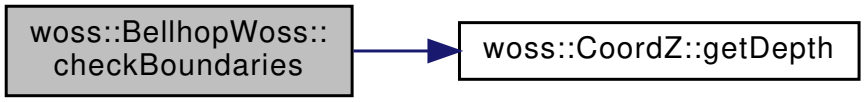
Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
rx_start_depth	receiver start depth [m]
rx_start_range	receiver start range [m]
rx_end_depth	receiver end depth [m]
rx_end_range	receiver end range [m]

References woss::Woss::frequencies, woss::CoordZ::getDepth(), woss::Woss::rx_coordz, rx_max_depth_offset, rx_max_range_offset, rx_min_depth_offset, rx_min_range_offset, woss::Woss::total_great_circle_distance, woss::Woss::tx_coordz, tx_max_depth_offset, and tx_min_depth_offset.

Referenced by getAvgPressure(), getPressure(), and getTimeArr().

Here is the call graph for this function:



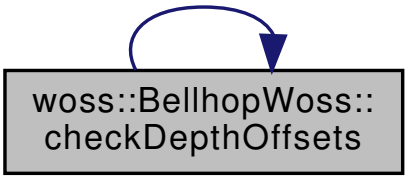
13.13.3.3 checkDepthOffsets() [1/2] void BellhopWoss::checkDepthOffsets ( ) [protected]

Checks the consistency of tx and rx depth offsets and modifies them if needed

References checkDepthOffsets(), woss::ACToolboxWoss::coordz_vector, woss::ACToolboxWoss::max_bathymetry_depth, max_normalized_ssp_depth, min_normalized_ssp_depth, woss::Woss::rx_coordz, rx_max_depth_offset, rx_min_depth_offset, woss::Woss::tx_coordz, tx_max_depth_offset, and tx_min_depth_offset.

Referenced by checkDepthOffsets(), and initialize().

Here is the call graph for this function:



13.13.3.4 checkDepthOffsets() [2/2] void BellhopWoss::checkDepthOffsets (const CoordZ & *coords*, double & *min_offset*, double & *max_offset*, double *min_depth_value*, double *max_depth_value*) [protected]

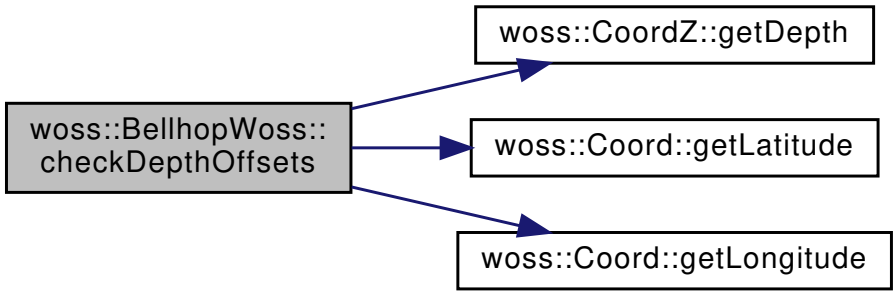
Checks the consistency of input offsets and modifies them if needed

Parameters

coords	3-D coordinates of the point to be tested
min_offset	min offset to be tested [m]
max_offset	max offset to be tested [m]
min_depth_value	min depth value of point to be tested [m]
max_depth_value	max depth value of point to be tested [m]

References woss::Woss::debug, woss::CoordZ::getDepth(), woss::Coord::getLatitude(), woss::Coord::getLongitude(), and woss::Woss::woss_id.

Here is the call graph for this function:



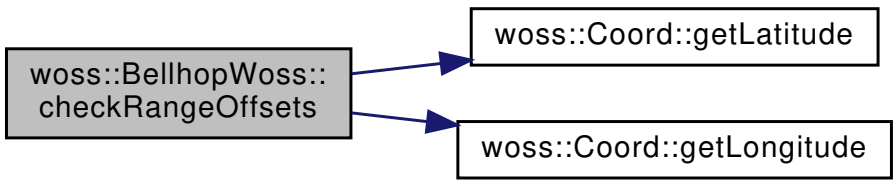
13.13.3.5 checkRangeOffsets() `void BellhopWoss::checkRangeOffsets ( ) [protected]`

Checks the consistency of rx range offsets and modifies them if needed

References woss::Woss::debug, woss::Coord::getLatitude(), woss::Coord::getLongitude(), woss::Woss::rx_coordz, rx_max_range_offset, rx_min_range_offset, woss::Woss::total_great_circle_distance, woss::Woss::tx_coordz, and woss::Woss::woss_id.

Referenced by initialize().

Here is the call graph for this function:



13.13.3.6 getAltimetryType() ::std::string woss::BellhopWoss::getAltimetryType ( ) const [inline]

Gets altimetry type string

Returns

"C" or "L"

References altimetry_type.

13.13.3.7 getAvgPressure()

```
Pressure * BellhopWoss::getAvgPressure (
        double frequency,
        double tx_depth,
        double start_rx_depth = WOSS_MIN_DEPTH,
        double start_rx_range = WOSS_MIN_RANGE,
        double end_rx_depth = WOSS_MAX_DEPTH,
        double end_rx_range = WOSS_MAX_RANGE ) const [virtual]
```

Gets the average Pressure value in given rx range-depth box

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

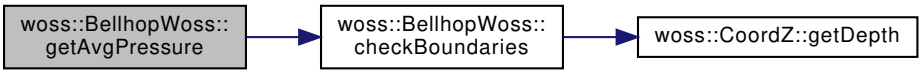
Returns

a valid Pressure value

Implements woss::Woss.

References checkBoundaries(), woss::WossResReader::res_reader_map, and woss::Woss::total_runs.

Here is the call graph for this function:



13.13.3.8 getBathymetryMethod() ::std::string woss::BellhopWoss::getBathymetryMethod ( ) const [inline]

Gets bathymetry write method string

Returns

"S" or "D"

References bathymetry_method.

13.13.3.9 getBathymetryType() ::std::string woss::BellhopWoss::getBathymetryType ( ) const [inline]

Gets bathymetry type string

Returns

"C" or "L"

References bathymetry_type.

13.13.3.10 getBeamOptions() ::std::string woss::BellhopWoss::getBeamOptions ( ) const [inline]

Gets the beam option string

Returns

"R" , "B", "C" or "G"

References beam_options.

13.13.3.11 getBellhopArrSyntax() BellhopArrSyntax woss::BellhopWoss::getBellhopArrSyntax ( ) const [inline]

Gets the .arr file sintax

Returns

arr file syntax

References bellhop_arr_syntax.

Referenced by woss::ArrBinResReader::getArrBinFile(), and woss::ArrBinResReader::getArrBinHeader().

13.13.3.12 getBellhopPath() ::std::string woss::BellhopWoss::getBellhopPath ( ) const [inline]

Gets the path of bellhop program

Returns

path of bellhop program

References bellhop_path.

13.13.3.13 getBellhopShdSyntax() BellhopShdSyntax woss::BellhopWoss::getBellhopShdSyntax ( ) const [inline]

Gets the .shd file sintax

Returns

shd file syntax

References bellhop_shd_syntax.

Referenced by woss::ShdResReader::getShdFile().

13.13.3.14 getBoxDepth() double woss::BellhopWoss::getBoxDepth ( ) const [inline]

Gets the maximum ray depth

Returns

maximum ray depth

References box_depth.

13.13.3.15 getBoxRange() double woss::BellhopWoss::getBoxRange ( ) const [inline]

Gets the maximum ray range

Returns

maximum ray range

References box_range.

13.13.3.16 getMaxAngle() `double woss::BellhopWoss::getMaxAngle ( ) const [inline]`

Gets the maximum launch angle

Returns

maximum launch angle

References max_angle.

13.13.3.17 getMinAngle() `double woss::BellhopWoss::getMinAngle ( ) const [inline]`

Gets the minimum launch angle

Returns

minimum launch angle

References min_angle.

13.13.3.18 getPressure()

```
Pressure * BellhopWoss::getPressure (
        double frequency,
        double tx_depth,
        double rx_depth,
        double rx_range ) const [virtual]
```

Gets a Pressure value of given range, depths

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

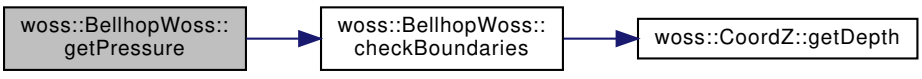
Returns

a valid Pressure value

Implements woss::Woss.

References checkBoundaries(), woss::WossResReader::res_reader_map, and woss::Woss::total_runs.

Here is the call graph for this function:



13.13.3.19 getRaysNumber() `int woss::BellhopWoss::getRaysNumber ( ) const [inline]`

Gets the total number of launched angles

Returns

number of launched angles

References total_rays.

13.13.3.20 getRxMaxDepthOffset() `double woss::BellhopWoss::getRxMaxDepthOffset ( ) const [inline]`

Gets the receiver maximum depth offset [m]

Returns

depth offset [m]

References rx_max_depth_offset.

13.13.3.21 getRxMaxRangeOffset() `double woss::BellhopWoss::getRxMaxRangeOffset ( ) const [inline]`

Gets the receiver maximum range offset [m]

Returns

range offset [m]

References rx_max_range_offset.

13.13.3.22 getRxMinDepthOffset() `double woss::BellhopWoss::getRxMinDepthOffset ( ) const [inline]`

Gets the receiver mimimum depth offset [m]

Returns

depth offset [m]

References rx_min_depth_offset.

13.13.3.23 getRxMinRangeOffset() `double woss::BellhopWoss::getRxMinRangeOffset ( ) const [inline]`

Gets the receiver mimimum range offset [m]

Returns

range offset [m]

References rx_min_range_offset.

13.13.3.24 getRxTotalDepths() `int woss::BellhopWoss::getRxTotalDepths ( ) const [inline]`

Gets the total number of receiver depths

Returns

number of receiver depths

References total_rx_depths.

13.13.3.25 getRxTotalRanges() `int woss::BellhopWoss::getRxTotalRanges ( ) const [inline]`

Gets the total number of receiver ranges

Returns

number of receiver ranges

References total_rx_ranges.

13.13.3.26 getThorpeAttFlag() `bool woss::BellhopWoss::getThorpeAttFlag ( ) const [inline]`

Gets the thorpe attenuation flag

Returns

boolean flag

References use_thorpe_att.

13.13.3.27 getTimeArr()

```
TimeArr * BellhopWoss::getTimeArr (
        double frequency,
        double tx_depth,
        double rx_depth,
        double rx_range ) const [virtual]
```

Gets a TimeArr value of given range, depths

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

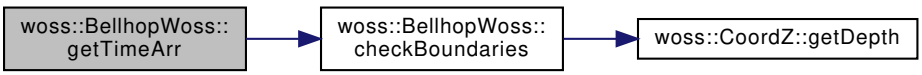
Returns

a valid Pressure value

Implements woss::Woss.

References checkBoundaries(), woss::Woss::debug, woss::WossResReader::res_reader_map, woss::Woss::total_runs, and woss::Woss::woss_id.

Here is the call graph for this function:



13.13.3.28 getTotalTransmitters() `int woss::BellhopWoss::getTotalTransmitters ( ) const [inline]`

Gets the number of transmitters

Returns

number of transmitters

References total_transmitters.

13.13.3.29 getTransducer() `const Transducer *const woss::BellhopWoss::getTransducer ( ) const [inline]`

Gets bathymetry type string

Returns

"C" or "L"

References transducer.

13.13.3.30 getTransformSSPDepthSteps() `int woss::BellhopWoss::getTransformSSPDepthSteps ( ) const [inline]`

Returns the depth steps (and possibly transform) of all SSP in use

Returns

transformed SSP depth steps

References transform_ssp_depth_steps.

13.13.3.31 getTxMaxDepthOffset() `double woss::BellhopWoss::getTxMaxDepthOffset ( ) const [inline]`

Gets the transmitter maximum depth offset [m]

Returns

depth offset [m]

References tx_max_depth_offset.

13.13.3.32 getTxMinDepthOffset() `double woss::BellhopWoss::getTxMinDepthOffset ( ) const [inline]`

Gets the transmitter mimimum depth offset [m]

Returns

depth offset [m]

References tx_min_depth_offset.

13.13.3.33 initBox()

```
void woss::BellhopWoss::initBox (
        double depth,
        double range ) [inline], [protected]
```

Initializes the Bellhop box

Parameters

depth	box depth [m]
range	box range [m]

References box_depth, and box_range.

Referenced by initialize().

13.13.3.34 initCfgFiles() `void BellhopWoss::initCfgFiles ( double curr_frequency, int curr_run ) [protected]`

Initializes the configuration file(s)

Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run number

References bellhop_env_file, curr_path, woss::Woss::current_time, woss::Woss::work_dir_path, and woss::Woss::woss_id.

Referenced by run(), and writeCfgFiles().

13.13.3.35 initialize() `bool BellhopWoss::initialize ( ) [virtual]`

Gets enviromental data and writes the env file

Returns

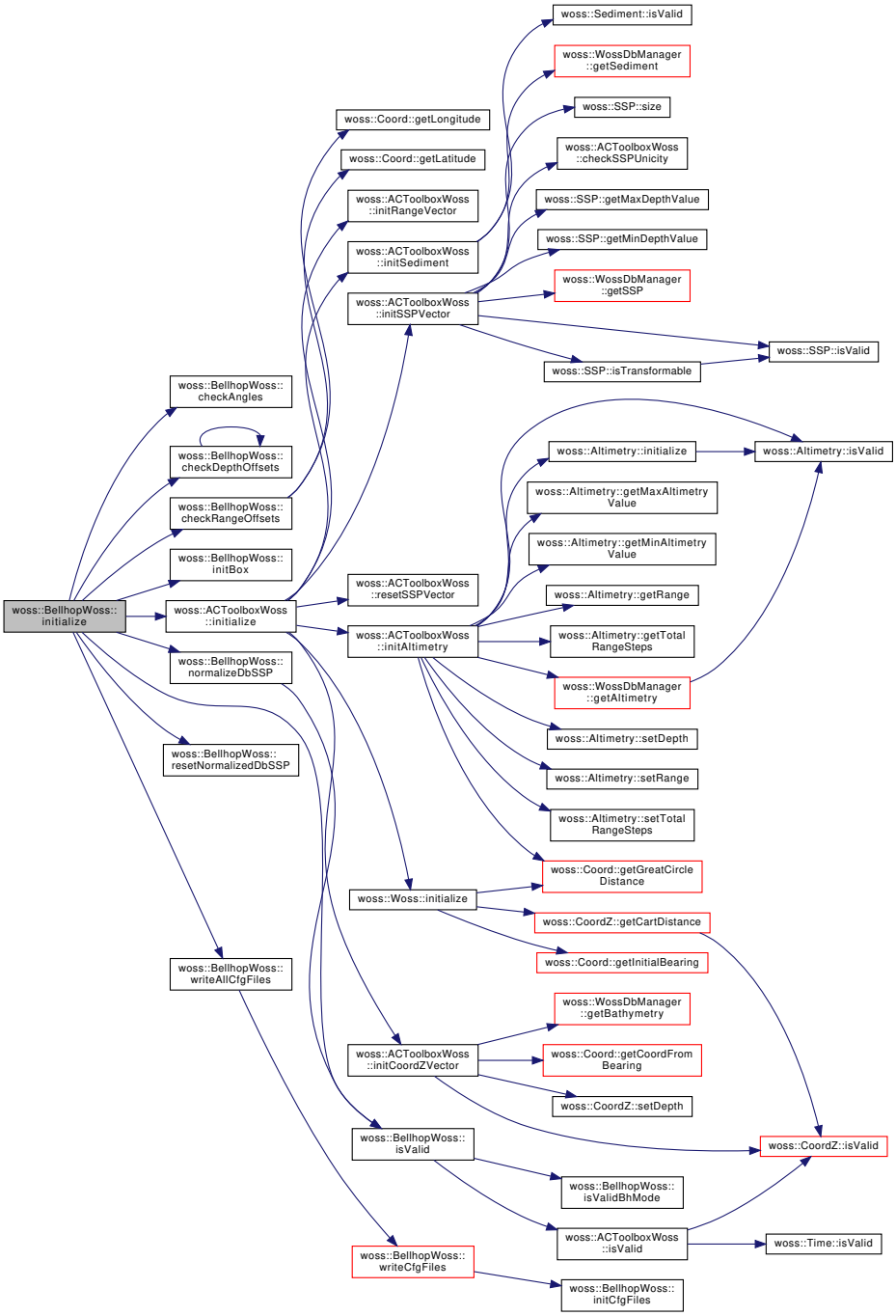
true if method was successful, *false* otherwise

Reimplemented from woss::ACToolboxWoss.

References checkAngles(), checkDepthOffsets(), checkRangeOffsets(), initBox(), woss::ACToolboxWoss::initialize(), isValid(), woss::ACToolboxWoss::max_bathymetry_depth, max_normalized_ssp_depth, normalizeDbSSP(), resetNormalizedDbSSP(), rx_max_range_offset, woss::Woss::total_distance, woss::Woss::total_great_circle_distance, and writeAllCfgFiles().

Referenced by initResReader(), and timeEvolve().

Here is the call graph for this function:



13.13.3.36 initPressResReader() `bool BellhopWoss::initPressResReader ( double curr_frequency ) [virtual]`

Initializes the ShdResReader object for given frequency

Parameters

curr_frequency	frequency in use [Hz]

Returns

true if method succeeded, *false* otherwise

References woss::WossResReader::res_reader_map, and shd_file.

Referenced by initResReader().

13.13.3.37 initResReader() `bool BellhopWoss::initResReader ( double curr_frequency ) [protected], [virtual]`

Initializes the ResReader object for given frequency

Parameters

curr_frequency	frequency in use [Hz]

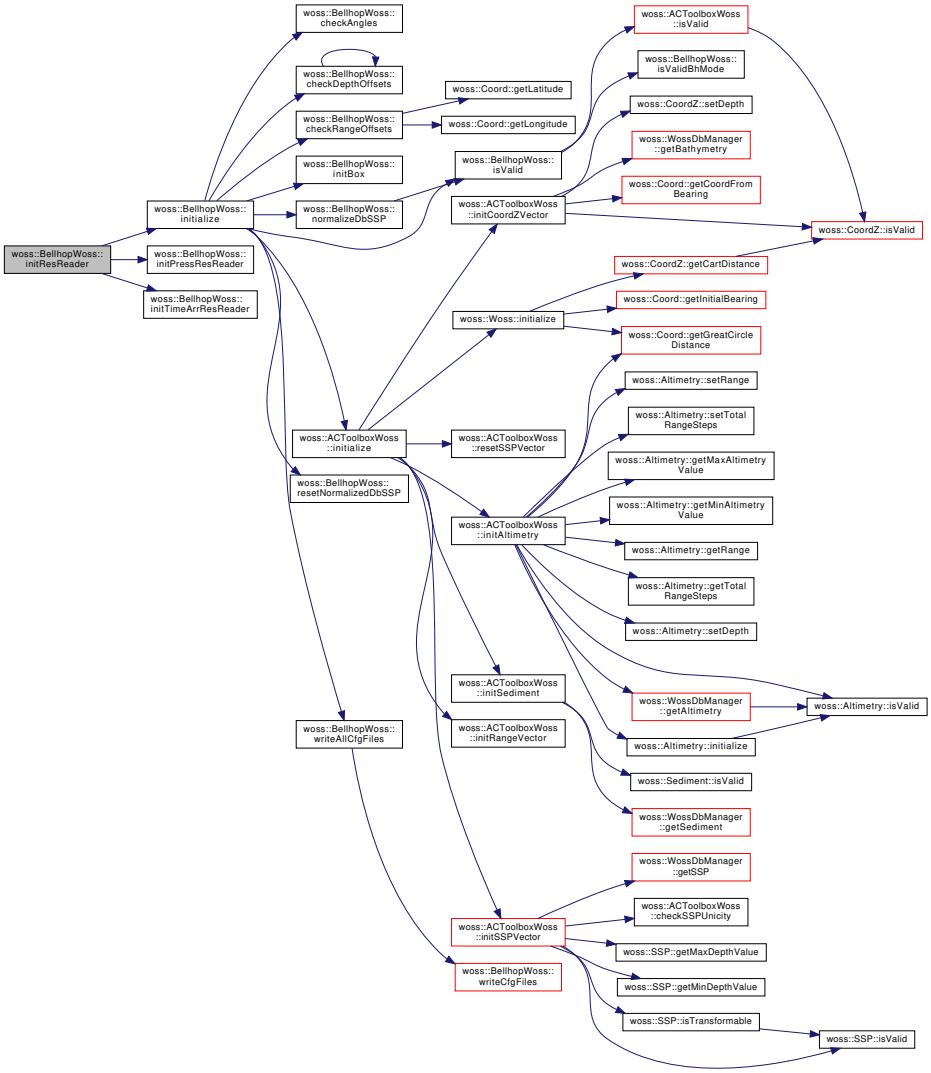
Returns

true if method succeeded, *false* otherwise

Implements woss::WossResReader.

References initialize(), initPressResReader(), initTimeArrResReader(), and woss::WossResReader::res_reader_map.

Here is the call graph for this function:



13.13.3.38 initTimeArrResReader() `bool BellhopWoss::initTimeArrResReader ( double curr_frequency ) [virtual]`

Initializes the ArrAscResReader or ArrBinResReader object for given frequency

Parameters

curr_frequency	frequency in use [Hz]

Returns

true if method succeeded, *false* otherwise

References arr_file, bellhop_op_mode, and woss::WossResReader::res_reader_map.

Referenced by initResReader().

13.13.3.39 isValid() `bool BellhopWoss::isValid ( ) const [virtual]`

Checks the validity of Woss

Returns

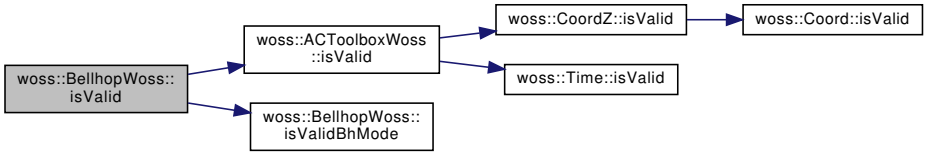
true if it's valid, *false* otherwise

Reimplemented from woss::ACToolboxWoss.

References bellhop_op_mode, woss::ACToolboxWoss::isValid(), isValidBhMode(), max_angle, min_angle, woss::ACToolboxWoss::total_range_steps, total_rays, total_rx_depths, total_rx_ranges, and total_transmitters.

Referenced by initialize(), and normalizeDbSSP().

Here is the call graph for this function:



13.13.3.40 isValidBhMode() `bool woss::BellhopWoss::isValidBhMode ( const ::std::string & mode ) const [inline]`

Checks the validity of Bellhop run mode

Returns

true if it's valid, *false* otherwise

Referenced by isValid(), and setBhMode().

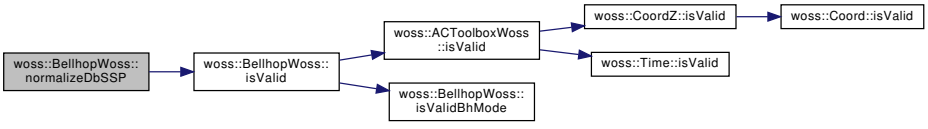
13.13.3.41 normalizeDbSSP() `void BellhopWoss::normalizeDbSSP ( ) [protected], [virtual]`

Normalizes the SSP matrix for Bellhop requirements (same depths and same depth-steps for all SSP involved)

References curr_norm_ssp_depth_steps, woss::Woss::debug, woss::ACToolboxWoss::is_ssp_vector_transformable, isValid(), woss::ACToolboxWoss::max_bathymetry_depth, max_normalized_ssp_depth, woss::ACToolboxWoss::max_ssp_depth_set, woss::ACToolboxWoss::max_ssp_depth_steps, woss::ACToolboxWoss::min_altimetry_depth, min_normalized_ssp_depth, woss::ACToolboxWoss::min_ssp_depth_set, woss::ACToolboxWoss::min_ssp_depth_steps, normalized_ssp_map, woss::ACToolboxWoss::range_vector, woss::ACToolboxWoss::ssp_unique_indexes, woss::ACToolboxWoss::ssp_vector, transform_ssp_depth_steps, and woss::Woss::tx_coordz.

Referenced by initialize().

Here is the call graph for this function:



13.13.3.42 removeAllCfgFiles() `void BellhopWoss::removeAllCfgFiles ( ) [protected]`

Writes all configuration files

References woss::Woss::frequencies, removeCfgFiles(), and woss::Woss::total_runs.

Here is the call graph for this function:

13.13.3.43 removeCfgFiles() `void BellhopWoss::removeCfgFiles ( double curr_frequency, int curr_run ) [protected]`

Removes the configuration file(s)

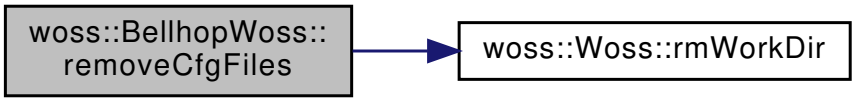
Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run number

References woss::Woss::rmWorkDir().

Referenced by removeAllCfgFiles().

Here is the call graph for this function:



13.13.3.44 resetNormalizedDbSSP() `void BellhopWoss::resetNormalizedDbSSP ( ) [protected], [virtual]`

Deletes all normalized SSPs

References normalized_ssp_map.

Referenced by initialize().

13.13.3.45 run() bool BellhopWoss::run ( ) [virtual]

Runs the channel simulator. It is mandatory to set **has_run** to *true* before returning from this function

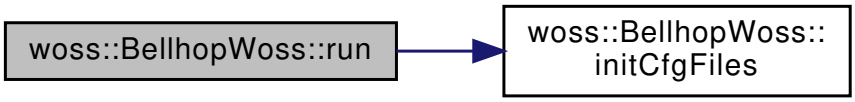
Returns

true if method was successful, *false* otherwise

Implements woss::Woss.

References bellhop_arr_syntax, woss::BELLHOP_CREATOR_ARR_FILE_INVALID, woss::BELLHOP_CREATOR_SHD_FILE_INVAL bellhop_path, bellhop_shd_syntax, curr_path, woss::Woss::debug, woss::Woss::frequencies, initCfgFiles(), woss::Woss::is_running, woss::Woss::total_runs, and woss::Woss::woss_id.

Here is the call graph for this function:



13.13.3.46 setAltimetryType() BellhopWoss & woss::BellhopWoss::setAltimetryType (const ::std::string & *type*) [inline]

Sets the altimetry type (L, C). See Bellhop documentation for more info

Parameters

type	beam options

Returns

reference to ∗**this**

References altimetry_type.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.47 setBathymetryMethod() BellhopWoss & woss::BellhopWoss::setBathymetryMethod (const ::std::string & *type*) [inline]

Sets the bathymetry write method (S, D)

Parameters

type	beam options

Returns

reference to ∗**this**

References bathymetry_method.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.48 setBathymetryType()

`BellhopWoss & woss::BellhopWoss::setBathymetryType ( const ::std::string & type ) [inline]`

Sets the bathymetry type (L, C). See Bellhop documentation for more info

Parameters

type	beam options

Returns

reference to ∗**this**

References bathymetry_type.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.49 setBeamOptions()

`BellhopWoss & woss::BellhopWoss::setBeamOptions ( const ::std::string & options ) [inline]`

Sets the beam option string . See Bellhop documentation for more info

Parameters

options	beam options (G, C, R , B)

Returns

reference to ∗**this**

References beam_options.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.50 setBeamPatternParam()

```
BellhopWoss & woss::BellhopWoss::setBeamPatternParam (
        double init_bearing,
        double vert_rot = 0.0,
        double horiz_rot = 0.0,
        double mult = 1.0,
        double add = 0.0 )  [inline]
```

Sets the transmitter beam pattern

Parameters

type	valid woss::BeamPattern

Returns

reference to ∗**this**

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.51 setBellhopArrSyntax()

```
BellhopWoss & woss::BellhopWoss::setBellhopArrSyntax (
        BellhopArrSyntax syntax )  [inline]
```

Sets the bellhop arr file syntax

Parameters

syntax	syntax to be used

Returns

reference to ∗**this**

References bellhop_arr_syntax.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.52 setBellhopPath()

```
BellhopWoss & woss::BellhopWoss::setBellhopPath (
        const ::std::string & path )  [inline]
```

Sets the path of bellhop program

Parameters

path	filesystem path

Returns

reference to ∗**this**

References bellhop_path.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.53 setBellhopShdSyntax() `BellhopWoss & woss::BellhopWoss::setBellhopShdSyntax ( BellhopShdSyntax syntax ) [inline]`

Sets the bellhop shd file syntax

Parameters

syntax	syntax to be used

Returns

reference to ∗**this**

References bellhop_shd_syntax.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.54 setBhMode() `BellhopWoss & BellhopWoss::setBhMode ( const ::std::string & mode )`

Sets the Bellhop run mode string. See Bellhop documentation for more info

Parameters

mode	Bellhop run mode (A, a , C, I , S)

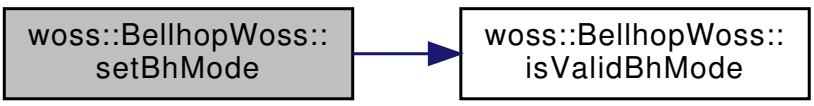
Returns

reference to ∗**this**

References bellhop_op_mode, isValidBhMode(), and woss::Woss::woss_id.

Referenced by woss::BellhopCreator::initializeBhWoss().

Here is the call graph for this function:



13.13.3.55 setBoxDepth() BellhopWoss & woss::BellhopWoss::setBoxDepth (double *depth*) [inline]

Sets the maximum depth for Bellhop rays

Parameters

depth	maximum ray depth [m]

Returns

reference to ∗**this**

References box_depth.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.56 setBoxRange() BellhopWoss & woss::BellhopWoss::setBoxRange (double *range*) [inline]

Sets the maximum range for Bellhop rays

Parameters

range	maximum ray range [m]

Returns

reference to ∗**this**

References box_range.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.57 setMaxAngle() BellhopWoss & woss::BellhopWoss::setMaxAngle (double *angle*) [inline]

Sets the maximum launch angle for all BellhopWoss that will be created

Parameters

angle	number of launched rays (>= 0)

Returns

reference to ∗**this**

References max_angle.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.58 setMinAngle() `BellhopWoss & woss::BellhopWoss::setMinAngle ( double angle ) [inline]`

Sets the mimimum launch angle for all BellhopWoss that will be created

Parameters

angle	number of launched rays (>= 0)

Returns

reference to ∗**this**

References min_angle.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.59 setRaysNumber() `BellhopWoss & woss::BellhopWoss::setRaysNumber ( int number ) [inline]`

Sets the number of launched rays

Parameters

number	number of launched rays (>= 0)

Returns

reference to ∗**this**

References total_rays.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.60 setRxMaxDepthOffset() `BellhopWoss & woss::BellhopWoss::setRxMaxDepthOffset ( double offset ) [inline]`

Sets the receiver maximum depth offset [m]

Parameters

offset	0 <= depth offset <= 0 [m]

Returns

reference to ∗**this**

References rx_max_depth_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.61 setRxMaxRangeOffset() `BellhopWoss & woss::BellhopWoss::setRxMaxRangeOffset ( double offset ) [inline]`

Sets the receiver maximum depth offset [m]

Parameters

offset	-total_distance <= range offset <= total_distance [m]

Returns

reference to ∗**this**

References rx_max_range_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.62 setRxMinDepthOffset() `BellhopWoss & woss::BellhopWoss::setRxMinDepthOffset ( double offset ) [inline]`

Sets the receiver mimimum depth offset [m]

Parameters

offset	0 <= depth offset <= 0 [m]

Returns

reference to ∗**this**

References rx_min_depth_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.63 setRxMinRangeOffset() BellhopWoss & woss::BellhopWoss::setRxMinRangeOffset (double *offset*) [inline]

Sets the receiver mimimum range offset [m]

Parameters

offset	-total_distance <= range offset <= total_distance [m]

Returns

reference to ∗**this**

References rx_min_range_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.64 setRxTotalDepths() BellhopWoss & woss::BellhopWoss::setRxTotalDepths (int *number*) [inline]

Sets the number of receiver depths

Parameters

number	number of receiver depths

Returns

reference to ∗**this**

References total_rx_depths.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.65 setRxTotalRanges() BellhopWoss & woss::BellhopWoss::setRxTotalRanges (int *number*) [inline]

Sets the number of receiver ranges

Parameters

number	number of receiver ranges

Returns

reference to ∗**this**

References total_rx_ranges.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.66 setThorpeAttFlag() BellhopWoss & woss::BellhopWoss::setThorpeAttFlag (bool *flag*) [inline]

Sets the thorpe attenuation flag

Parameters

flag	boolean flag

Returns

reference to ∗**this**

References use_thorpe_att.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.67 setTotalTransmitters() BellhopWoss & woss::BellhopWoss::setTotalTransmitters (int *sources*) [inline]

Sets the number of transmitters

Parameters

sources	number of transmitters

Returns

reference to ∗**this**

References total_transmitters.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.68 setTransducer() BellhopWoss & woss::BellhopWoss::setTransducer (const Transducer ∗const *ptr*) [inline]

Sets the transducer type. See TransducerDbCreator for more info

Parameters

type	tranducer type

Returns

reference to ∗**this**

References transducer.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.69 setTransformSSPDepthSteps()

`BellhopWoss & woss::BellhopWoss::setTransformSSPDepthSteps ( int depth_steps ) [inline]`

Sets the depth steps (and possibly transform) of all SSP in use

Parameters

type	beam options

Returns

reference to ∗**this**

References transform_ssp_depth_steps.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.70 setTxMaxDepthOffset()

`BellhopWoss & woss::BellhopWoss::setTxMaxDepthOffset ( double offset ) [inline]`

Sets the transmitter maximum depth offset [m]

Parameters

offset	0 <= depth offset <= 0 [m]

Returns

reference to ∗**this**

References tx_max_depth_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.71 setTxMinDepthOffset()

```
BellhopWoss & woss::BellhopWoss::setTxMinDepthOffset (
        double offset ) [inline]
```

Sets the transmitter mimimum depth offset [m]

Parameters

offset	0 <= depth offset <= 0 [m]

Returns

reference to ∗**this**

References tx_min_depth_offset.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.13.3.72 timeEvolve()

```
bool BellhopWoss::timeEvolve (
        const Time & time_value ) [virtual]
```

Performs a time evoulion of all time-dependant parameters

Parameters

time_value	constant reference to a valid Time object (between start_time and end_time)

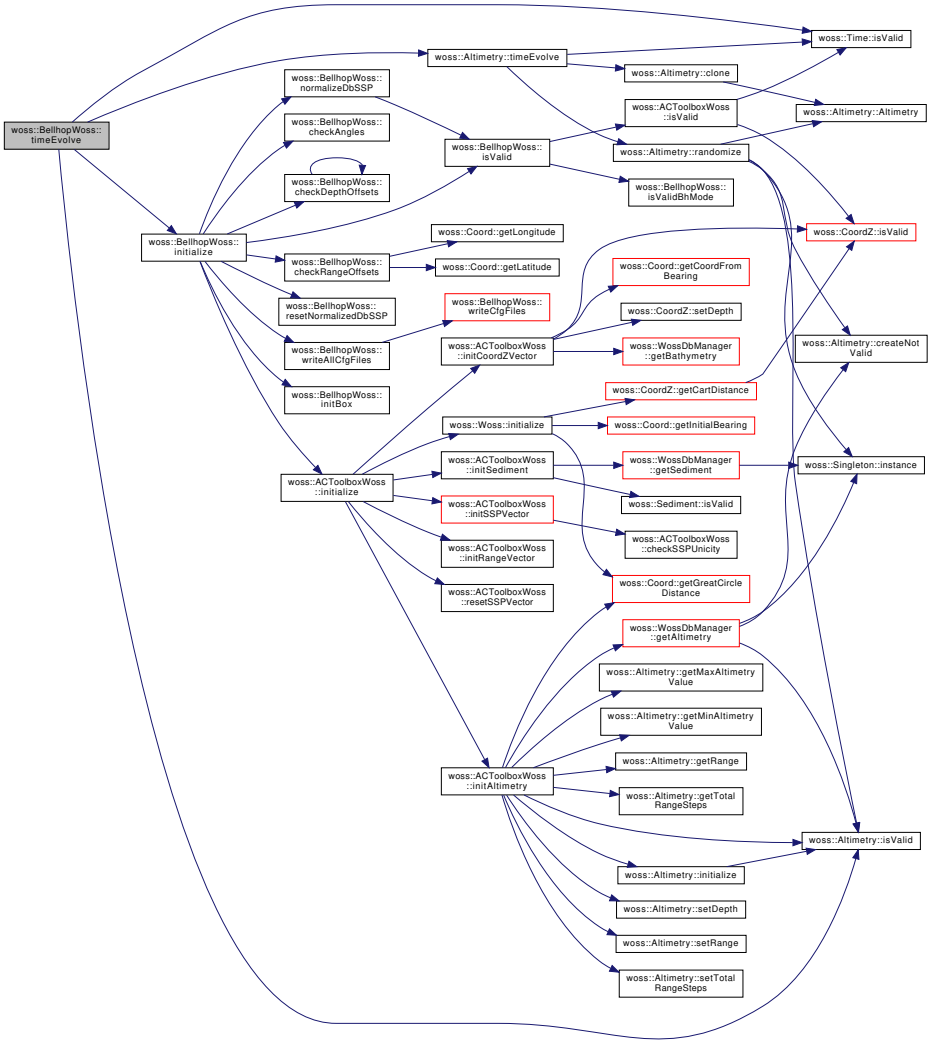
Returns

true if method was successful, *false* otherwise

Implements woss::Woss.

References woss::ACToolboxWoss::altimetry_value, woss::Woss::current_time, woss::Woss::debug, woss::Woss::end_time, woss::Woss::evolution_time_quantum, initialize(), woss::Altimetry::isValid(), woss::Time::isValid(), woss::Woss::start_time, woss::Altimetry::timeEvolve(), and woss::Woss::woss_id.

Here is the call graph for this function:



13.13.3.73 usingPressMode() `bool woss::BellhopWoss::usingPressMode ( ) const [inline]`

Signals if the instance is using Bellhop in SHD mode (C, S , I)

Returns

true if it is using it, *false* otherwise

13.13.3.74 usingSSPFile() `bool woss::BellhopWoss::usingSSPFile ( ) const [inline]`

Signals if the instance is using Bellhop with SSP matrix file

Returns

true if it is using it, *false* otherwise

13.13.3.75 usingTimeArrMode() `bool woss::BellhopWoss::usingTimeArrMode ( ) const [inline]`

Signals if the instance is using Bellhop in ARR mode (A, a)

Returns

true if it is using it, *false* otherwise

13.13.3.76 writeAllCfgFiles() `void BellhopWoss::writeAllCfgFiles ( ) [protected]`

Writes all configuration files

References woss::Woss::frequencies, woss::Woss::total_runs, and writeCfgFiles().

Referenced by initialize().

Here is the call graph for this function:

13.13.3.77 writeAltimetryFile() `void BellhopWoss::writeAltimetryFile ( int curr_run ) [protected]`

Writes db created altimetry in the configuration file(s)

Parameters

curr_run	current run number

References altimetry_file.

13.13.3.78 writeBathymetryFile() `void BellhopWoss::writeBathymetryFile ( ) [protected]`

Writes db created bathymetry in the configuration file(s)

References bathymetry_file.

13.13.3.79 writeBeamPatternFile() `void BellhopWoss::writeBeamPatternFile ( ) [protected]`

Writes transmitter beam pattern file(s)

References beam_pattern_file.

13.13.3.80 writeBox() `void woss::BellhopWoss::writeBox ( ) [inline], [protected]`

Writes the box options in the configuration file(s)

References box_depth, box_range, and f_out.

13.13.3.81 writeCfgFiles() `void BellhopWoss::writeCfgFiles ( double curr_frequency, int curr_run ) [protected]`

Writes the configuration file(s)

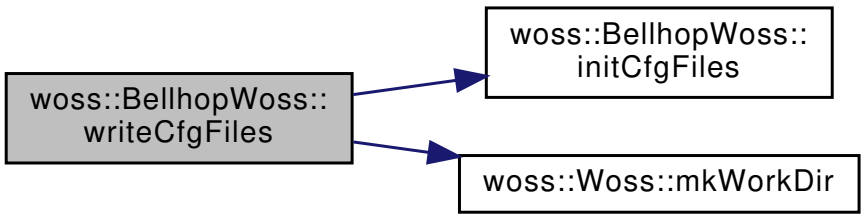
Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run number

References bellhop_env_file, f_out, initCfgFiles(), and woss::Woss::mkWorkDir().

Referenced by writeAllCfgFiles().

Here is the call graph for this function:



13.13.3.82 writeHeader() `void woss::BellhopWoss::writeHeader ( double curr_frequency, int curr_run ) [inline], [protected]`

Writes the header in the configuration file(s)

Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run number

References f_out.

13.13.3.83 writeNormalizedSSP() `void BellhopWoss::writeNormalizedSSP ( int curr_run ) [protected]`

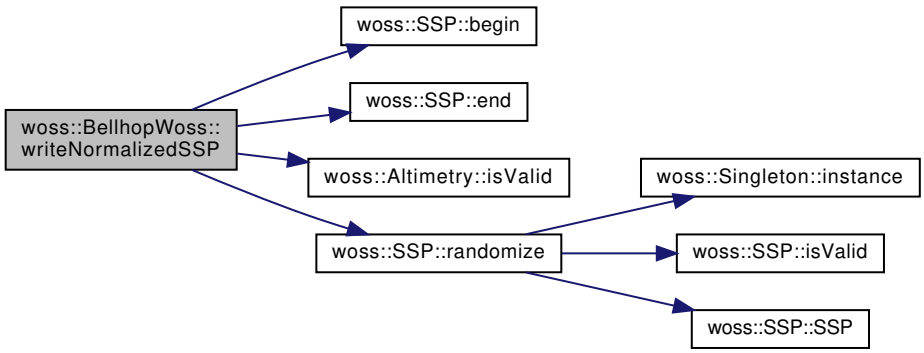
Writes the normalized SSP matrix in the configuration file(s)

Parameters

curr_run	current run number

References woss::ACToolboxWoss::altimetry_value, woss::SSP::begin(), woss::SSP::end(), f_out, woss::Altimetry::isValid(), max_normalized_ssp_depth, min_normalized_ssp_depth, normalized_ssp_map, woss::SSP::randomize(), randomized_ssp_map, ssp_file, and use_thorpe_att.

Here is the call graph for this function:

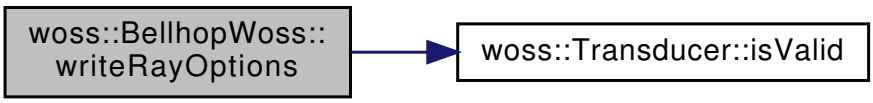


13.13.3.84 writeRayOptions() `void woss::BellhopWoss::writeRayOptions ( ) [inline], [protected]`

Writes the ray options in the configuration file(s)

References beam_options, bellhop_op_mode, f_out, woss::Transducer::isValid(), max_angle, min_angle, total_rays, and transducer.

Here is the call graph for this function:

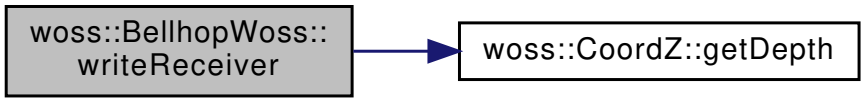


13.13.3.85 writeReceiver() `void woss::BellhopWoss::writeReceiver ( ) [inline], [protected]`

Writes the receiver info in the configuration file(s)

References f_out, woss::CoordZ::getDepth(), woss::Woss::rx_coordz, rx_max_depth_offset, rx_max_range_offset, rx_min_depth_offset, rx_min_range_offset, woss::Woss::total_great_circle_distance, total_rx_depths, and total_rx_ranges.

Here is the call graph for this function:

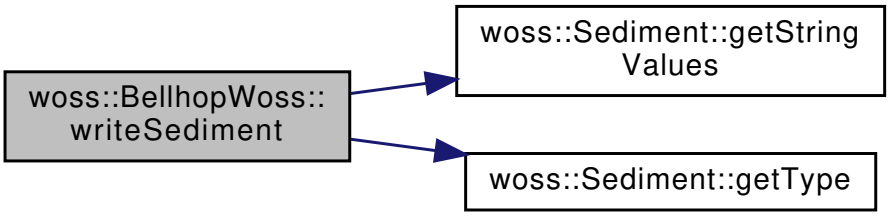


13.13.3.86 writeSediment() `void woss::BellhopWoss::writeSediment ( ) [inline], [protected]`

Writes the db Sediment in the configuration file(s)

References f_out, woss::Sediment::getStringValues(), woss::Sediment::getType(), max_normalized_ssp_depth, and woss::ACToolboxWoss::sediment_value.

Here is the call graph for this function:

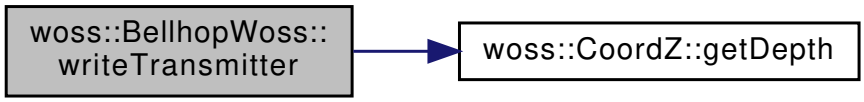


13.13.3.87 writeTransmitter() `void woss::BellhopWoss::writeTransmitter ( ) [inline], [protected]`

Writes the transmitter info in the configuration file(s)

References f_out, woss::CoordZ::getDepth(), total_transmitters, woss::Woss::tx_coordz, tx_max_depth_offset, and tx_min_depth_offset.

Here is the call graph for this function:



13.13.4 Member Data Documentation

13.13.4.1 altimetry_file ::std::string woss::BellhopWoss::altimetry_file [protected]

Pathname of Bellhop altimetry file

Referenced by writeAltimetryFile().

13.13.4.2 altimetry_type ::std::string woss::BellhopWoss::altimetry_type [protected]

Altimetry type (L, C)

Referenced by getAltimetryType(), and setAltimetryType().

13.13.4.3 arr_file ::std::string woss::BellhopWoss::arr_file [protected]

Pathname of Bellhop ARR file

Referenced by initTimeArrResReader().

13.13.4.4 bathymetry_file ::std::string woss::BellhopWoss::bathymetry_file [protected]

Pathname of Bellhop bathymetry file

Referenced by writeBathymetryFile().

13.13.4.5 bathymetry_method ::std::string woss::BellhopWoss::bathymetry_method [protected]

Bathymetry write method (S, D)

Referenced by getBathymetryMethod(), and setBathymetryMethod().

13.13.4.6 bathymetry_type ::std::string woss::BellhopWoss::bathymetry_type [protected]

Bathymetry type (L, C)

Referenced by getBathymetryType(), and setBathymetryType().

13.13.4.7 beam_options ::std::string woss::BellhopWoss::beam_options [protected]

Bellhop beam options (G, C, R, B)

Referenced by getBeamOptions(), setBeamOptions(), and writeRayOptions().

13.13.4.8 beam_pattern_file ::std::string woss::BellhopWoss::beam_pattern_file [protected]

Pathname of Bellhop beam pattern file

Referenced by writeBeamPatternFile().

13.13.4.9 bellhop_arr_syntax BellhopArrSyntax woss::BellhopWoss::bellhop_arr_syntax [protected]

.arr file syntax to be used during file parsing

Referenced by getBellhopArrSyntax(), run(), and setBellhopArrSyntax().

13.13.4.10 bellhop_env_file ::std::string woss::BellhopWoss::bellhop_env_file [protected]

Pathname of Bellhop configuration file

Referenced by initCfgFiles(), and writeCfgFiles().

13.13.4.11 bellhop_op_mode ::std::string woss::BellhopWoss::bellhop_op_mode [protected]

Bellhop run mode (A, a, C, I, S)

Referenced by initTimeArrResReader(), isValid(), setBhMode(), and writeRayOptions().

13.13.4.12 bellhop_path ::std::string woss::BellhopWoss::bellhop_path [protected]

Pathname of Bellhop program

Referenced by getBellhopPath(), run(), and setBellhopPath().

13.13.4.13 bellhop_shd_syntax `BellhopShdSyntax woss::BellhopWoss::bellhop_shd_syntax [protected]`

.shd file syntax to be used during file parsing

Referenced by getBellhopShdSyntax(), run(), and setBellhopShdSyntax().

13.13.4.14 box_depth `double woss::BellhopWoss::box_depth [protected]`

Bellhop box depth [m]

Referenced by getBoxDepth(), initBox(), setBoxDepth(), and writeBox().

13.13.4.15 box_range `double woss::BellhopWoss::box_range [protected]`

Bellhop box range [m]

Referenced by getBoxRange(), initBox(), setBoxRange(), and writeBox().

13.13.4.16 curr_norm_ssp_depth_steps `int woss::BellhopWoss::curr_norm_ssp_depth_steps [protected]`

Depth steps of all SSP involved

Referenced by normalizeDbSSP().

13.13.4.17 curr_path `::std::string woss::BellhopWoss::curr_path [protected]`

Current working path

Referenced by initCfgFiles(), and run().

13.13.4.18 f_out `::std::ofstream woss::BellhopWoss::f_out [protected]`

ofstream for file writing

Referenced by writeBox(), writeCfgFiles(), writeHeader(), writeNormalizedSSP(), writeRayOptions(), writeReceiver(), writeSediment(), and writeTransmitter().

13.13.4.19 max_angle `double woss::BellhopWoss::max_angle [protected]`

Maximum launching angle [decimal degrees]

Referenced by checkAngles(), getMaxAngle(), isValid(), setMaxAngle(), and writeRayOptions().

13.13.4.20 max_normalized_ssp_depth `double woss::BellhopWoss::max_normalized_ssp_depth [protected]`

Maximum depth of normalized SSP matrix [m]

Referenced by checkDepthOffsets(), initialize(), normalizeDbSSP(), writeNormalizedSSP(), and writeSediment().

13.13.4.21 min_angle `double woss::BellhopWoss::min_angle [protected]`

Minimum launching angle [decimal degrees]

Referenced by checkAngles(), getMinAngle(), isValid(), setMinAngle(), and writeRayOptions().

13.13.4.22 min_normalized_ssp_depth `double woss::BellhopWoss::min_normalized_ssp_depth [protected]`

Minimum depth of normalized SSP matrix [m]

Referenced by checkDepthOffsets(), normalizeDbSSP(), and writeNormalizedSSP().

13.13.4.23 normalized_ssp_map `NormSSPMap woss::BellhopWoss::normalized_ssp_map [protected]`

SSP matrix normalized for Bellhop requirements (same depths and same depth-steps for all SSP involved)

Referenced by normalizeDbSSP(), resetNormalizedDbSSP(), and writeNormalizedSSP().

13.13.4.24 randomized_ssp_map `NormSSPMap woss::BellhopWoss::randomized_ssp_map [protected]`

SSP matrix normalized and randomized for Bellhop requirements (same depths and same depth-steps for all SSP involved)

Referenced by writeNormalizedSSP().

13.13.4.25 rx_max_depth_offset `double woss::BellhopWoss::rx_max_depth_offset [protected]`

Receiver maximum depth offset [m]

Referenced by checkBoundaries(), checkDepthOffsets(), getRxMaxDepthOffset(), setRxMaxDepthOffset(), and writeReceiver().

13.13.4.26 rx_max_range_offset `double woss::BellhopWoss::rx_max_range_offset [protected]`

Receiver maximum range offset [m]

Referenced by checkBoundaries(), checkRangeOffsets(), getRxMaxRangeOffset(), initialize(), setRxMaxRangeOffset(), and writeReceiver().

13.13.4.27 rx_min_depth_offset `double woss::BellhopWoss::rx_min_depth_offset [protected]`

Receiver minimum depth offset [m]

Referenced by checkBoundaries(), checkDepthOffsets(), getRxMinDepthOffset(), setRxMinDepthOffset(), and writeReceiver().

13.13.4.28 rx_min_range_offset `double woss::BellhopWoss::rx_min_range_offset [protected]`

Receiver minimum range offset [m]

Referenced by checkBoundaries(), checkRangeOffsets(), getRxMinRangeOffset(), setRxMinRangeOffset(), and writeReceiver().

13.13.4.29 shd_file `::std::string woss::BellhopWoss::shd_file [protected]`

Pathname of Bellhop SHD file

Referenced by initPressResReader().

13.13.4.30 ssp_file `::std::string woss::BellhopWoss::ssp_file [protected]`

Pathname of Bellhop SSP file

Referenced by writeNormalizedSSP().

13.13.4.31 total_rays `int woss::BellhopWoss::total_rays [protected]`

Number of launched rays

Referenced by getRaysNumber(), isValid(), setRaysNumber(), and writeRayOptions().

13.13.4.32 total_rx_depths `int woss::BellhopWoss::total_rx_depths [protected]`

Number of receiver depths

Referenced by getRxTotalDepths(), isValid(), setRxTotalDepths(), and writeReceiver().

13.13.4.33 total_rx_ranges `int woss::BellhopWoss::total_rx_ranges [protected]`

Number of receiver ranges.

On some configuration (linux distribution / cpu) bellhop will output an empty file with a value of *total_rx_ranges* = 1.

Referenced by getRxTotalRanges(), isValid(), setRxTotalRanges(), and writeReceiver().

13.13.4.34 total_transmitters `int woss::BellhopWoss::total_transmitters [protected]`

Number of transmitter

Referenced by getTotalTransmitters(), isValid(), setTotalTransmitters(), and writeTransmitter().

13.13.4.35 transducer `const Transducer* woss::BellhopWoss::transducer [protected]`

Transmitter beam pattern

Referenced by getTransducer(), setTransducer(), and writeRayOptions().

13.13.4.36 transform_ssp_depth_steps `int woss::BellhopWoss::transform_ssp_depth_steps [protected]`

transformed SSP depth steps. Set <= 0 in order to disable the feature.

Referenced by getTransformSSPDepthSteps(), normalizeDbSSP(), and setTransformSSPDepthSteps().

13.13.4.37 tx_max_depth_offset `double woss::BellhopWoss::tx_max_depth_offset [protected]`

Transmitter maximum depth offset [m]

Referenced by checkBoundaries(), checkDepthOffsets(), getTxMaxDepthOffset(), setTxMaxDepthOffset(), and writeTransmitter().

13.13.4.38 tx_min_depth_offset `double woss::BellhopWoss::tx_min_depth_offset [protected]`

Transmitter minimum depth offset [m]

Referenced by checkBoundaries(), checkDepthOffsets(), getTxMinDepthOffset(), setTxMinDepthOffset(), and writeTransmitter().

13.13.4.39 use_thorpe_att `bool woss::BellhopWoss::use_thorpe_att [protected]`

let bellhop calculate thorpe attenuation

Referenced by getThorpeAttFlag(), setThorpeAttFlag(), and writeNormalizedSSP().

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

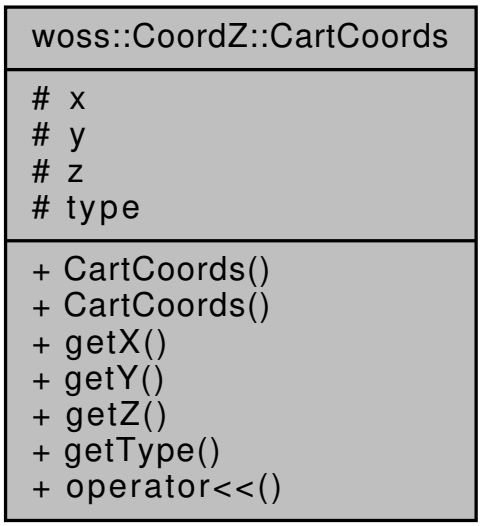
- woss/bellhop-woss.h
- woss/bellhop-woss.cpp

13.14 woss::CoordZ::CartCoords Class Reference

Class that represents cartesian coordinates.

`#include <coordinates-definitions.h>`

Collaboration diagram for woss::CoordZ::CartCoords:



Public Member Functions

- CartCoords ()
- CartCoords (double in_x, double in_y, double in_z, CoordZSpheroidType in_type)
- double getX () const
- double getY () const
- double getZ () const
- CoordZSpheroidType getType () const
- friend::std::ostream & operator<< (::std::ostream &os, const CartCoords &instance)

Protected Attributes

- double **x**
- double **y**

  X-asis value in meters.
- double **z**

  Y-asis value in meters.
- CoordZSpheroidType **type**

  Z-asis value in meters.

13.14.1 Detailed Description

Class that represents cartesian coordinates.

Class that represents cartesian coordinates

13.14.2 Constructor & Destructor Documentation

13.14.2.1 CartCoords() [1/2] `CoordZ::CartCoords::CartCoords ( )`

CartCoords default constructor.

13.14.2.2 CartCoords() [2/2]

```
CoordZ::CartCoords::CartCoords (
        double in_x,
        double in_y,
        double in_z,
        CoordZSpheroidType in_type )
```

Coord constructor.

Parameters

in_x	number of **meters** along the X axis
in_y	number of **meters** along the X axis
in_z	number of **meters** along the X axis
in_type	Spheroid model that has been used

13.14.3 Member Function Documentation

13.14.3.1 getType() `CoordZSpheroidType woss::CoordZ::CartCoords::getType ( ) const [inline]`

Returns the CoordZSpheroidType used for computation

Returns

CoordZSpheroidType spheroid model

References type.

Referenced by woss::CoordZ::getCoordZFromCartesianCoords().

13.14.3.2 getX() `double woss::CoordZ::CartCoords::getX ( ) const [inline]`

Returns the number of meters along the X axis

Returns

x axis value [m]

Referenced by woss::CoordZ::getCartDistance(), woss::CoordZ::getCartRelAzimuth(), woss::CoordZ::getCartX(), woss::CoordZ::getCoordZAlongCartLine(), and woss::CoordZ::getCoordZFromCartesianCoords().

13.14.3.3 getY() `double woss::CoordZ::CartCoords::getY ( ) const [inline]`

Returns the number of meters along the Y axis

Returns

y axis value [m]

References y.

Referenced by woss::CoordZ::getCartDistance(), woss::CoordZ::getCartRelAzimuth(), woss::CoordZ::getCartY(), woss::CoordZ::getCoordZAlongCartLine(), and woss::CoordZ::getCoordZFromCartesianCoords().

13.14.3.4 getZ() `double woss::CoordZ::CartCoords::getZ ( ) const [inline]`

Returns the number of meters along the z axis

Returns

Z axis value [m]

References z.

Referenced by woss::CoordZ::getCartDistance(), woss::CoordZ::getCartRelZenith(), woss::CoordZ::getCartZ(), woss::CoordZ::getCoordZAlongCartLine(), and woss::CoordZ::getCoordZFromCartesianCoords().

13.14.3.5 operator<<() `friend::std::ostream & woss::CoordZ::CartCoords::operator<< ( ::std::ostream & os, const CartCoords & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const CartCoords reference

Returns

os reference after the operation

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

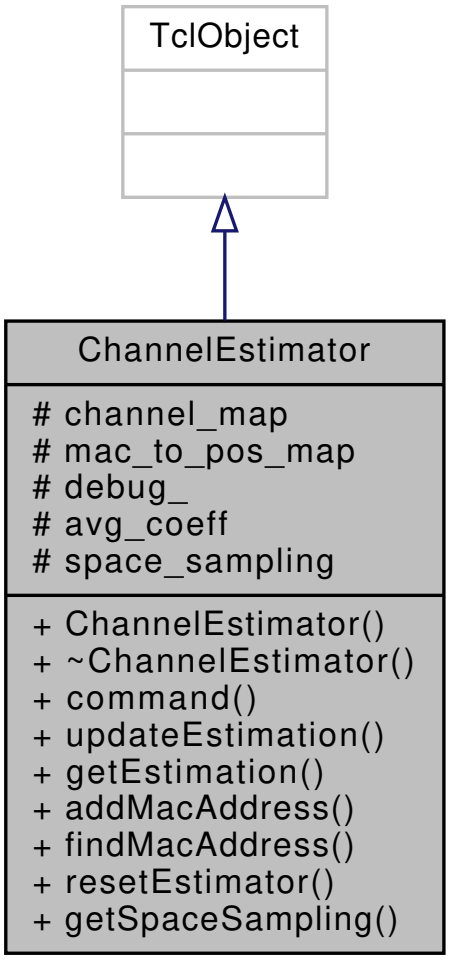
- woss/woss_def/coordinates-definitions.h
- woss/woss_def/coordinates-definitions.cpp

13.15 ChannelEstimator Class Reference

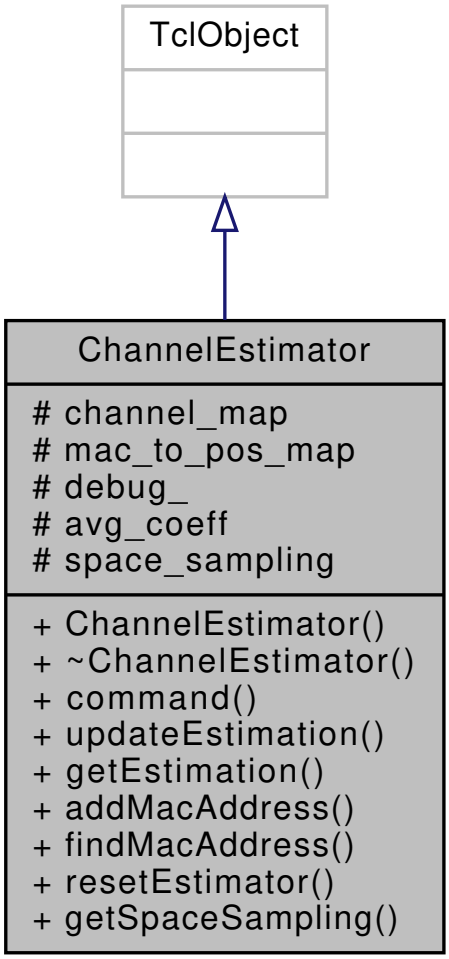
Class for channel estimation and averaging.

`#include <uw-woss-channel-estimator.h>`

Inheritance diagram for ChannelEstimator:



Collaboration diagram for ChannelEstimator:



Public Types

- typedef ::std::map< woss::CoordZ, woss::TimeArr *, woss::CoordComparator< ChannelEstimator, woss::CoordZ > > **RxMap**
- typedef RxMap::iterator **RxMIter**
- typedef RxMap::const_iterator **RxMCIter**
- typedef RxMap::reverse_iterator **RxMRIter**
- typedef ::std::map< woss::CoordZ, RxMap, woss::CoordComparator< ChannelEstimator, woss::CoordZ > > **ChannelMap**
- typedef ChannelMap::iterator **ChMapIter**
- typedef ChannelMap::reverse_iterator **ChMapRIter**
- typedef std::map< int, WossPosition * > **MacToPosMap**
- typedef MacToPosMap::iterator **MacMapIter**
- typedef MacToPosMap::reverse_iterator **MacMapRIter**

Public Member Functions

- virtual int **command** (int argc, const char *const *argv)
- virtual void updateEstimation (const woss::CoordZ &tx, const woss::CoordZ &rx, woss::TimeArr *curr_↩ channel)
- woss::TimeArr * getEstimation (const woss::CoordZ &tx, const woss::CoordZ &rx)
- void **addMacAddress** (int addr, WossPosition *pos)
- virtual WossPosition * **findMacAddress** (int addr)
- virtual bool **resetEstimator** ()

Static Public Member Functions

- static double **getSpaceSampling** ()

Protected Attributes

- ChannelMap **channel_map**
- MacToPosMap **mac_to_pos_map**
- double **debug_**
- double **avg_coeff**

Static Protected Attributes

- static double **space_sampling** = 0.0

13.15.1 Detailed Description

Class for channel estimation and averaging.

ChannelEstimator provides extensible estimation and averaging methods.

13.15.2 Member Function Documentation

13.15.2.1 getEstimation()

```
woss::TimeArr * ChannelEstimator::getEstimation (
    const woss::CoordZ & tx,
    const woss::CoordZ & rx )
```

Returns channel estimation for the given tx-rx couple

Parameters

tx	ns address type of transmitter node
rx	ns address type of receiver node

Returns

pointer to a heap-allocated woss::TimeArr channel

13.15.2.2 updateEstimation()

```
void ChannelEstimator::updateEstimation (
    const woss::CoordZ & tx,
    const woss::CoordZ & rx,
    woss::TimeArr * curr_channel )  [virtual]
```

Updates channel estimation for the given tx-rx couple

Parameters

tx	ns address type of transmitter node
rx	ns address type of receiver node
curr_channel	pointer to new channel value

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

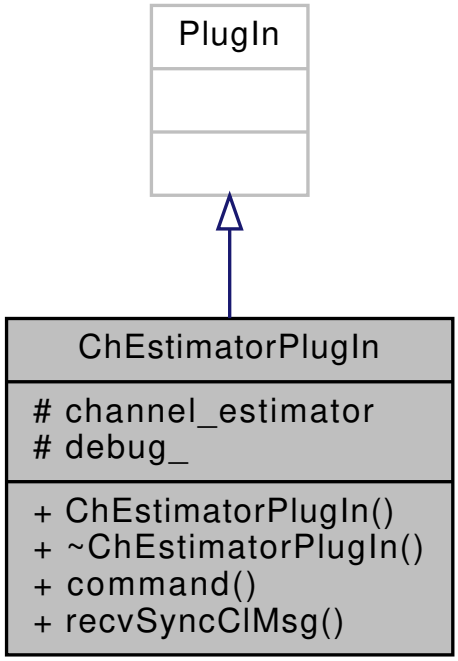
- woss_phy/uw-woss-channel-estimator.h
- woss_phy/uw-woss-channel-estimator.cpp

13.16 ChEstimatorPlugIn Class Reference

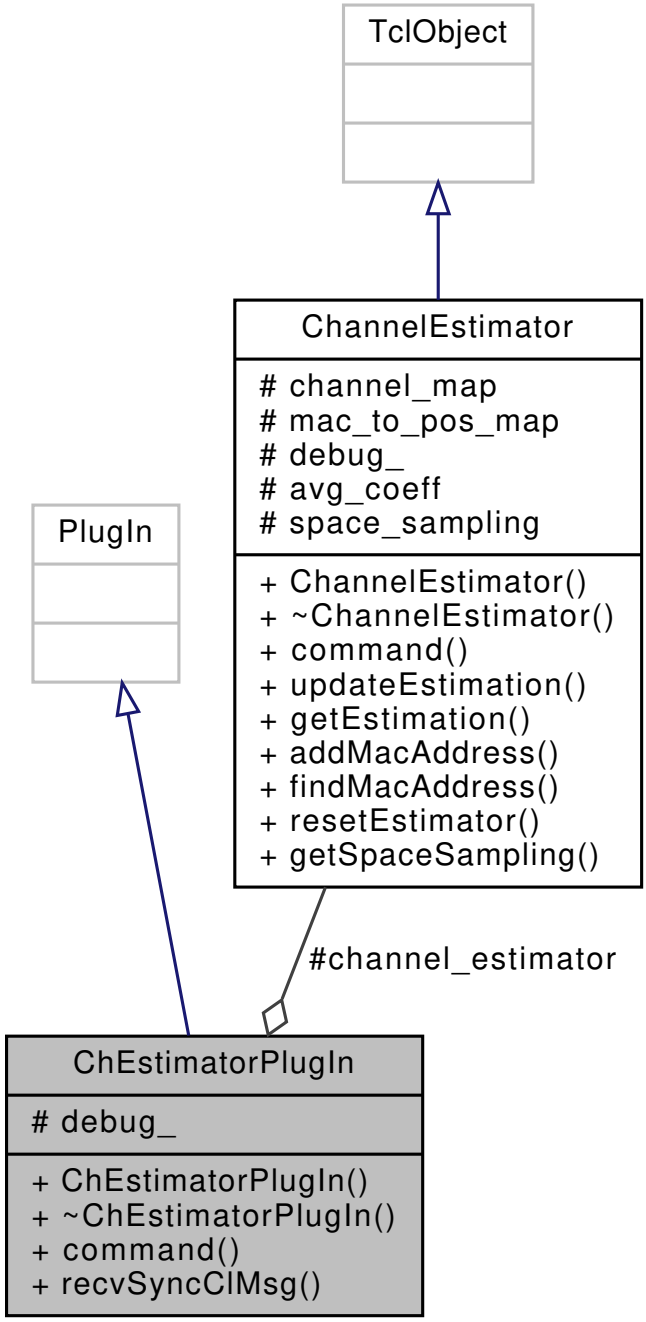
Service class for attaching a ChannelEstimator to the node bus.

`#include <uw-woss-channel-estimator.h>`

Inheritance diagram for ChEstimatorPlugIn:



Collaboration diagram for ChEstimatorPlugIn:



Public Member Functions

- virtual int **command** (int argc, const char ∗const ∗argv)
- virtual int **recvSyncClMsg** (ClMessage ∗m)

Protected Attributes

- ChannelEstimator ∗ **channel_estimator**
- double **debug_**

13.16.1 Detailed Description

Service class for attaching a ChannelEstimator to the node bus.

ChEstimatorPlugIn allows attaching of a ChannelEstimator instance to the node bus, permitting cross-layer communications

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

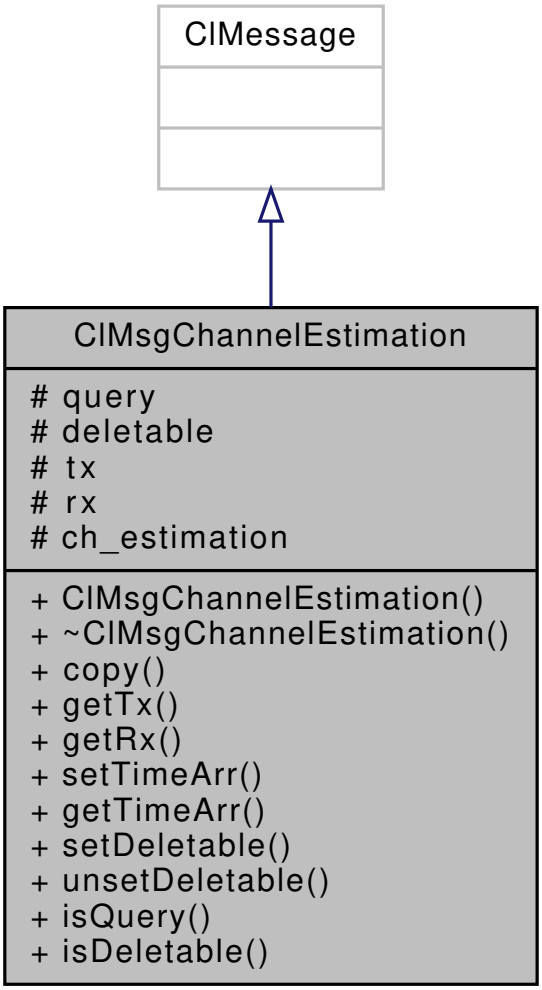
- woss_phy/uw-woss-channel-estimator.h
- woss_phy/uw-woss-channel-estimator.cpp

13.17 ClMsgChannelEstimation Class Reference

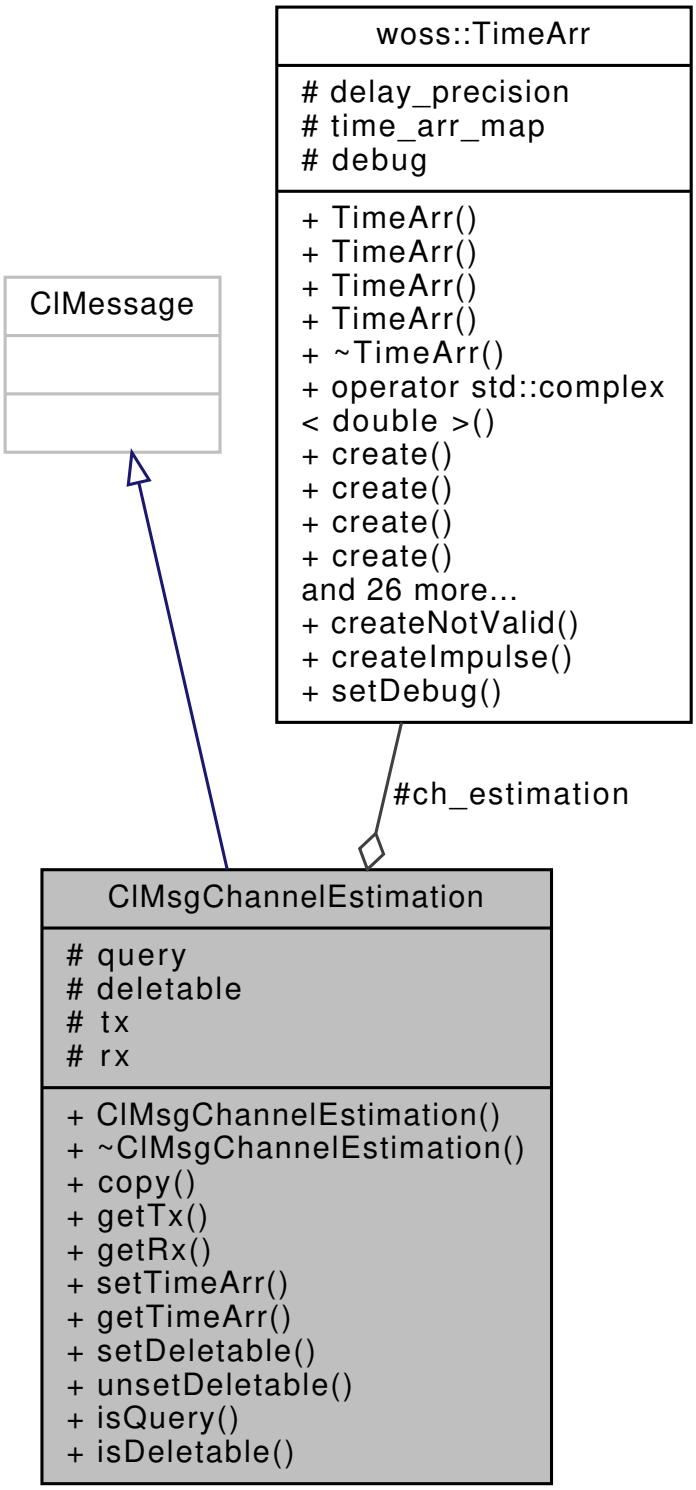
Class for channel estimation synchronous cross-layer messaging.

`#include <uw-woss-clmsg-channel-estimation.h>`

Inheritance diagram for ClMsgChannelEstimation:



Collaboration diagram for ClMsgChannelEstimation:



Public Member Functions

- **ClMsgChannelEstimation** (int i, int j, woss::TimeArr *time_arr=NULL)
- virtual ClMessage * **copy** ()
- const int **getTx** ()
- const int **getRx** ()

- void **setTimeArr** (woss::TimeArr *time_arr)
- woss::TimeArr * **getTimeArr** ()
- void **setDeletable** ()
- void **unsetDeletable** ()
- bool **isQuery** ()
- bool **isDeletable** ()

Protected Attributes

- bool **query**
- bool **deletable**
- int **tx**
- int **rx**
- woss::TimeArr * **ch_estimation**

13.17.1 Detailed Description

Class for channel estimation synchronous cross-layer messaging.

CIMsgChannelEstimation provides synchronous cross-layer communication for updating and requests of channel estimation

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

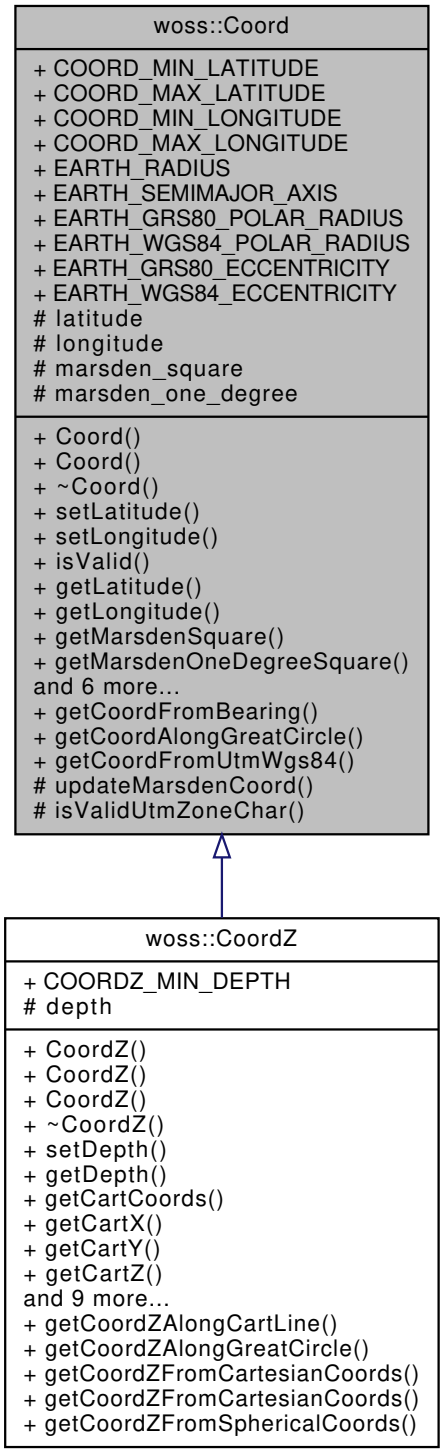
- woss_phy/uw-woss-clmsg-channel-estimation.h
- woss_phy/uw-woss-clmsg-channel-estimation.cpp

13.18 woss::Coord Class Reference

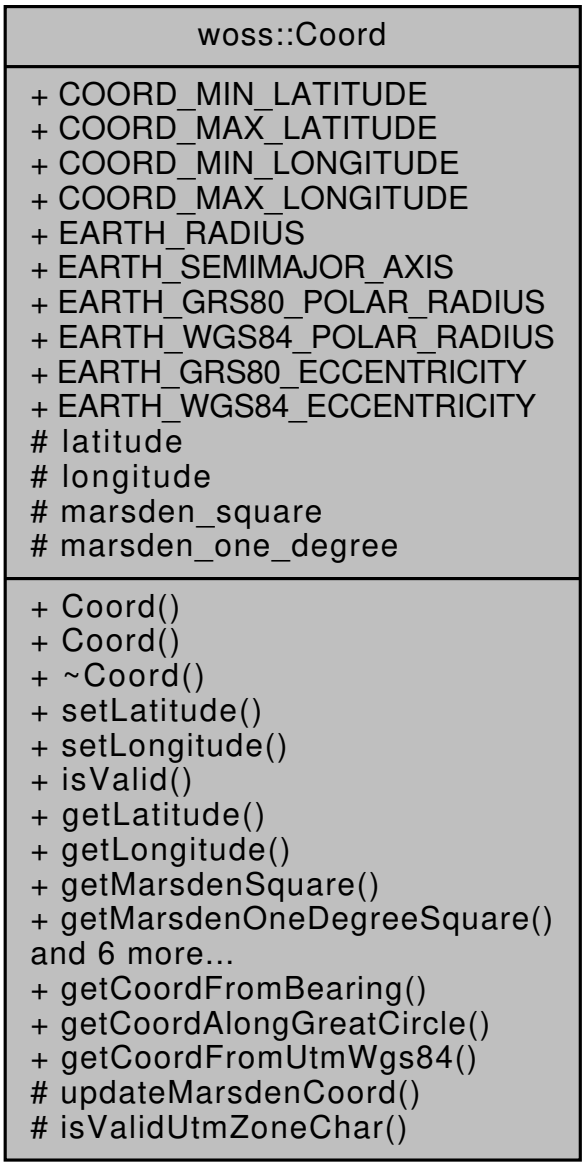
Coordinates (lat, long) class definitions and functions library.

```
#include <coordinates-definitions.h>
```

Inheritance diagram for woss::Coord:



Collaboration diagram for woss::Coord:



Public Member Functions

- Coord (double lat=COORD_NOT_SET_VALUE, double lon=COORD_NOT_SET_VALUE)
- Coord (const Coord &copy)
- void setLatitude (double lat)
- void setLongitude (double lon)
- virtual bool isValid () const
- double getLatitude () const
- double getLongitude () const
- int getMarsdenSquare () const
- int getMarsdenOneDegreeSquare () const
- MarsdenCoord getMarsdenCoord () const
- double getInitialBearing (const Coord &destination) const
- double getFinalBearing (const Coord &destination) const
- double getGreatCircleDistance (const Coord &destination, double depth=0) const
- Coord & operator= (const Coord &copy)
- friend::std::ostream & operator<< (::std::ostream &os, const Coord &instance)

Static Public Member Functions

- static const Coord getCoordFromBearing (const Coord &start_coord, double bearing, double distance, double depth=0.0)
- static const Coord getCoordAlongGreatCircle (const Coord &start_coord, const Coord &end_coord, double distance, double depth=0.0)
- static const Coord getCoordFromUtmWgs84 (double easting, double northing, double utm_zone_number, UtmZoneChar utm_zone_char)

Static Public Attributes

- static const double **COORD_MIN_LATITUDE** = -90.0

  Minimum valid Latitude.
- static const double **COORD_MAX_LATITUDE** = 90.0

  Maximum valid Latitude.
- static const double **COORD_MIN_LONGITUDE** = -180.0

  Minimum valid Longitude.
- static const double **COORD_MAX_LONGITUDE** = 180.0

  Maximum valid Longitude.
- static const double **EARTH_RADIUS** = 6371000.0

  Mean earth radius in meters.
- static const double **EARTH_SEMIMAJOR_AXIS** = 6378137.0

  Earth's semi-major axis in meters as defined by both GRS80 and WGS84.
- static const double **EARTH_GRS80_POLAR_RADIUS** = 6356752.3141

  Earth's semi-major axis in meters as defined by GRS80.
- static const double **EARTH_WGS84_POLAR_RADIUS** = 6356752.314245

  Earth's polar radius in meters as defined by WGS84.
- static const double **EARTH_GRS80_ECCENTRICITY** = 0.0818191910428158

  Earth's first eccentricity as defined by GRS80.
- static const double **EARTH_WGS84_ECCENTRICITY** = 0.0818191908426215

  Earth's first eccentricity as defined by WGS84.

Protected Member Functions

- void updateMarsdenCoord ()

Static Protected Member Functions

- static bool isValidUtmZoneChar (UtmZoneChar utm_zone_char)

Protected Attributes

- double latitude
- double longitude
- int marsden_square
- int marsden_one_degree

Friends

- const Coord operator+ (const Coord &left, const Coord &right)
- const Coord operator- (const Coord &left, const Coord &right)
- Coord & operator+= (Coord &left, const Coord &right)
- Coord & operator-= (Coord &left, const Coord &right)
- bool operator== (const Coord &left, const Coord &right)
- bool operator!= (const Coord &left, const Coord &right)
- bool operator> (const Coord &left, const Coord &right)
- bool operator< (const Coord &left, const Coord &right)
- bool operator>= (const Coord &left, const Coord &right)
- bool operator<= (const Coord &left, const Coord &right)

13.18.1 Detailed Description

Coordinates (lat, long) class definitions and functions library.

Coord class stores a double **decimal degree** latitude and double **decimal degree** longitude. It has a reach library for Marsden coordinates, distance, bearing and arithmetic calculations. This class has virtual methods for the sole purpose of CoordZ inheritance.

13.18.2 Constructor & Destructor Documentation

13.18.2.1 Coord() [1/2]

```
Coord::Coord (
        double lat = COORD_NOT_SET_VALUE,
        double lon = COORD_NOT_SET_VALUE )
```

Coord constructor.

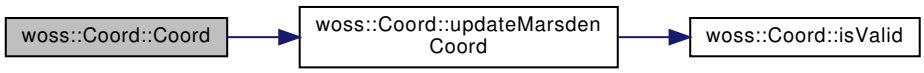
Parameters

lat	**decimal degree** latitude value. Default value makes the instance *not valid*
lon	**decimal degree** longitude value. Default value makes the instance *not valid*

References updateMarsdenCoord().

Referenced by getCoordFromBearing(), and getCoordFromUtmWgs84().

Here is the call graph for this function:



13.18.2.2 Coord() [2/2] Coord::Coord (const Coord & *copy*)

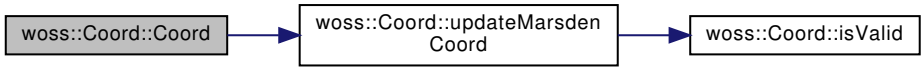
Coord copy constructor.

Parameters

copy	Coord to be copied

References latitude, longitude, and updateMarsdenCoord().

Here is the call graph for this function:



13.18.3 Member Function Documentation

13.18.3.1 getCoordAlongGreatCircle()

```
const Coord Coord::getCoordAlongGreatCircle (
    const Coord & start_coord,
    const Coord & end_coord,
    double distance,
    double depth = 0.0 )  [static]
```

Gets destination Cood given bearing and distance from a start Coord, travelling along a (shortest distance) great circle arc of given depth to end Coord

Parameters

start_coord	**valid** start Coord instance
end_coord	**valid** end Coord instance
distance	distance measured in *meters*
depth	depth measured in *meters*

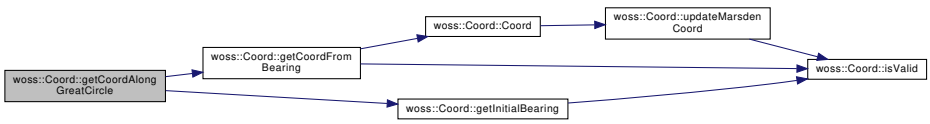
Returns

a new Coord instance containing the destination coordinates

References getCoordFromBearing(), and getInitialBearing().

Referenced by woss::CoordZ::getCoordZAlongGreatCircle().

Here is the call graph for this function:



13.18.3.2 getCoordFromBearing()

```
const Coord Coord::getCoordFromBearing (
    const Coord & start_coord,
    double bearing,
    double distance,
    double depth = 0.0 )  [static]
```

Gets destination Cood given bearing and distance from a start Coord, travelling along a (shortest distance) great circle arc of given depth

Parameters

start_coord	**valid** start Coord instance
bearing	bearing measured in *radians*
distance	distance measured in *meters*
depth	depth measured in *meters*

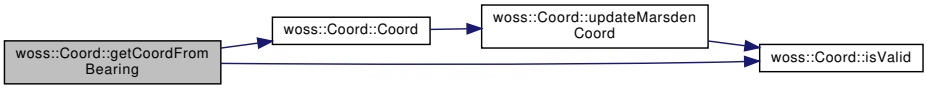
Returns

a new Coord instance containing the destination coordinates

References Coord(), EARTH_RADIUS, isValid(), latitude, and longitude.

Referenced by getCoordAlongGreatCircle(), and woss::ACToolboxWoss::initCoordZVector().

Here is the call graph for this function:



13.18.3.3 getCoordFromUtmWgs84()

```
const Coord Coord::getCoordFromUtmWgs84 (
    double easting,
    double northing,
    double utm_zone_number,
    UtmZoneChar utm_zone_char )  [static]
```

Gets destination Cood given easting and northing in UTM - WGS84 coordinates

Parameters

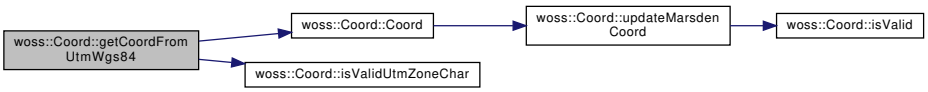
easting	**valid** easting relative to central UTM zone meridian
end_coord	**valid** northing
utm_zone_number	**valid**zone number
utm_zone_char	**valid**zone character

Returns

a new Coord instance containing the destination coordinates

References Coord(), EARTH_SEMIMAJOR_AXIS, EARTH_WGS84_POLAR_RADIUS, isValidUtmZoneChar(), latitude, and longitude.

Here is the call graph for this function:



13.18.3.4 getFinalBearing() double Coord::getFinalBearing (const Coord & *destination*) const

Gets the final bearing of a trajectory

Parameters

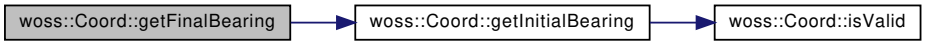
destination	destination Coord instance

Returns

initial bearing measured in *radians*

References getInitialBearing().

Here is the call graph for this function:



13.18.3.5 getGreatCircleDistance() double Coord::getGreatCircleDistance (const Coord & *destination*, double *depth* = *0*) const

Gets calculates great-circle distances between the two points – that is, the shortest distance over the earth's surface – using the 'Haversine' formula

Parameters

destination	**valid** destination Coord instance

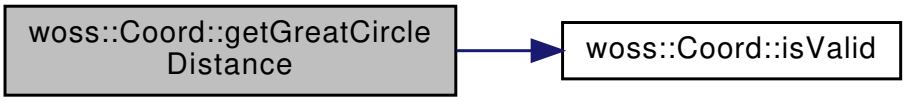
Returns

distance measured in *meters*

References EARTH_RADIUS, isValid(), latitude, and longitude.

Referenced by woss::CoordZ::getCoordZAlongGreatCircle(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::
woss::WossManagerResDbMT::getWossTimeArr(), woss::WossManager::getWossTimeArr(), woss::ACToolboxWoss::initAltimetry(),
woss::Woss::initialize(), and woss::WossDbManager::RangeOperator::operator()().

Here is the call graph for this function:



13.18.3.6 getInitialBearing() `double Coord::getInitialBearing ( const Coord & destination ) const`

Gets the initial bearing of a trajectory

Parameters

destination	destination Coord instance

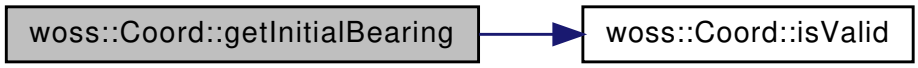
Returns

initial bearing measured in *radians*

References isValid(), latitude, and longitude.

Referenced by getCoordAlongGreatCircle(), getFinalBearing(), woss::Woss::initialize(), and woss::WossDbManager::BearingOperator

Here is the call graph for this function:



13.18.3.7 getLatitude() `double woss::Coord::getLatitude ( ) const [inline]`

Gets the latitude value

Returns

latitude value of the instance

References latitude.

Referenced by woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeOffsets(), woss::BathyGebcoDb::get1DBa
woss::BathyGebcoDb::get2DBathyIndexes(), woss::SSP::getDepthfromPressure(), woss::Location::getLatitude(),
woss::SSP::getPressureFromDepth(), woss::SedimDeck41CoordDb::getSeaFloorType(), woss::SedimDeck41CoordDb::getSedimInd
woss::SedimDeck41CoordDb::getSedimIndexes(), woss::SspWoa2005Db::getSSPIndexes(), woss::UtmWgs84::getUtmWgs84FromC
woss::BathyGebcoDb::getValue(), woss::SSP::isAntarcticOcean(), woss::SSP::isArcticOcean(), woss::SSP::isBalticSea(),
woss::SSP::isBlackSea(), woss::SSP::isCanonOcean(), woss::SSP::isCelebesSea(), woss::SSP::isHalmaheraSea(),
woss::SSP::isJapanSea(), woss::SSP::isMediterraneanSea(), woss::SSP::isNEAtlanticOcean(), woss::SSP::isRedSea(),
woss::SSP::isSuluSea(), and woss::Location::operator<<().

13.18.3.8 getLongitude() `double woss::Coord::getLongitude ( ) const [inline]`

Gets the longitude value

Returns

longitude value of the instance

References longitude.

Referenced by woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeOffsets(), woss::Location::getLongitude(), woss::SedimDeck41CoordDb::getSedimIndexes(), woss::SspWoa2005Db::getSSPIndexes(), woss::UtmWgs84::getUtmWgs84FromC woss::SSP::isBalticSea(), woss::SSP::isBlackSea(), woss::SSP::isCelebesSea(), woss::SSP::isHalmaheraSea(), woss::SSP::isJapanSea(), woss::SSP::isMediterraneanSea(), woss::SSP::isNEAtlanticOcean(), woss::SSP::isRedSea(), woss::SSP::isSuluSea(), and woss::Location::operator<<().

13.18.3.9 getMarsdenCoord() `MarsdenCoord woss::Coord::getMarsdenCoord ( ) const [inline]`

Gets the marsden coordinates

Returns

complete marsden coordinates of the instance

References marsden_one_degree, and marsden_square.

13.18.3.10 getMarsdenOneDegreeSquare() `int woss::Coord::getMarsdenOneDegreeSquare ( ) const [inline]`

Gets the marsden one degree square value

Returns

marsden one degree square value of the instance

References marsden_one_degree.

13.18.3.11 getMarsdenSquare() `int woss::Coord::getMarsdenSquare ( ) const [inline]`

Gets the marsden square value

Returns

marsden square value of the instance

References marsden_square.

13.18.3.12 isValid() `virtual bool woss::Coord::isValid ( ) const [inline], [virtual]`

Checks the validity of coordinates provided

Returns

true if coordinates are valid, *false* otherwise

Reimplemented in woss::CoordZ.

References COORD_MAX_LATITUDE, COORD_MAX_LONGITUDE, COORD_MIN_LATITUDE, COORD_MIN_LONGITUDE, latitude, and longitude.

Referenced by getCoordFromBearing(), getGreatCircleDistance(), getInitialBearing(), woss::SSP::insertValue(), woss::CoordZ::isValid(), woss::operator+=(), woss::operator-=(), woss::SSP::transform(), and updateMarsdenCoord().

13.18.3.13 isValidUtmZoneChar() `bool Coord::isValidUtmZoneChar ( UtmZoneChar utm_zone_char ) [static], [protected]`

Checks if the passed utm zone character is valid

Parameters

utm_zone_char	utm zone character

Returns

true if valid, *false* otherwise

Referenced by getCoordFromUtmWgs84().

13.18.3.14 operator<<() `friend::std::ostream & woss::Coord::operator<< ( ::std::ostream & os, const Coord & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Coord reference

Returns

os reference after the operation

13.18.3.15 operator=() `Coord & Coord::operator= ( const Coord & copy )`

Assignment operator

Parameters

copy	const reference to a Coord object to be copied

Returns

Coord reference to *this*

References latitude, longitude, marsden_one_degree, and marsden_square.

13.18.3.16 setLatitude() `void woss::Coord::setLatitude ( double lat ) [inline]`

Sets latitude and updates marsden coordinates

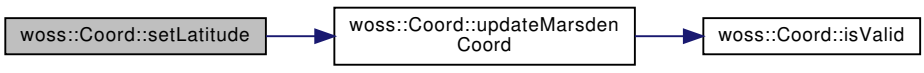
Parameters

lat	latitude value

References latitude, and updateMarsdenCoord().

Referenced by woss::Location::setLatitude().

Here is the call graph for this function:



13.18.3.17 setLongitude() `void woss::Coord::setLongitude ( double lon ) [inline]`

Sets longitude and updates marsden coordinates

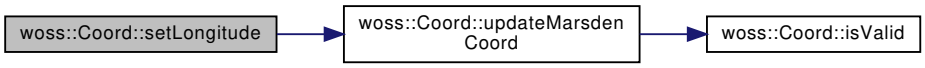
Parameters

lon	longitude value

References longitude, and updateMarsdenCoord().

Referenced by woss::Location::setLongitude().

Here is the call graph for this function:



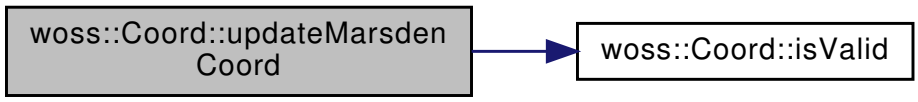
13.18.3.18 updateMarsdenCoord() `void Coord::updateMarsdenCoord ( ) [protected]`

Calculates marsden coordinates from latitude and longitude

References isValid(), latitude, longitude, marsden_one_degree, and marsden_square.

Referenced by Coord(), setLatitude(), and setLongitude().

Here is the call graph for this function:



13.18.4 Friends And Related Function Documentation

13.18.4.1 operator"!= `bool operator!= ( const Coord & left, const Coord & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.18.4.2 operator+ `const Coord operator+ ( const Coord & left, const Coord & right ) [friend]`

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.18.4.3 operator+= `Coord & operator+= ( Coord & left, const Coord & right ) [friend]`

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.18.4.4 operator- `const Coord operator- ( const Coord & left, const Coord & right ) [friend]`

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.18.4.5 operator-= `Coord & operator-= ( Coord & left, const Coord & right ) [friend]`

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.18.4.6 operator< `bool operator< ( const Coord & left, const Coord & right ) [friend]`

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

13.18.4.7 operator<= `bool operator<= ( const Coord & left, const Coord & right ) [friend]`

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

13.18.4.8 operator== `bool operator== ( const Coord & left, const Coord & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.18.4.9 operator> `bool operator> ( const Coord & left, const Coord & right ) [friend]`

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

13.18.4.10 operator>= `bool operator>= ( const Coord & left, const Coord & right ) [friend]`

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

13.18.5 Member Data Documentation

13.18.5.1 latitude `double woss::Coord::latitude [protected]`

Latitude value

Referenced by Coord(), woss::CoordZ::getCartCoords(), getCoordFromBearing(), getCoordFromUtmWgs84(), woss::CoordZ::getCoordZFromCartesianCoords(), getGreatCircleDistance(), getInitialBearing(), getLatitude(), woss::CoordZ::getSphericalTheta(), isValid(), woss::operator+=(), woss::operator-=(), operator=(), woss::CoordZ::operator=(), setLatitude(), and updateMarsdenCoord().

13.18.5.2 longitude `double woss::Coord::longitude [protected]`

Longitude value

Referenced by Coord(), woss::CoordZ::getCartCoords(), getCoordFromBearing(), getCoordFromUtmWgs84(), woss::CoordZ::getCoordZFromCartesianCoords(), getGreatCircleDistance(), getInitialBearing(), getLongitude(), woss::CoordZ::getSphericalPhi(), isValid(), woss::operator+=(), woss::operator-=(), operator=(), woss::CoordZ::operator=(), setLongitude(), and updateMarsdenCoord().

13.18.5.3 marsden_one_degree `int woss::Coord::marsden_one_degree [protected]`

Marsden one degree square value

Referenced by getMarsdenCoord(), getMarsdenOneDegreeSquare(), operator=(), woss::CoordZ::operator=(), and updateMarsdenCoord().

13.18.5.4 marsden_square `int woss::Coord::marsden_square [protected]`

Marsden square value

Referenced by getMarsdenCoord(), getMarsdenSquare(), operator=(), woss::CoordZ::operator=(), and updateMarsdenCoord().

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

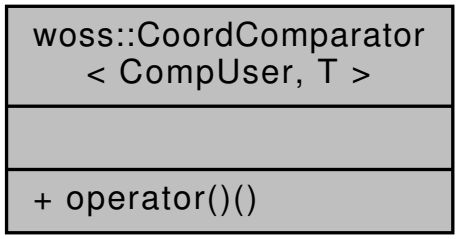
- woss/woss_def/coordinates-definitions.h
- woss/woss_def/coordinates-definitions.cpp

13.19 woss::CoordComparator< CompUser, T > Class Template Reference

Function object for partial ordering of coordinates.

`#include <coordinates-definitions.h>`

Collaboration diagram for woss::CoordComparator< CompUser, T >:



Public Member Functions

- bool operator() (const T &x, const T &y) const

13.19.1 Detailed Description

template<class CompUser, class T = Coord>
class woss::CoordComparator< CompUser, T >

Function object for partial ordering of coordinates.

Function object class for partial ordering of two coordinates instances based on great circle distance. The user class has to provide a *static* method called **getSpaceSampling()** that returns the threshold distance

13.19.2 Member Function Documentation

13.19.2.1 operator()() `template<class CompUser , class T = Coord>`

```
bool woss::CoordComparator< CompUser, T >::operator() (
            const T & x,
            const T & y ) const  [inline]
```

Function that compares to T instances. If CompUser::getSpaceSampling() is valid (>=0) two valid T are considered equivalent if their great circle distance is less or equal to the space sampling value

Parameters

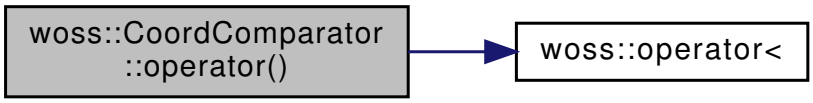
tx	const reference to a valid T object
rx	const reference to a valid T object

Returns

true if x less than y, *false* otherwise

References woss::operator<().

Here is the call graph for this function:



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

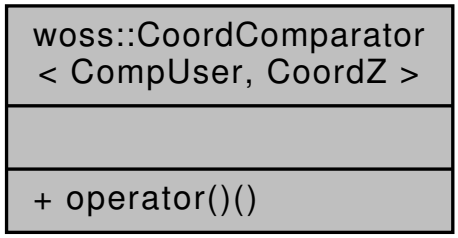
- woss/woss_def/coordinates-definitions.h

13.20 woss::CoordComparator< CompUser, CoordZ > Class Template Reference

Partial specialization for partial ordering of CoordZ coordinates.

`#include <coordinates-definitions.h>`

Collaboration diagram for woss::CoordComparator< CompUser, CoordZ >:



Public Member Functions

- bool operator() (const CoordZ &x, const CoordZ &y) const

13.20.1 Detailed Description

template<class CompUser>
class woss::CoordComparator< CompUser, CoordZ >

Partial specialization for partial ordering of CoordZ coordinates.

Function object class for partial ordering of two CoordZ instances based on cartesian distance. The user class has to provide a *static* method called **getSpaceSampling()** that returns the threshold distance

13.20.2 Member Function Documentation

13.20.2.1 operator()() template<class CompUser >

```
bool woss::CoordComparator< CompUser, CoordZ >::operator() (
            const CoordZ & x,
            const CoordZ & y ) const  [inline]
```

Function that compares to woss::CoordZ instances. If CompUser::getSpaceSampling() is valid (>=0) two valid CoordZ are considered equivalent if their cartesian distance is less or equal to the space sampling value

Parameters

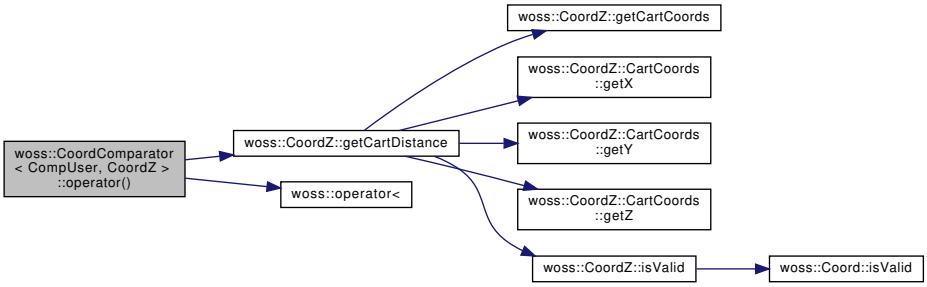
tx	const reference to a valid CoordZ object
rx	const reference to a valid CoordZ object

Returns

true if x less than y, *false* otherwise

References woss::CoordZ::getCartDistance(), and woss::operator<().

Here is the call graph for this function:



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

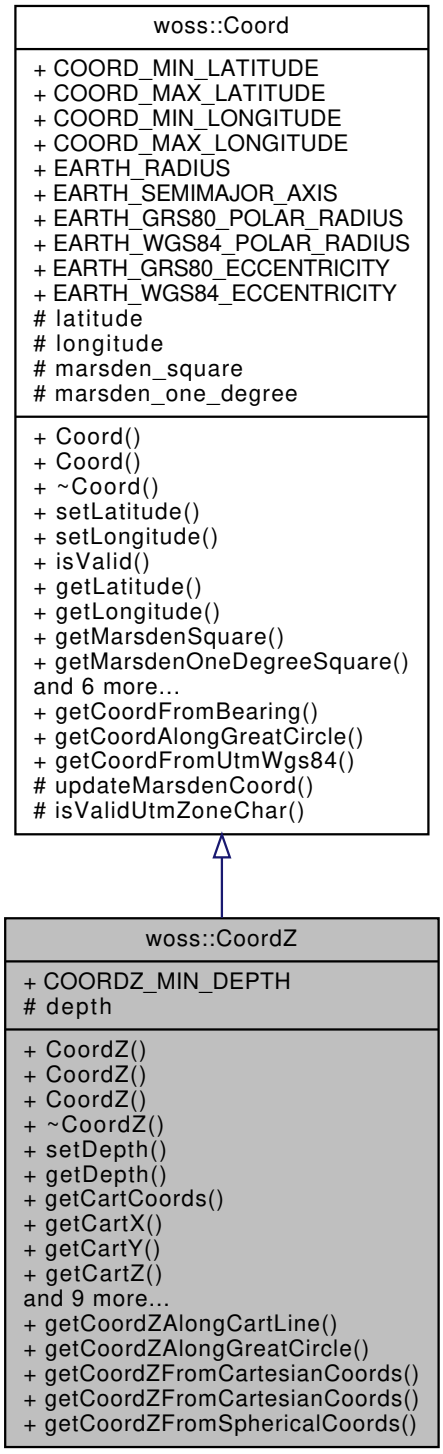
- woss/woss_def/coordinates-definitions.h

13.21 woss::CoordZ Class Reference

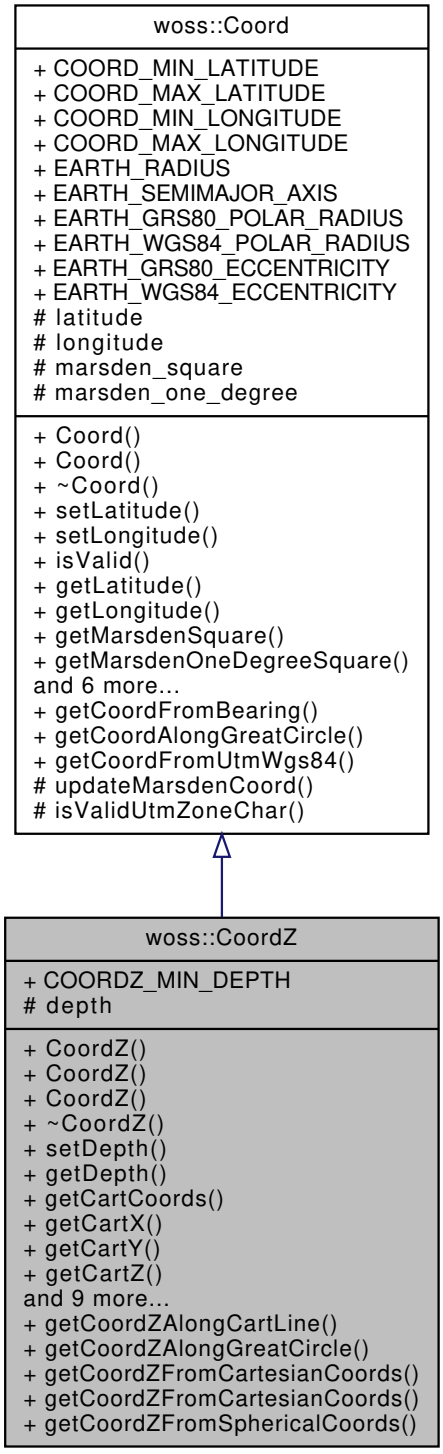
3D-Coordinates (lat, long, depth) class definitions and functions library

`#include <coordinates-definitions.h>`

Inheritance diagram for woss::CoordZ:



Collaboration diagram for woss::CoordZ:



Classes

- class CartCoords

  Class that represents cartesian coordinates.

Public Types

- enum CoordZSpheroidType { **COORDZ_SPHERE** = 0 , **COORDZ_GRS80** , **COORDZ_WGS84** }

Spheroid model to use.

Public Member Functions

- CoordZ (double lat=COORD_NOT_SET_VALUE, double lon=COORD_NOT_SET_VALUE, double z=COORD_NOT_SET_VALUE)
- CoordZ (const Coord &coords, double depth=COORD_NOT_SET_VALUE)
- CoordZ (const CoordZ &copy)
- void setDepth (double d)
- double getDepth () const
- CartCoords getCartCoords (CoordZSpheroidType type=COORDZ_SPHERE) const
- double getCartX (CoordZSpheroidType type=COORDZ_SPHERE) const
- double getCartY (CoordZSpheroidType type=COORDZ_SPHERE) const
- double getCartZ (CoordZSpheroidType type=COORDZ_SPHERE) const
- double getSphericalRho () const
- double getSphericalTheta () const
- double getSphericalPhi () const
- double getCartDistance (const CoordZ &coords, CoordZSpheroidType type=COORDZ_SPHERE) const
- double getCartRelZenith (const CoordZ &coords) const
- double getCartRelAzimuth (const CoordZ &coords) const
- virtual bool isValid () const
- CoordZ & operator= (const CoordZ &coordz)
- friend::std::ostream & operator<< (::std::ostream &os, const CoordZ &instance)

Static Public Member Functions

- static const CoordZ getCoordZAlongCartLine (const CoordZ &start, const CoordZ &end, double distance)
- static const CoordZ getCoordZAlongGreatCircle (const CoordZ &start, const CoordZ &end, double distance)
- static const CoordZ getCoordZFromCartesianCoords (double x, double y, double z, CoordZSpheroidType type=COORDZ_SPHERE)
- static const CoordZ getCoordZFromCartesianCoords (const CartCoords &cart_coords)
- static const CoordZ getCoordZFromSphericalCoords (double rho, double theta, double phi)

Static Public Attributes

- static const double **COORDZ_MIN_DEPTH** = 0.0

  Minimum valid depth.

Protected Attributes

- double depth

Friends

- const CoordZ operator+ (const CoordZ &left, const CoordZ &right)
- const CoordZ operator- (const CoordZ &left, const CoordZ &right)
- CoordZ & operator+= (CoordZ &left, const CoordZ &right)
- CoordZ & operator-= (CoordZ &left, const CoordZ &right)
- bool operator== (const CoordZ &left, const CoordZ &right)
- bool operator!= (const CoordZ &left, const CoordZ &right)
- bool operator> (const CoordZ &left, const CoordZ &right)
- bool operator< (const CoordZ &left, const CoordZ &right)
- bool operator>= (const CoordZ &left, const CoordZ &right)
- bool operator<= (const CoordZ &left, const CoordZ &right)

Additional Inherited Members

13.21.1 Detailed Description

3D-Coordinates (lat, long, depth) class definitions and functions library

CoordZ class inherit from Coord, adding a depth value

13.21.2 Member Enumeration Documentation

13.21.2.1 CoordZSpheroidType `enum woss::CoordZ::CoordZSpheroidType`

Spheroid model to use.

Spheroid model to use: perfect sphere (COORDZ_SPHERE), Geodetic Reference System 1980 (COORDZ_GRS80), or World Geodetic System 1984 (COORDZ_WGS84)

13.21.3 Constructor & Destructor Documentation

13.21.3.1 CoordZ() [1/3]

```
CoordZ::CoordZ (
        double lat = COORD_NOT_SET_VALUE,
        double lon = COORD_NOT_SET_VALUE,
        double z = COORD_NOT_SET_VALUE )
```

CoordZ constructor.

Parameters

lat	**decimal degree** latitude value. Default value makes the instance *not valid*
lon	**decimal degree** longitude value. Default value makes the instance *not valid*
z	depth value in meters. Default value makes the instance *not valid*

Referenced by getCoordZAlongCartLine(), getCoordZAlongGreatCircle(), getCoordZFromCartesianCoords(), and getCoordZFromSphericalCoords().

13.21.3.2 CoordZ() [2/3]

```
CoordZ::CoordZ (
        const Coord & coords,
        double depth = COORD_NOT_SET_VALUE )  [explicit]
```

Explicit CoordZ constructor. No implicit cast from Coord to CoordZ is allowed.

Parameters

coords	a Coord reference
depth	depth value in meters. Default value makes the instance *not valid*

13.21.3.3 CoordZ() [3/3] CoordZ::CoordZ (const CoordZ & *copy*)

CoordZ copy constructor.

Parameters

copy	const reference to a CoordZ to be copied

13.21.4 Member Function Documentation

13.21.4.1 getCartCoords() CoordZ::CartCoords CoordZ::getCartCoords (CoordZSpheroidType *type* = *COORDZ_SPHERE*) const

Gets cartesian x coordinate

Parameters

type	Earh Model type

Returns

x in *meters*

References depth, woss::Coord::EARTH_GRS80_ECCENTRICITY, woss::Coord::EARTH_RADIUS, woss::Coord::EARTH_SEMIMA
woss::Coord::EARTH_WGS84_ECCENTRICITY, woss::Coord::latitude, and woss::Coord::longitude.

Referenced by getCartDistance(), getCartRelAzimuth(), getCartRelZenith(), getCartX(), getCartY(), getCartZ(), and getCoordZAlongCartLine().

13.21.4.2 getCartDistance() double CoordZ::getCartDistance (const CoordZ & *coords*, CoordZSpheroidType *type* = *COORDZ_SPHERE*) const

Gets the distance from cartesian coordinates approximations

Parameters

coords	a const reference to a valid CoordZ object

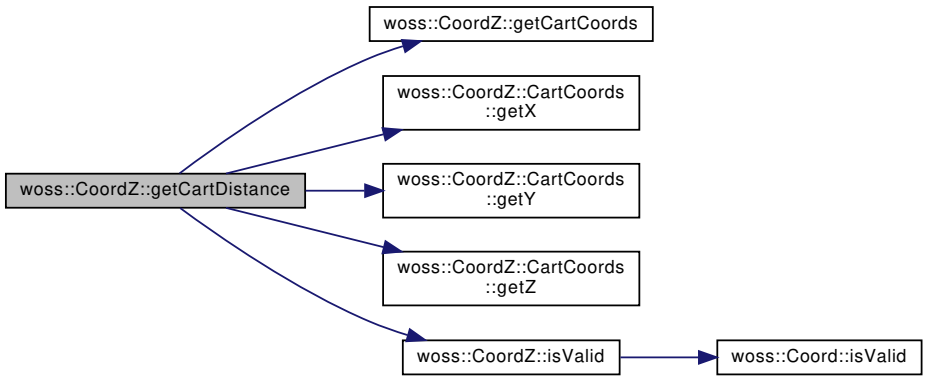
Returns

distance in *meters*

References getCartCoords(), woss::CoordZ::CartCoords::getX(), woss::CoordZ::CartCoords::getY(), woss::CoordZ::CartCoords::getZ and isValid().

Referenced by WossMPropagation::computeGain(), getCartRelZenith(), WossMPropagation::getGain(), woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), woss::WossManager::getWossTimeArr(), woss::Woss::initialize(), woss::Location::isEquivalentTo(), WossWpPosition::isEquivalentTo(), and woss::CoordComparator< CompUser, CoordZ >::operator()

Here is the call graph for this function:



13.21.4.3 getCartRelAzimuth() `double CoordZ::getCartRelAzimuth ( const CoordZ & coords ) const`

Gets relative azimuth from cartesian coordinates approximations, assuming earth as a sphere of ray = 6371000.0 meters

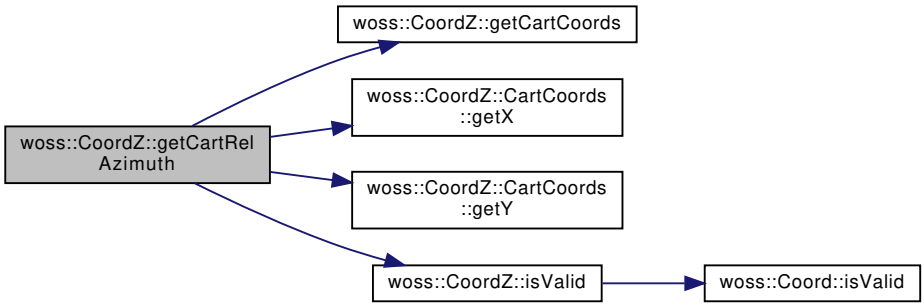
Returns

azimuth in *radians*

References getCartCoords(), woss::CoordZ::CartCoords::getX(), woss::CoordZ::CartCoords::getY(), and isValid().

Referenced by getCoordZAlongCartLine().

Here is the call graph for this function:



13.21.4.4 getCartRelZenith() `double CoordZ::getCartRelZenith ( const CoordZ & coords ) const`

Gets relative zenith from cartesian coordinates approximations, assuming earth as a sphere of ray = 6371000.0 meters

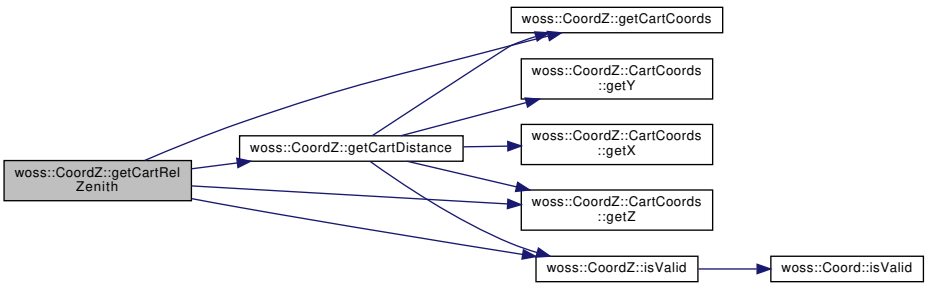
Returns

zenith in *radians*

References getCartCoords(), getCartDistance(), woss::CoordZ::CartCoords::getZ(), and isValid().

Referenced by getCoordZAlongCartLine().

Here is the call graph for this function:



13.21.4.5 getCartX() `double CoordZ::getCartX ( CoordZSpheroidType type = COORDZ_SPHERE ) const`

Gets cartesian x coordinate

Parameters

type	Earh Model type

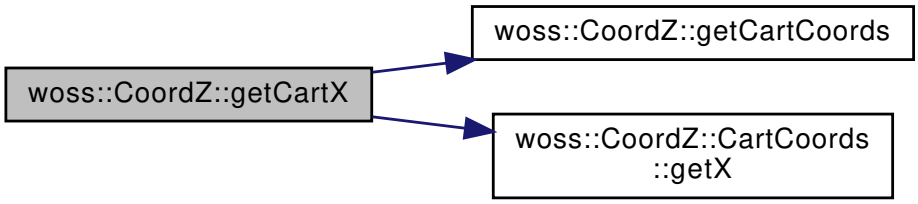
Returns

x in *meters*

References getCartCoords(), and woss::CoordZ::CartCoords::getX().

Referenced by woss::Location::getX(), and woss::Transducer::writeVertBeamPattern().

Here is the call graph for this function:



13.21.4.6 getCartY()

```
double CoordZ::getCartY (
        CoordZSpheroidType type = COORDZ_SPHERE ) const
```

Gets cartesian y coordinate

Parameters

type	Earh Model type

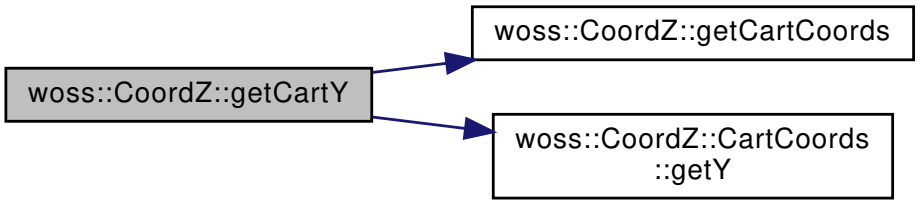
Returns

y in *meters*

References getCartCoords(), and woss::CoordZ::CartCoords::getY().

Referenced by woss::Location::getY(), and woss::Transducer::writeVertBeamPattern().

Here is the call graph for this function:



13.21.4.7 getCartZ()

```
double CoordZ::getCartZ (
        CoordZSpheroidType type = COORDZ_SPHERE ) const
```

Gets cartesian z coordinate

Parameters

type	Earh Model type

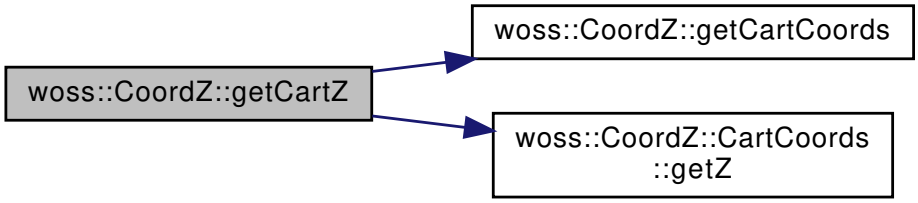
Returns

z in *meters*

References getCartCoords(), and woss::CoordZ::CartCoords::getZ().

Referenced by woss::Location::getZ(), and woss::Transducer::writeVertBeamPattern().

Here is the call graph for this function:



13.21.4.8 getCoordZAlongCartLine()

```
const CoordZ CoordZ::getCoordZAlongCartLine (
    const CoordZ & start,
    const CoordZ & end,
    double distance )  [static]
```

Gets The CoordZ at given distance along the line in cartesian coordinates, assuming earth as a sphere of ray = 6371000.0 meters that ties *start* and *end* CoordZ

Parameters

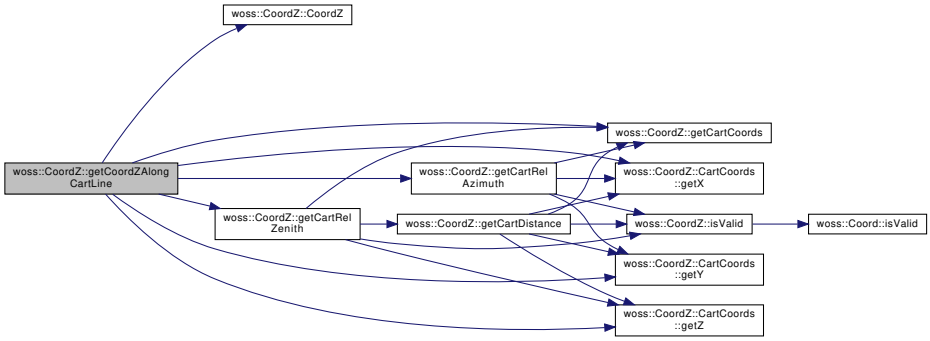
start	a const reference to a valid CoordZ object
end	a const reference to a valid CoordZ object
distance	travel distance in meters

Returns

a valid CoordZ object

References CoordZ(), depth, woss::Coord::EARTH_RADIUS, getCartCoords(), getCartRelAzimuth(), getCartRelZenith(), woss::CoordZ::CartCoords::getX(), woss::CoordZ::CartCoords::getY(), and woss::CoordZ::CartCoords::getZ().

Here is the call graph for this function:



13.21.4.9 getCoordZAlongGreatCircle()

```
const CoordZ CoordZ::getCoordZAlongGreatCircle (
    const CoordZ & start,
    const CoordZ & end,
    double distance )  [static]
```

Gets The CoordZ at given distance along the great circle at starting CoordZ depth. The output depth is calculated uniformly between start and end depth that ties *start* and *end* CoordZ

Parameters

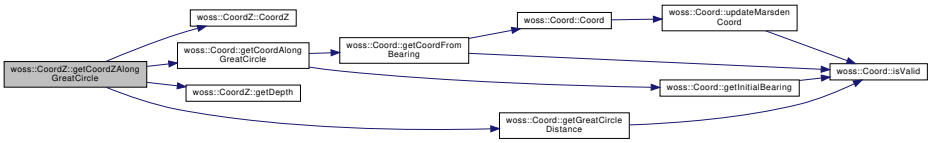
start	a const reference to a valid CoordZ object
end	a const reference to a valid CoordZ object
distance	travel distance in meters

Returns

a valid CoordZ object

References CoordZ(), woss::Coord::getCoordAlongGreatCircle(), getDepth(), and woss::Coord::getGreatCircleDistance().

Here is the call graph for this function:



13.21.4.10 getCoordZFromCartesianCoords() [1/2]

```
const CoordZ CoordZ::getCoordZFromCartesian↩
Coords (
    const CartCoords & cart_coords )  [static]
```

Gets the CoordZ converted from given cartesian coordinates

Parameters

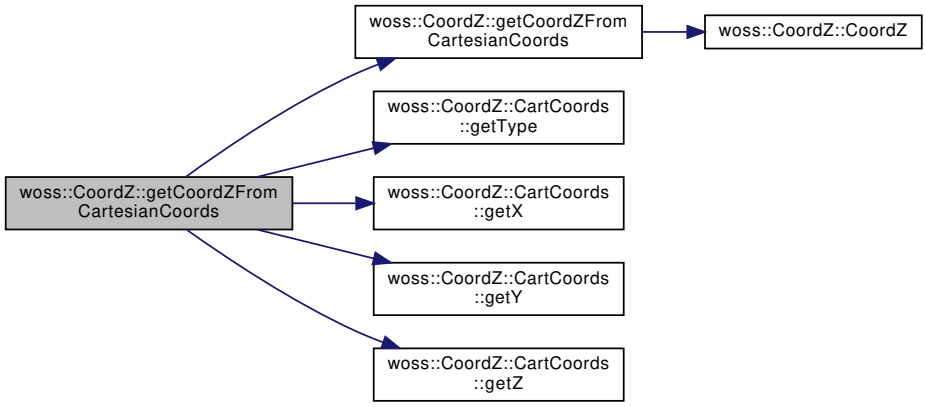
cart_coords	const reference to a valid CoordZ::CartCoords object

Returns

a valid CoordZ object

References getCoordZFromCartesianCoords(), woss::CoordZ::CartCoords::getType(), woss::CoordZ::CartCoords::getX(), woss::CoordZ::CartCoords::getY(), and woss::CoordZ::CartCoords::getZ().

Here is the call graph for this function:



13.21.4.11 getCoordZFromCartesianCoords() [2/2]

```
const CoordZ CoordZ::getCoordZFromCartesian↩
Coords (
            double x,
            double y,
            double z,
            CoordZSpheroidType type = COORDZ_SPHERE )  [static]
```

Gets the CoordZ converted from given cartesian coordinates

Parameters

x	x axis
y	y axis
z	z axis
type	Earth model type

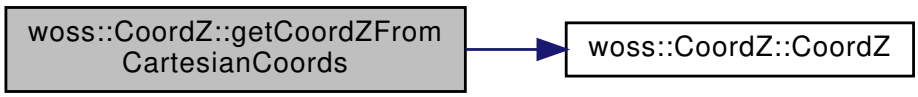
Returns

a valid CoordZ object

References CoordZ(), woss::Coord::EARTH_GRS80_POLAR_RADIUS, woss::Coord::EARTH_RADIUS, woss::Coord::EARTH_SEM
woss::Coord::EARTH_WGS84_POLAR_RADIUS, woss::Coord::latitude, and woss::Coord::longitude.

Referenced by getCoordZFromCartesianCoords().

Here is the call graph for this function:



13.21.4.12 getCoordZFromSphericalCoords()

```
const CoordZ CoordZ::getCoordZFromSphericalCoords (
    double rho,
    double theta,
    double phi )  [static]
```

Gets the CoordZ converted from given spherical coordinates

Parameters

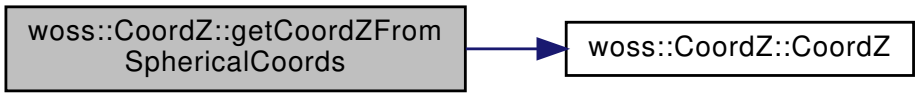
rho	radius
theta	theta angle
phi	phi angle

Returns

a valid CoordZ object

References CoordZ(), and woss::Coord::EARTH_RADIUS.

Here is the call graph for this function:



13.21.4.13 getDepth()

```
double woss::CoordZ::getDepth ( ) const  [inline]
```

Gets depth

Returns

depth in *meters*

References depth.

Referenced by woss::BellhopWoss::checkBoundaries(), woss::BellhopWoss::checkDepthOffsets(), getCoordZAlongGreatCircle(), woss::Location::getDepth(), woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWoss woss::WossManager::getWossTimeArr(), woss::Location::operator<<(), woss::BellhopWoss::writeReceiver(), and woss::BellhopWoss::writeTransmitter().

13.21.4.14 getSphericalPhi() `double CoordZ::getSphericalPhi ( ) const`

Gets spherical phi coordinate, assuming earth as a sphere of ray = 6371000.0 meters

Returns

z in *meters*

References woss::Coord::longitude.

13.21.4.15 getSphericalRho() `double CoordZ::getSphericalRho ( ) const`

Gets spherical rho coordinate, assuming earth as a sphere of ray = 6371000.0 meters

Returns

x in *meters*

References depth, and woss::Coord::EARTH_RADIUS.

13.21.4.16 getSphericalTheta() `double CoordZ::getSphericalTheta ( ) const`

Gets spherical theta coordinate, assuming earth as a sphere of ray = 6371000.0 meters

Returns

y in *meters*

References woss::Coord::latitude.

13.21.4.17 isValid() `virtual bool woss::CoordZ::isValid ( ) const [inline], [virtual]`

Checks the validity of coordinates and depth provided

Returns

true if coordinates and depth are valid, *false* otherwise

Reimplemented from woss::Coord.

References COORDZ_MIN_DEPTH, depth, and woss::Coord::isValid().

Referenced by getCartDistance(), getCartRelAzimuth(), getCartRelZenith(), woss::ACToolboxWoss::initCoordZVector(), woss::Location::isEquivalentTo(), woss::ACToolboxWoss::isValid(), woss::Location::isValid(), woss::operator+=(), woss::operator-=(), and woss::Woss::Woss().

Here is the call graph for this function:

13.21.4.18 operator<<() `friend::std::ostream & woss::CoordZ::operator<< ( ::std::ostream & os, const CoordZ & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const CoordZ reference

Returns

os reference after the operation

13.21.4.19 operator=() `CoordZ & CoordZ::operator= ( const CoordZ & coordz )`

Assignment operator

Parameters

copy	const reference to a CoordZ object to be copied

Returns

CoordZ reference to *this*

References depth, woss::Coord::latitude, woss::Coord::longitude, woss::Coord::marsden_one_degree, and woss::Coord::marsden_square.

13.21.4.20 setDepth() `void woss::CoordZ::setDepth ( double d ) [inline]`

Sets depth

Parameters

d	depth in *meters*

References depth.

Referenced by woss::ACToolboxWoss::initCoordZVector(), and woss::Location::setDepth().

13.21.5 Friends And Related Function Documentation

13.21.5.1 operator"!= `bool operator!= ( const CoordZ & left, const CoordZ & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.21.5.2 operator+ `const CoordZ operator+ ( const CoordZ & left, const CoordZ & right ) [friend]`

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.21.5.3 operator+= `CoordZ & operator+= ( CoordZ & left, const CoordZ & right ) [friend]`

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.21.5.4 operator- const CoordZ operator- (const CoordZ & *left*, const CoordZ & *right*) [friend]

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.21.5.5 operator-= CoordZ & operator-= (CoordZ & *left*, const CoordZ & *right*) [friend]

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.21.5.6 operator< bool operator< (const CoordZ & *left*, const CoordZ & *right*) [friend]

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

13.21.5.7 operator<= `bool operator<= ( const CoordZ & left, const CoordZ & right ) [friend]`

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

13.21.5.8 operator== `bool operator== ( const CoordZ & left, const CoordZ & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

13.21.5.9 operator> `bool operator> ( const CoordZ & left, const CoordZ & right ) [friend]`

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

13.21.5.10 operator>= `bool operator>= ( const CoordZ & left, const CoordZ & right ) [friend]`

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

13.21.6 Member Data Documentation

13.21.6.1 depth `double woss::CoordZ::depth [protected]`

Depth value

Referenced by getCartCoords(), getCoordZAlongCartLine(), getDepth(), getSphericalRho(), isValid(), woss::operator+=(), woss::operator-=(), operator=(), and setDepth().

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

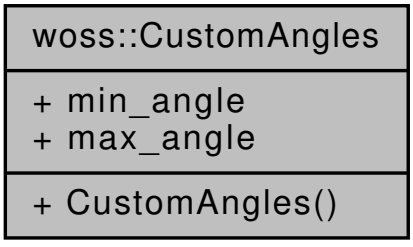
- woss/woss_def/coordinates-definitions.h
- woss/woss_def/coordinates-definitions.cpp

13.22 woss::CustomAngles Struct Reference

Bellhop min max angles.

`#include <bellhop-creator.h>`

Collaboration diagram for woss::CustomAngles:



Public Member Functions

- CustomAngles (double min=double(), double max=double())

Public Attributes

- double min_angle
- double max_angle

Friends

- std::ostream & **operator**<< (std::ostream &os, const CustomAngles &instance)

13.22.1 Detailed Description

Bellhop min max angles.

Struct that stores Bellhop minimum and maximum ray launching angle

13.22.2 Constructor & Destructor Documentation

13.22.2.1 CustomAngles()

```
woss::CustomAngles::CustomAngles (
    double min = double(),
    double max = double() )  [inline]
```

Default constructor

Parameters

min	minimum angle [signed degrees]
max	maximum angle [signed degrees]

13.22.3 Member Data Documentation

13.22.3.1 max_angle

```
double woss::CustomAngles::max_angle
```

maximum angle [signed degrees]

Referenced by woss::BellhopCreator::initializeBhWoss().

13.22.3.2 min_angle `double woss::CustomAngles::min_angle`

minimum angle [signed degrees]

Referenced by woss::BellhopCreator::initializeBhWoss().

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

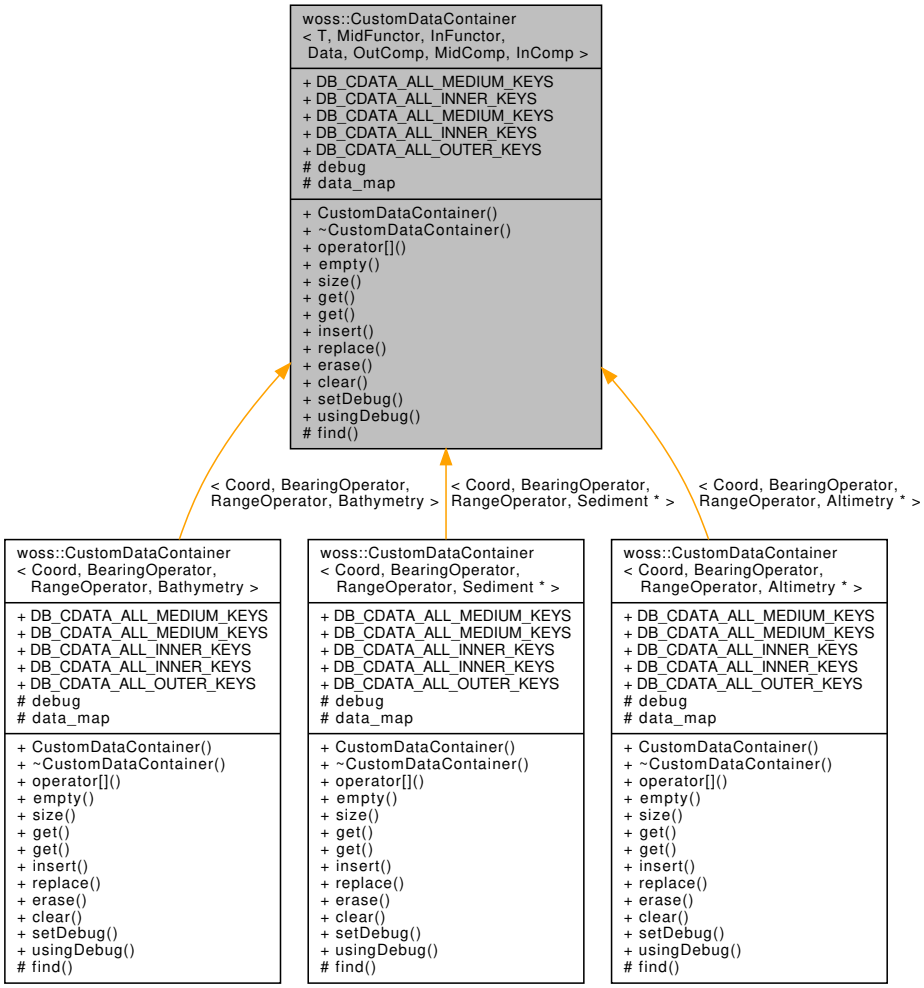
- woss/bellhop-creator.h

13.23 woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp > Class Template Reference

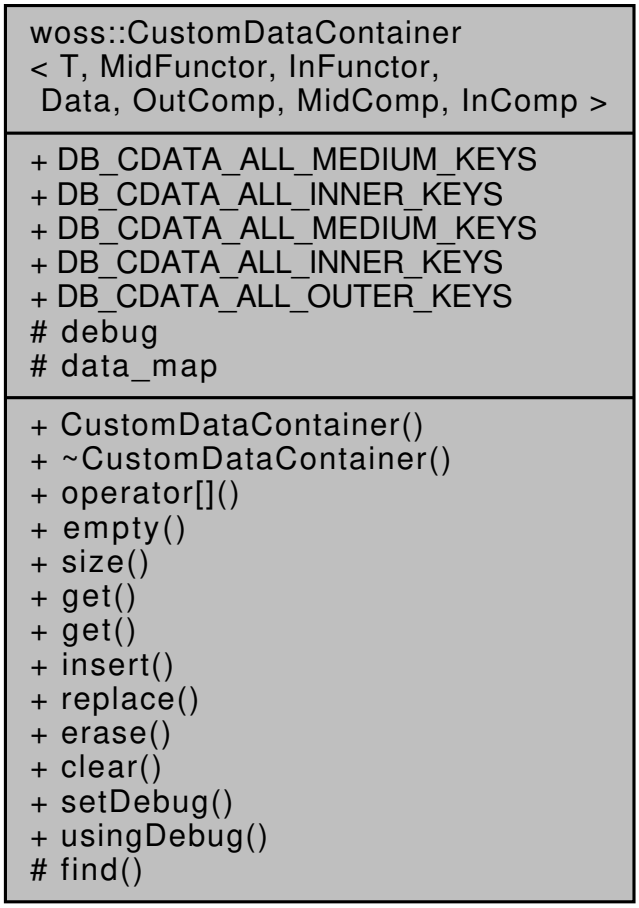
Class for managing custom db data.

`#include <woss-db-custom-data-container.h>`

Inheritance diagram for woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >:



Collaboration diagram for woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >:



Public Member Functions

- CustomDataContainer ()
- ~CustomDataContainer ()
- MediumData & operator[ ] (const T &key)
- bool empty () const
- int size () const
- const Data * get (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS) const
- const Data * get (const T &tx, const T &rx) const
- bool insert (const Data &data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void replace (const Data &data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void erase (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void clear ()
- void setDebug (bool flag)
- bool usingDebug ()

Static Public Attributes

- static constexpr double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static constexpr double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static const double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const T **DB_CDATA_ALL_OUTER_KEYS** = T()

Protected Types

- typedef ::std::map< double, Data, InComp > InnerData
- typedef InnerData::iterator **CDCInnerIt**
- typedef InnerData::reverse_iterator **CDCInnerRIt**
- typedef InnerData::const_iterator **CDCInnerCIt**
- typedef InnerData::const_reverse_iterator **CDCInnerCRIt**
- typedef ::std::map< double, InnerData, MidComp > MediumData
- typedef MediumData::iterator **CDCMediumIt**
- typedef MediumData::const_iterator **CDCMediumCIt**
- typedef MediumData::reverse_iterator **CDCMediumRIt**
- typedef MediumData::const_reverse_iterator **CDCMediumCRIt**
- typedef ::std::map< T, MediumData, OutComp > CustomContainer
- typedef CustomContainer::iterator **CDCIt**
- typedef CustomContainer::reverse_iterator **CDCRIt**
- typedef CustomContainer::const_iterator **CDCCIt**
- typedef CustomContainer::const_reverse_iterator **CDCCRIt**

Protected Member Functions

- const Data * find (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS) const

Protected Attributes

- bool debug
- CustomContainer data_map

13.23.1 Detailed Description

template<class T, class MidFunctor, class InFunctor, class Data, class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
class woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >

Class for managing custom db data.

CustomDataContainer is a template that manages environmental data provided by the user. It sorts data by the first template parameter then by two doubles, (e.g. bearing and range)

13.23.2 Member Typedef Documentation

13.23.2.1 CustomContainer

```
template<class T , class MidFunctor , class InFunctor , class Data
, class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
typedef ::std::map< T, MediumData, OutComp > woss::CustomDataContainer< T, MidFunctor, In↩
Functor, Data, OutComp, MidComp, InComp >::CustomContainer  [protected]
```

The outer level map, that links a the first template parameter to a MediumData instance

13.23.2.2 InnerData

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
typedef ::std::map< double, Data, InComp > woss::CustomDataContainer< T, MidFunctor, In↩
Functor, Data, OutComp, MidComp, InComp >::InnerData  [protected]
```

The inner level map, that links a double to actual data

13.23.2.3 MediumData

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
typedef ::std::map< double, InnerData, MidComp > woss::CustomDataContainer< T, MidFunctor,
InFunctor, Data, OutComp, MidComp, InComp >::MediumData  [protected]
```

The medium level map, that links a double to a InnerData instance

13.23.3 Constructor & Destructor Documentation

13.23.3.1 CustomDataContainer()

```
template<class T , class MidFunctor , class InFunctor , class
Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ↩
::std::less<double>>
woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::↩
CustomDataContainer ( )  [inline]
```

CustomDataContainer default constructor

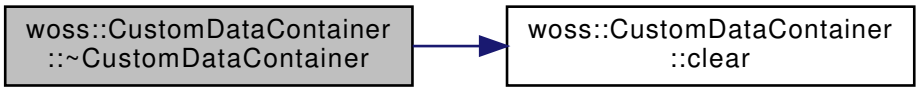
13.23.3.2 ~CustomDataContainer()

```
template<class T , class MidFunctor , class InFunctor ,
class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class In↩
Comp = ::std::less<double>>
woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::~CustomDataContainer
( )  [inline]
```

CustomDataContainer destructor

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::clear().

Here is the call graph for this function:



13.23.4 Member Function Documentation

13.23.4.1 clear() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

`void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::clear`

Clears the map

Referenced by woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::~CustomDataContai

13.23.4.2 empty() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>`

`bool woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty ( ) const [inline]`

Checks if the outer map is empty

Returns

true if it is empty, *false* otherwise

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

Referenced by woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::erase(), woss::WossDbManager::getBathymetry(), and woss::WossDbManager::getSediment().

13.23.4.3 erase() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

```
void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::erase (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS )
```

Erases a Data object for given parameters.

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

true if function was successful, *false* otherwise

Referenced by woss::WossDbManager::eraseCustomAltimetry(), woss::WossDbManager::eraseCustomBathymetry(), and woss::WossDbManager::eraseCustomSediment().

13.23.4.4 find()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
const Data * woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, In↩
Comp >::find (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS ) const  [protected]
```

Finds the given keys. Returns NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

const pointer to a Data object, NULL if not found

Referenced by woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::erase(), and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::insert().

13.23.4.5 get() [1/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
const Data * woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, In↩
Comp >::get (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS ) const
```

Finds and returns a const pointer to a Data for given parameters

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

const pointer to a Data if found, NULL otherwise

Referenced by woss::WossDbManager::getAltimetry(), woss::WossDbManager::getBathymetry(), woss::WossDbManager::getCustom
woss::WossDbManager::getCustomBathymetry(), woss::WossDbManager::getCustomSediment(), and woss::WossDbManager::getS

13.23.4.6 get() [2/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
const Data * woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, In↩
Comp >::get (
            const T & tx,
            const T & rx ) const
```

Finds and returns a const pointer to a Data for the nearest match in all generated keys

Parameters

generator	const reference to a T instance
tobefound	const reference to a T instance

Returns

const pointer to a Data if found, NULL otherwise

13.23.4.7 insert()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
bool woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::insert (
            const Data & data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS )
```

Inserts a Data object for given parameters. If keys are already present, the object is discarded

Parameters

data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

true if function was successful, *false* otherwise

Referenced by woss::WossDbManager::setCustomAltimetry(), woss::WossDbManager::setCustomBathymetry(), and woss::WossDbManager::setCustomSediment().

13.23.4.8 operator[]()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
MediumData & woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, In↩
Comp >::operator[] (
            const T & key )  [inline]
```

operator[]

Parameters

key	a const reference to a T type (first class parameter of template)

Returns

a reference to the linked MediumData

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

13.23.4.9 replace()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::replace (
            const Data & data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS )
```

Replaces a Data object for given parameters. If keys are not present, the object is inserted

Parameters

data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

true if function was successful, *false* otherwise

13.23.4.10 setDebug()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::setDebug (
            bool flag )  [inline]
```

Sets debug flag

Parameters

flag	debug flag

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.23.4.11 size()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
int woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::size ( ) const  [inline]
```

Returns the size of the outer map

Returns

the size

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

Referenced by woss::WossDbManager::getAltimetry(), woss::WossDbManager::getBathymetry(), and woss::WossDbManager::getSe

13.23.4.12 usingDebug()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
bool woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::usingDebug ( )  [inline]
```

Gets debug flag

Returns

flag debug flag

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.23.5 Member Data Documentation

13.23.5.1 data_map

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
CustomContainer woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp,
InComp >::data_map  [protected]
```

data map

Referenced by woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::clear(), woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty(), woss::CustomDataContain woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::find(), woss::CustomDataContaine woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::operator[](), and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::size().

13.23.5.2 debug

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
bool woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >↩
::debug  [protected]
```

Debug flag

Referenced by woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::find(), woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::insert(), woss::CustomDataContai and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::usingDebug().

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

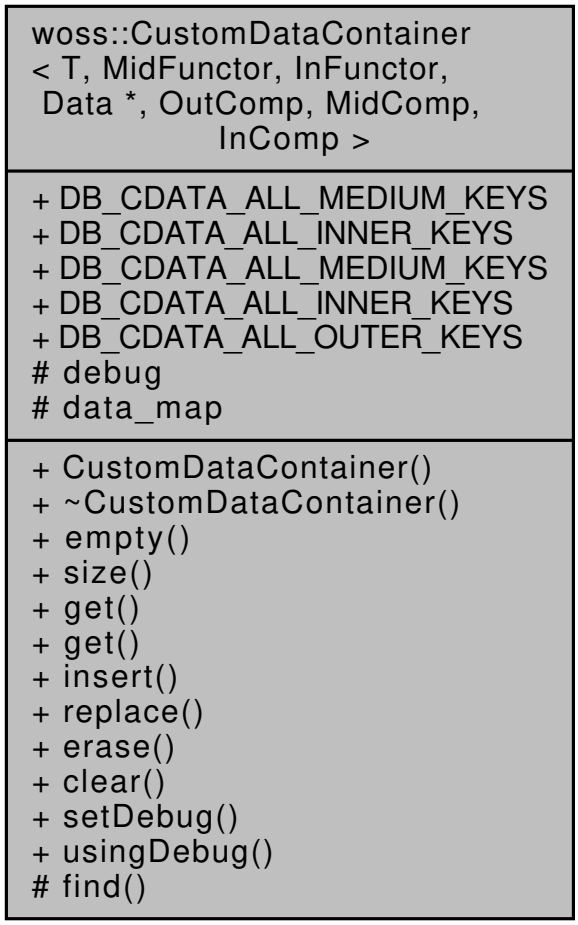
- woss/woss_db/woss-db-custom-data-container.h

13.24 woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp > Class Template Reference

CustomDataContainer template partial specialization for pointers.

```
#include <woss-db-custom-data-container.h>
```

Collaboration diagram for woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >:



Public Member Functions

- bool **empty** () const
- int **size** () const
- Data * **get** (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS) const
- Data * **get** (const T &tx, const T &rx) const
- bool insert (Data *data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_↩ MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void **replace** (Data *data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_↩ MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void erase (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS)
- void clear ()
- void **setDebug** (bool flag)
- bool **usingDebug** ()

Static Public Attributes

- static constexpr double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static constexpr double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static const double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const T **DB_CDATA_ALL_OUTER_KEYS** = T()

Protected Types

- typedef ::std::map< double, Data *, InComp > **InnerData**
- typedef InnerData::iterator **CDCInnerIt**
- typedef InnerData::reverse_iterator **CDCInnerRIt**
- typedef InnerData::const_iterator **CDCInnerCIt**
- typedef InnerData::const_reverse_iterator **CDCInnerCRIt**
- typedef ::std::map< double, InnerData, MidComp > **MediumData**
- typedef MediumData::iterator **CDCMediumIt**
- typedef MediumData::const_iterator **CDCMediumCIt**
- typedef MediumData::reverse_iterator **CDCMediumRIt**
- typedef MediumData::const_reverse_iterator **CDCMediumCRIt**
- typedef ::std::map< T, MediumData, OutComp > **CustomContainer**
- typedef CustomContainer::iterator **CDCIt**
- typedef CustomContainer::reverse_iterator **CDCRIt**
- typedef CustomContainer::const_iterator **CDCCIt**
- typedef CustomContainer::const_reverse_iterator **CDCCRIt**

Protected Member Functions

- Data *& find (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS) const

Protected Attributes

- bool **debug**
- CustomContainer **data_map**

13.24.1 Detailed Description

template<class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class InComp>
class woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >

CustomDataContainer template partial specialization for pointers.

CustomDataContainer template partial specialization for pointers

13.24.2 Member Function Documentation

13.24.2.1 clear() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`
`void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::clear`

Deletes all pointers and clears map

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

13.24.2.2 erase()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >↩
::erase (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS )
```

Deletes the pointer for given parameters.

Parameters

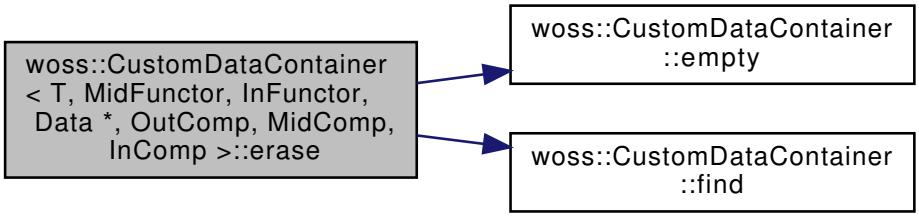
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

true if function was successful, *false* otherwise

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty(), and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



13.24.2.3 find()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
Data *& woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::find (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS ) const  [protected]
```

Finds the given keys. Returns a Data pointer to NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

a reference to a pointer to Data object, NULL if not found

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.24.2.4 insert()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
bool woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >↩
::insert (
            Data * data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS )
```

Inserts a Data pointer for given parameters. If keys are already present, the pointer is deleted

Parameters

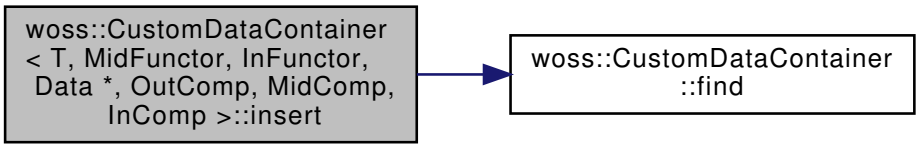
data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData

Returns

true if function was successful, *false* otherwise

References woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



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

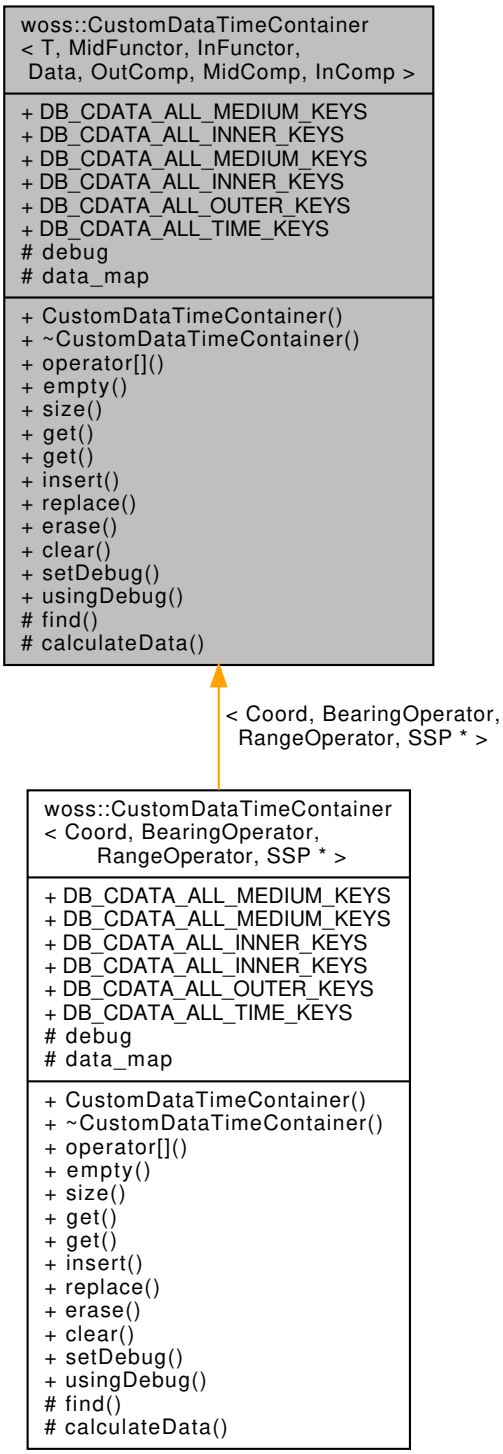
- woss/woss_db/woss-db-custom-data-container.h

13.25 woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp > Class Template Reference

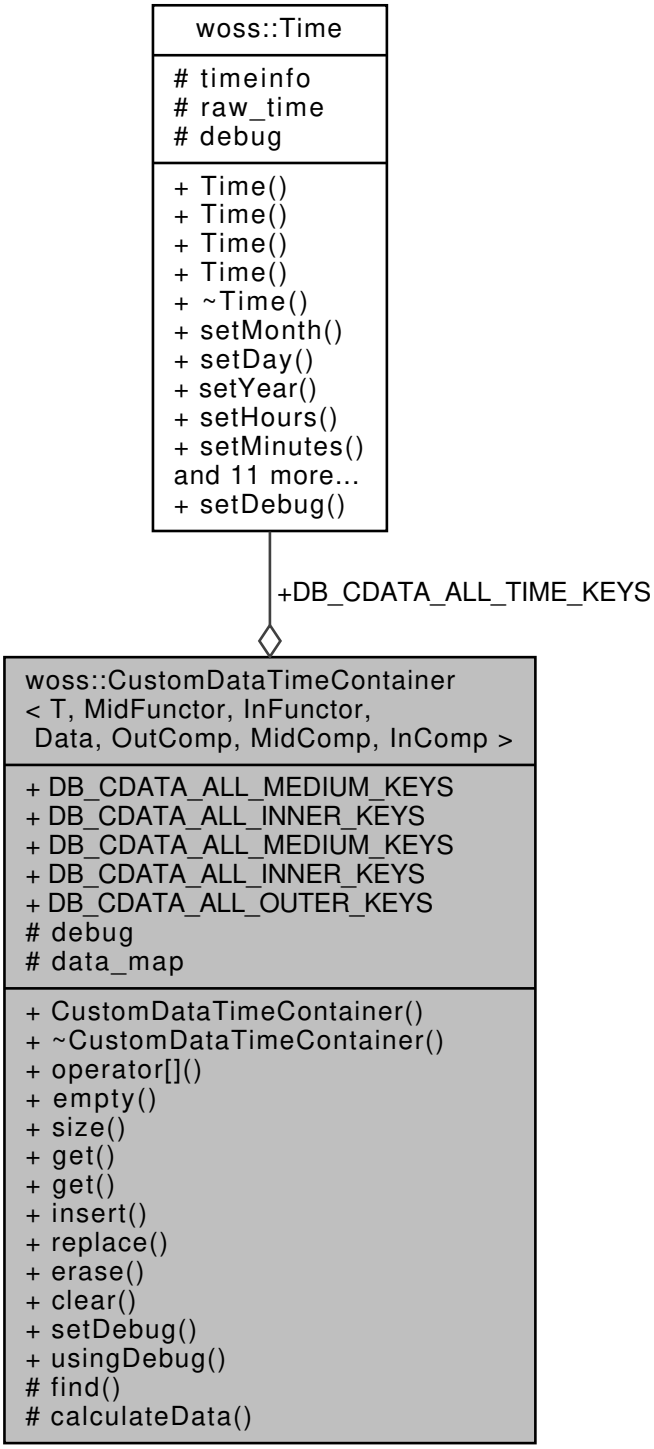
Class for managing custom db data.

`#include <woss-db-custom-data-container.h>`

Inheritance diagram for woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >:



Collaboration diagram for woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >:



Public Member Functions

- CustomDataTimeContainer ()
- ~CustomDataTimeContainer ()
- MediumData & operator[ ] (const T &key)
- bool empty () const

- int size () const
- Data get (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- Data get (const T &tx, const T &rx, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- bool insert (const Data &data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void replace (const Data &data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void erase (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void clear ()
- void setDebug (bool flag)
- bool usingDebug ()

Static Public Attributes

- static constexpr double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static constexpr double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static const double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const T **DB_CDATA_ALL_OUTER_KEYS** = T()
- static const Time **DB_CDATA_ALL_TIME_KEYS** = Time(1, 1, 1901, 0, 0, 0)

Protected Types

- typedef ::std::map< time_t, Data > TimeData
- typedef TimeData::iterator **CDTCTimeIt**
- typedef TimeData::reverse_iterator **CDTCTimeRIt**
- typedef TimeData::const_iterator **CDTCTimeCIt**
- typedef TimeData::const_reverse_iterator **CDTCTimeCRIt**
- typedef ::std::map< double, TimeData, InComp > InnerData
- typedef InnerData::iterator **CDCInnerIt**
- typedef InnerData::reverse_iterator **CDCInnerRIt**
- typedef InnerData::const_iterator **CDCInnerCIt**
- typedef InnerData::const_reverse_iterator **CDCInnerCRIt**
- typedef ::std::map< double, InnerData, MidComp > MediumData
- typedef MediumData::iterator **CDCMediumIt**
- typedef MediumData::const_iterator **CDCMediumCIt**
- typedef MediumData::reverse_iterator **CDCMediumRIt**
- typedef MediumData::const_reverse_iterator **CDCMediumCRIt**
- typedef ::std::map< T, MediumData, OutComp > CustomContainer
- typedef CustomContainer::iterator **CDCIt**
- typedef CustomContainer::reverse_iterator **CDCRIt**
- typedef CustomContainer::const_iterator **CDCCIt**
- typedef CustomContainer::const_reverse_iterator **CDCCRIt**
- typedef ::std::pair< Data, bool > **DataFind**

Protected Member Functions

- DataFind find (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- Data calculateData (const TimeData &time_data, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const

Protected Attributes

- bool debug
- CustomContainer data_map

13.25.1 Detailed Description

template<class T, class MidFunctor, class InFunctor, class Data, class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
class woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >

Class for managing custom db data.

CustomDataTimeContainer is a template that manages environmental data provided by the user. It sorts data by the first template parameter then by two doubles, (e.g. bearing and range) and woss::Time

13.25.2 Member Typedef Documentation

13.25.2.1 CustomContainer

```
template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
typedef ::std::map< T, MediumData, OutComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::CustomContainer [protected]
```

The outer level map, that links a the first template parameter to a MediumData instance

13.25.2.2 InnerData

```
template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
typedef ::std::map< double, TimeData, InComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::InnerData [protected]
```

The inner level map, that links a double to TimeData

13.25.2.3 MediumData

```
template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
typedef ::std::map< double, InnerData, MidComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::MediumData [protected]
```

The medium level map, that links a double to a InnerData instance

13.25.2.4 TimeData

```
template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
typedef ::std::map< time_t, Data > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::TimeData [protected]
```

The Time map, that links a time_t to actual data

13.25.3 Constructor & Destructor Documentation

13.25.3.1 CustomDataTimeContainer() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>`

`woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::CustomDataTimeContainer ( ) [inline]`

CustomDataTimeContainer default constructor

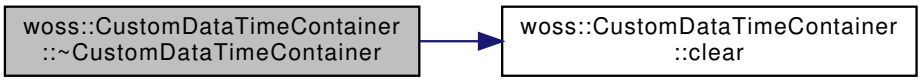
13.25.3.2 ~CustomDataTimeContainer() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>`

`woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::~CustomDataTimeContainer ( ) [inline]`

CustomDataTimeContainer destructor

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::clear().

Here is the call graph for this function:



13.25.4 Member Function Documentation

13.25.4.1 calculateData() `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

`Data woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::calculateData ( const TimeData & time_data, const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const [protected]`

Finds the given keys. Returns NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::get().

13.25.4.2 clear()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::clear
```

Clears the map

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::~CustomDataT

13.25.4.3 empty()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::empty ( ) const  [inline]
```

Checks if the outer map is empty

Returns

true if it is empty, *false* otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::erase(), woss::WossDbManager::getAverageSSP(), and woss::WossDbManager::getSSP().

13.25.4.4 erase()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::erase (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Erases a Data object for given parameters.

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

Referenced by woss::WossDbManager::eraseCustomSSP().

13.25.4.5 find()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::Data↩
Find woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::find (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const  [protected]
```

Finds the given keys. Returns NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

pair of Data and a boolean. if boolean == true ==> Data is valid. Data is not valid otherwise.

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::erase(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::get(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::insert(), and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::replace().

13.25.4.6 get() [1/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
Data woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::get (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const
```

Finds and returns a const pointer to a Data for given parameters

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

Referenced by woss::WossDbManager::getAverageSSP(), woss::WossDbManager::getCustomSSP(), and woss::WossDbManager::getSSP().

13.25.4.7 get() [2/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
Data woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::get (
            const T & tx,
            const T & rx,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const
```

Finds and returns a const pointer to a Data for the nearest match in all generated keys

Parameters

tx	const reference to a T instance
rx	const reference to a T instance
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

13.25.4.8 insert()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::insert (
            const Data & data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
```

```
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Inserts a Data object for given parameters. If keys are already present, the object is discarded

Parameters

data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

Referenced by woss::WossDbManager::importCustomSSP(), and woss::WossDbManager::setCustomSSP().

13.25.4.9 operator[]()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
MediumData & woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp,
InComp >::operator[] (
            const T & key )  [inline]
```

operator[]

Parameters

key	a const reference to a T type (first class parameter of template)

Returns

a reference to the linked MediumData

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

13.25.4.10 replace()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::replace (
            const Data & data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Replaces a Data object for given parameters. If keys are not present, the object is inserted

Parameters

data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

13.25.4.11 setDebug()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::setDebug (
            bool flag )  [inline]
```

Sets debug flag

Parameters

flag	debug flag

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.25.4.12 size()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
int woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::size ( ) const  [inline]
```

Returns the size of the outer map

Returns

the size

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

13.25.4.13 usingDebug()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std↩
::less<double>>
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::usingDebug ( )  [inline]
```

Gets debug flag

Returns

flag debug flag

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.25.5 Member Data Documentation

13.25.5.1 data_map

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
CustomContainer woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, Mid↩
Comp, InComp >::data_map  [protected]
```

data map

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::clear(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::erase(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::find(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::get(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::insert(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::operator[](), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::replace(), and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::size().

13.25.5.2 debug

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp = ::std::less<T>, class MidComp = ::std::less<double>, class InComp = ::std::less<double>>
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
>::debug  [protected]
```

Debug flag

Referenced by woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::calculateData(woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::find(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::get(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::replace(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::setDebug(), and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::usingDebug().

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

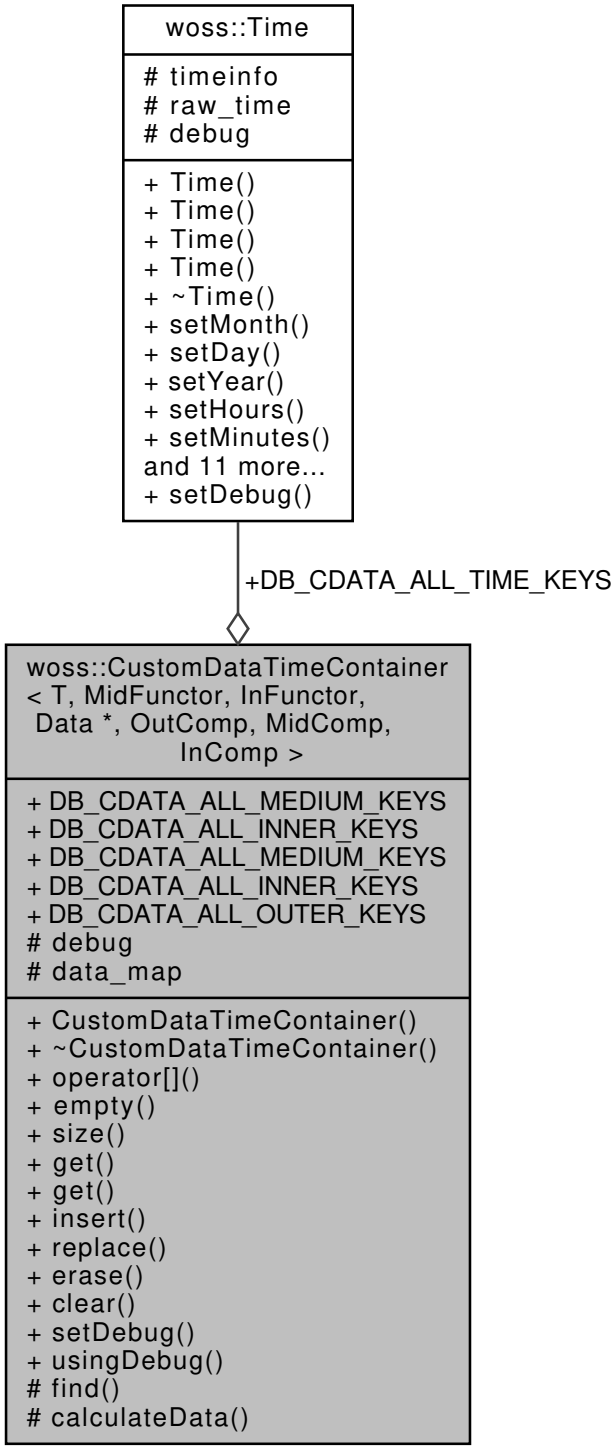
- woss/woss_db/woss-db-custom-data-container.h

13.26 woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp > Class Template Reference

Class for managing custom db data.

`#include <woss-db-custom-data-container.h>`

Collaboration diagram for woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >:



Public Member Functions

- CustomDataTimeContainer ()
- ~CustomDataTimeContainer ()
- MediumData & operator[ ] (const T &key)
- bool empty () const
- int size () const
- Data * get (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- Data * get (const T &tx, const T &rx, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- bool insert (Data *data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void replace (Data *data, const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void erase (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS)
- void clear ()
- void setDebug (bool flag)
- bool usingDebug ()

Static Public Attributes

- static constexpr double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static constexpr double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const double **DB_CDATA_ALL_MEDIUM_KEYS** = -190.0
- static const double **DB_CDATA_ALL_INNER_KEYS** = -10.0
- static const T **DB_CDATA_ALL_OUTER_KEYS** = T()
- static const Time **DB_CDATA_ALL_TIME_KEYS** = Time(1, 1, 1901, 0, 0, 0)

Protected Types

- typedef ::std::map< time_t, Data * > TimeData
- typedef TimeData::iterator **CDTCTimeIt**
- typedef TimeData::reverse_iterator **CDTCTimeRIt**
- typedef TimeData::const_iterator **CDTCTimeCIt**
- typedef TimeData::const_reverse_iterator **CDTCTimeCRIt**
- typedef ::std::map< double, TimeData, InComp > InnerData
- typedef InnerData::iterator **CDCInnerIt**
- typedef InnerData::reverse_iterator **CDCInnerRIt**
- typedef InnerData::const_iterator **CDCInnerCIt**
- typedef InnerData::const_reverse_iterator **CDCInnerCRIt**
- typedef ::std::map< double, InnerData, MidComp > MediumData
- typedef MediumData::iterator **CDCMediumIt**
- typedef MediumData::const_iterator **CDCMediumCIt**
- typedef MediumData::reverse_iterator **CDCMediumRIt**
- typedef MediumData::const_reverse_iterator **CDCMediumCRIt**
- typedef ::std::map< T, MediumData, OutComp > CustomContainer
- typedef CustomContainer::iterator **CDCIt**
- typedef CustomContainer::reverse_iterator **CDCRIt**
- typedef CustomContainer::const_iterator **CDCCIt**
- typedef CustomContainer::const_reverse_iterator **CDCCRIt**
- typedef ::std::pair< Data *, bool > **DataFind**

Protected Member Functions

- DataFind find (const T &t=DB_CDATA_ALL_OUTER_KEYS, double b=DB_CDATA_ALL_MEDIUM_KEYS, double r=DB_CDATA_ALL_INNER_KEYS, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const
- Data * calculateData (const TimeData &time_data, const Time &time_key=DB_CDATA_ALL_TIME_KEYS) const

Protected Attributes

- bool debug
- CustomContainer data_map

13.26.1 Detailed Description

template<class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class InComp>
class woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >

Class for managing custom db data.

CustomDataTimeContainer is a template that manages environmental data provided by the user. It sorts data by the first template parameter then by two doubles, (e.g. bearing and range) and woss::Time

13.26.2 Member Typedef Documentation

13.26.2.1 CustomContainer `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

`typedef ::std::map< T, MediumData, OutComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::CustomContainer [protected]`

The outer level map, that links a the first template parameter to a MediumData instance

13.26.2.2 InnerData `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

`typedef ::std::map< double, TimeData, InComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::InnerData [protected]`

The inner level map, that links a double to TimeData

13.26.2.3 MediumData `template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >`

`typedef ::std::map< double, InnerData, MidComp > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::MediumData [protected]`

The medium level map, that links a double to a InnerData instance

13.26.2.4 TimeData

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
typedef ::std::map< time_t, Data* > woss::CustomDataTimeContainer< T, MidFunctor, InFunctor,
Data *, OutComp, MidComp, InComp >::TimeData  [protected]
```

The Time map, that links a time_t to actual data

13.26.3 Constructor & Destructor Documentation

13.26.3.1 CustomDataTimeContainer()

```
template<class T , class MidFunctor , class InFunctor ,
class Data , class OutComp , class MidComp , class InComp >
woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >↩
::CustomDataTimeContainer ( )  [inline]
```

CustomDataTimeContainer default constructor

13.26.3.2 ~CustomDataTimeContainer()

```
template<class T , class MidFunctor , class InFunctor ,
class Data , class OutComp , class MidComp , class InComp >
woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >↩
::~CustomDataTimeContainer ( )  [inline]
```

CustomDataTimeContainer destructor

13.26.4 Member Function Documentation

13.26.4.1 calculateData()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
Data * woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, In↩
Comp >::calculateData (
            const TimeData & time_data,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const  [protected]
```

Finds the given keys. Returns NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.26.4.2 clear()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::clear
```

Clears the map

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map.

13.26.4.3 empty()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::empty ( ) const  [inline]
```

Checks if the outer map is empty

Returns

true if it is empty, *false* otherwise

13.26.4.4 erase()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::erase (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Erases a Data object for given parameters.

Parameters

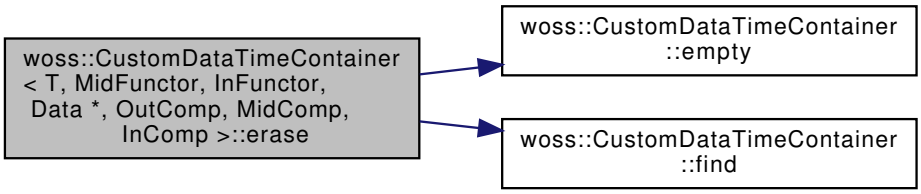
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty(), and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



13.26.4.5 find()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::Data↩
Find woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::find (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const  [protected]
```

Finds the given keys. Returns NULL if not found

Parameters

t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

pair of Data and a boolean. if boolean == true ==> Data is valid. Data is not valid otherwise.

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

13.26.4.6 get() [1/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
Data * woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, In↩
Comp >::get (
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const
```

Finds and returns a const pointer to a Data for given parameters

Parameters

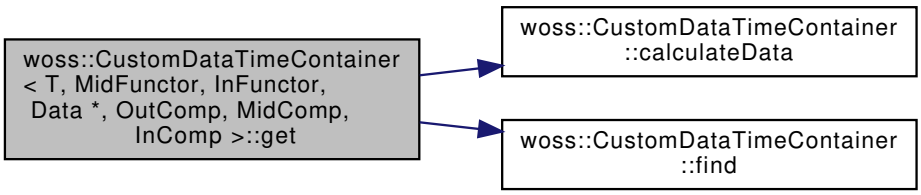
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::calculateData(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



13.26.4.7 get() [2/2]

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
Data * woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, In↩
Comp >::get (
            const T & tx,
            const T & rx,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS ) const
```

Finds and returns a const pointer to a Data for the nearest match in all generated keys

Parameters

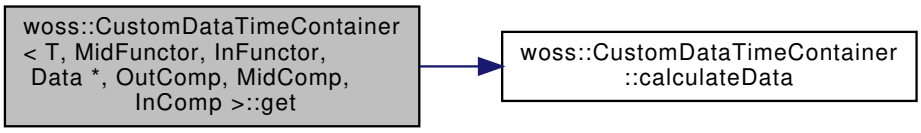
tx	const reference to a T instance
rx	const reference to a T instance
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

valid Data object if map is not empty. Data() otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::calculateData(), woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug.

Here is the call graph for this function:



13.26.4.8 insert()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::insert (
            Data * data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Inserts a Data object for given parameters. If keys are already present, the object is discarded

Parameters

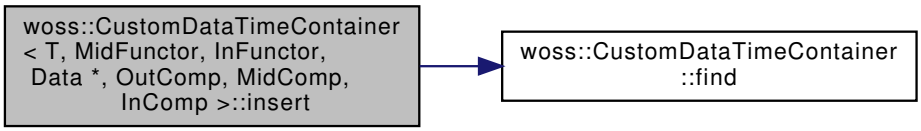
data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



13.26.4.9 operator[]()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
MediumData & woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, Mid↩
Comp, InComp >::operator[] (
            const T & key )  [inline]
```

operator[]

Parameters

key	a const reference to a T type (first class parameter of template)

Returns

a reference to the linked MediumData

13.26.4.10 replace()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::replace (
            Data * data,
            const T & t = DB_CDATA_ALL_OUTER_KEYS,
            double b = DB_CDATA_ALL_MEDIUM_KEYS,
            double r = DB_CDATA_ALL_INNER_KEYS,
            const Time & time_key = DB_CDATA_ALL_TIME_KEYS )
```

Replaces a Data object for given parameters. If keys are not present, the object is inserted

Parameters

data	const reference to a Data object to be inserted
t	const reference to a T instance, default value is DB_CDATA_ALL_OUTER_KEYS. Default value means valid for all possible instances of T

Parameters

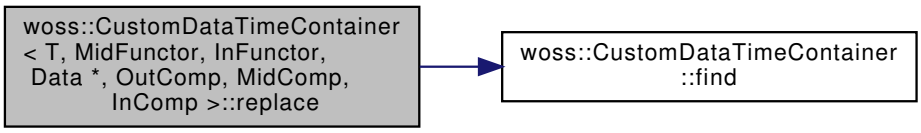
b	double, default value is DB_CDATA_ALL_MEDIUM_KEYS. Default value means valid for all possible doubles of MediumData
r	double, default value is DB_CDATA_ALL_INNER_KEYS. Default value means valid for all possible doubles of InnerData
time_key	const reference to a Time instance, default value is DB_CDATA_ALL_TIME_KEYS. Default value means it is valid for all possible Time values of TimeData

Returns

true if function was successful, *false* otherwise

References woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::data_map, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::debug, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find().

Here is the call graph for this function:



13.26.4.11 setDebug()

```
template<class T , class MidFunctor , class InFunctor , class Data ,
class OutComp , class MidComp , class InComp >
void woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::setDebug (
            bool flag )  [inline]
```

Sets debug flag

Parameters

flag	debug flag

13.26.4.12 size()

```
template<class T , class MidFunctor , class InFunctor , class Data , class
OutComp , class MidComp , class InComp >
int woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp
>::size ( ) const  [inline]
```

Returns the size of the outer map

Returns

the size

13.26.4.13 usingDebug() template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::usingDebug ( ) [inline]

Gets debug flag

Returns

flag debug flag

13.26.5 Member Data Documentation

13.26.5.1 data_map template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >
CustomContainer woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::data_map [protected]

data map

13.26.5.2 debug template<class T , class MidFunctor , class InFunctor , class Data , class OutComp , class MidComp , class InComp >
bool woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >::debug [protected]

Debug flag

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

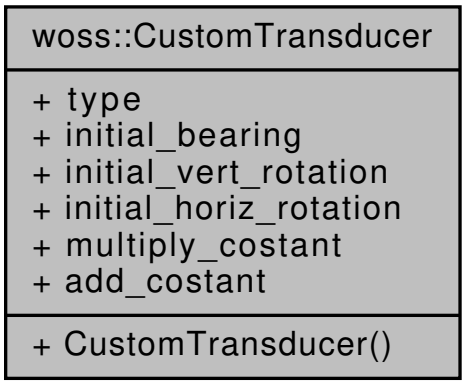
- woss/woss_db/woss-db-custom-data-container.h

13.27 woss::CustomTransducer Struct Reference

Initial set up of a transducer.

`#include <woss-creator-container.h>`

Collaboration diagram for woss::CustomTransducer:



Public Member Functions

- CustomTransducer (const ::std::string &name="", double bearing=0.0, double vert_rot=0.0, double horiz_↩ rot=0.0, double mult=1.0, double add=0.0)

Public Attributes

- ::std::string type
- double initial_bearing
- double initial_vert_rotation
- double initial_horiz_rotation
- double multiply_costant
- double add_costant

Friends

- std::ostream & **operator**<< (std::ostream &os, const CustomTransducer &instance)

13.27.1 Detailed Description

Initial set up of a transducer.

Struct that stores the Transducer's type name, its vertical beam pattern initial rotation, multiplicative and additive constants.

13.27.2 Constructor & Destructor Documentation

13.27.2.1 CustomTransducer() `woss::CustomTransducer::CustomTransducer (`

```
const ::std::string & name = "",
double bearing = 0.0,
double vert_rot = 0.0,
double horiz_rot = 0.0,
double mult = 1.0,
double add = 0.0 )  [inline]
```

Default constructor

Parameters

name	transducer's type name
rot	initial rotation [decimal degrees]
mult	multiplicative constant
add	additive constant

13.27.3 Member Data Documentation

13.27.3.1 add_costant `double woss::CustomTransducer::add_costant`

beam pattern additive constant

Referenced by woss::BellhopCreator::initializeBhWoss().

13.27.3.2 initial_bearing `double woss::CustomTransducer::initial_bearing`

beam pattern bearing orientation

Referenced by woss::BellhopCreator::initializeBhWoss().

13.27.3.3 initial_horiz_rotation `double woss::CustomTransducer::initial_horiz_rotation`

beam pattern initial horizontal rotation [decimal degrees]

Referenced by woss::BellhopCreator::initializeBhWoss().

13.27.3.4 initial_vert_rotation `double woss::CustomTransducer::initial_vert_rotation`

beam pattern initial vertical rotation [decimal degrees]

Referenced by woss::BellhopCreator::initializeBhWoss().

13.27.3.5 multiply_costant `double woss::CustomTransducer::multiply_costant`

beam pattern multiplicative constant

Referenced by woss::BellhopCreator::initializeBhWoss().

13.27.3.6 type `::std::string woss::CustomTransducer::type`

transducer type name

Referenced by woss::BellhopCreator::initializeBhWoss().

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

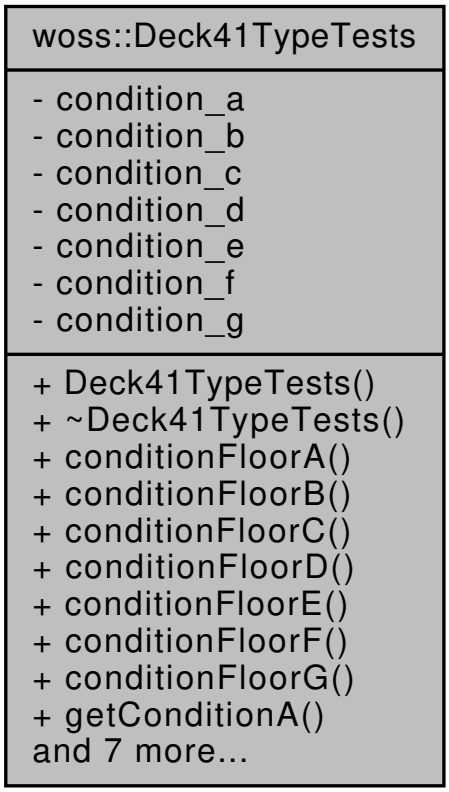
- woss/woss-creator-container.h

13.28 woss::Deck41TypeTests Class Reference

Abstraction layer for database and data manipulation.

`#include <sediment-deck41-db-logic-control.h>`

Collaboration diagram for woss::Deck41TypeTests:



Public Member Functions

- Deck41TypeTests ()
- bool conditionFloorA (const Deck41Types &types) const
- bool conditionFloorB (const Deck41Types &types) const
- bool conditionFloorC (const Deck41Types &types) const
- bool conditionFloorD (const Deck41Types &types) const
- bool conditionFloorE (const Deck41Types &types) const
- bool conditionFloorF (const Deck41Types &types) const
- bool conditionFloorG (const Deck41Types &types) const
- bool getConditionA () const
- bool getConditionB () const
- bool getConditionC () const
- bool getConditionD () const
- bool getConditionE () const
- bool getConditionF () const
- bool getConditionG () const
- void updateAllConditions (const Deck41Types &types)

Private Attributes

- bool **condition_a**
- bool **condition_b**
- bool **condition_c**
- bool **condition_d**
- bool **condition_e**
- bool **condition_f**
- bool **condition_g**

13.28.1 Detailed Description

Abstraction layer for database and data manipulation.

Deck41TypeTests helps SedimDeck41Db deciding what is the best DECK41 Sediment value to return. To this purpose it provides 8 conditional tests

13.28.2 Constructor & Destructor Documentation

13.28.2.1 Deck41TypeTests() `Deck41TypeTests::Deck41TypeTests ( )`

Deck41TypeTests default constructor

13.28.3 Member Function Documentation

13.28.3.1 conditionFloorA() `bool woss::Deck41TypeTests::conditionFloorA ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test A. If it succeed SedimDeck41Db will return DECK41 main type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.2 conditionFloorB() `bool woss::Deck41TypeTests::conditionFloorB ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test B. If it succeed SedimDeck41Db will return DECK41 secondary type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.3 conditionFloorC() `bool woss::Deck41TypeTests::conditionFloorC ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test C. If it succeed SedimDeck41Db will try other coordinates, if those fails it will return DECK41 main type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.4 conditionFloorD() `bool woss::Deck41TypeTests::conditionFloorD ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test D. If it succeed SedimDeck41Db will try other coordinates, if those fails it will return DECK41 secondary type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.5 conditionFloorE() `bool woss::Deck41TypeTests::conditionFloorE ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test E. If it succeed SedimDeck41Db will return DECK41 a weighted average of both main and secondary type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.6 conditionFloorF() `bool woss::Deck41TypeTests::conditionFloorF ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test F. If it succeed SedimDeck41Db will return DECK41 a weighted average of both main and secondary type as Sediment

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.7 conditionFloorG() `bool woss::Deck41TypeTests::conditionFloorG ( const Deck41Types & types ) const [inline]`

Deck41TypeTests test G. If it succeed SedimDeck41Db will try other coordinates, if those fails a Sediment not valid will be returned

Parameters

types	const reference to a Deck41Types object

Returns

true if condition holds, *false* otherwise

Referenced by updateAllConditions().

13.28.3.8 getConditionA() bool woss::Deck41TypeTests::getConditionA ( ) const [inline]

Gets condition_a bool value

Returns

condition_a value

Referenced by woss::SedimDeck41Db::doTestA().

13.28.3.9 getConditionB() bool woss::Deck41TypeTests::getConditionB ( ) const [inline]

Gets condition_b bool value

Returns

condition_b value

Referenced by woss::SedimDeck41Db::doTestA().

13.28.3.10 getConditionC() bool woss::Deck41TypeTests::getConditionC ( ) const [inline]

Gets condition_c bool value

Returns

condition_c value

Referenced by woss::SedimDeck41Db::doTestB(), and woss::SedimDeck41Db::doTestC().

13.28.3.11 getConditionD() bool woss::Deck41TypeTests::getConditionD ( ) const [inline]

Gets condition_d bool value

Returns

condition_d value

Referenced by woss::SedimDeck41Db::doTestB(), and woss::SedimDeck41Db::doTestC().

13.28.3.12 getConditionE() `bool woss::Deck41TypeTests::getConditionE ( ) const [inline]`

Gets condition_e bool value

Returns

condition_e value

Referenced by woss::SedimDeck41Db::doTestA().

13.28.3.13 getConditionF() `bool woss::Deck41TypeTests::getConditionF ( ) const [inline]`

Gets condition_f bool value

Returns

condition_f value

Referenced by woss::SedimDeck41Db::doTestA().

13.28.3.14 getConditionG() `bool woss::Deck41TypeTests::getConditionG ( ) const [inline]`

Gets condition_g bool value

Returns

condition_g value

Referenced by woss::SedimDeck41Db::doTestB().

13.28.3.15 updateAllConditions() `void woss::Deck41TypeTests::updateAllConditions ( const Deck41Types & types ) [inline]`

Performs all tests at once

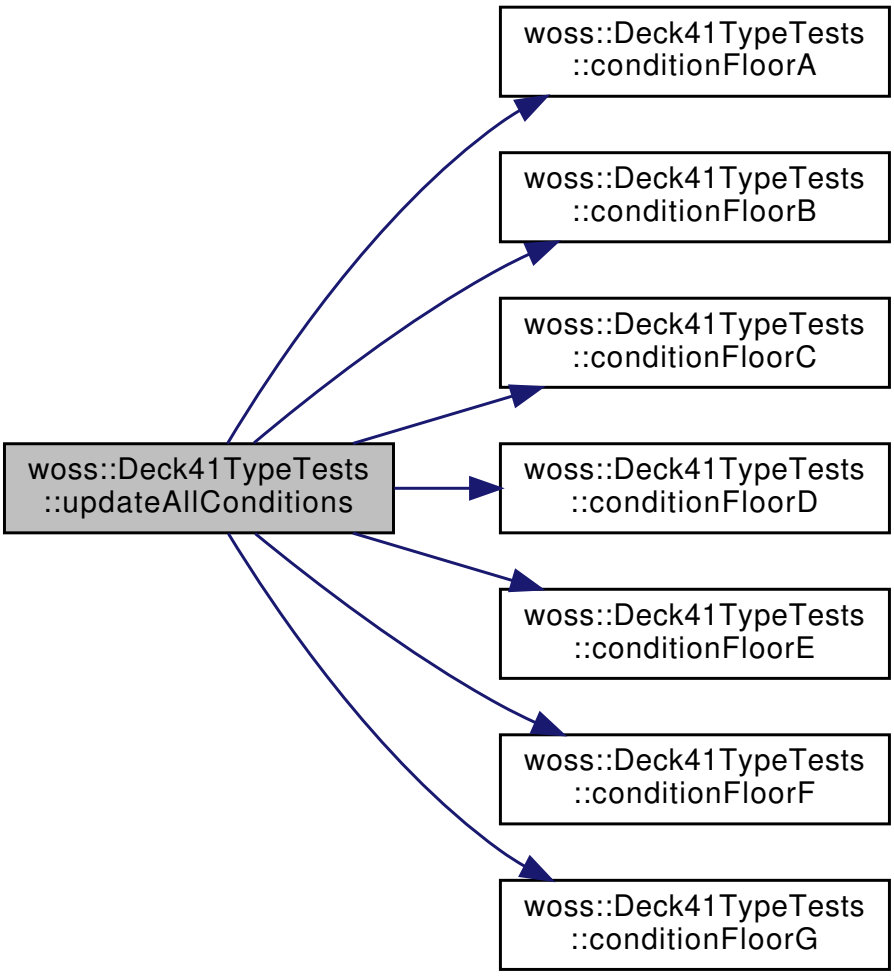
Parameters

types	const reference to a Deck41Types object

References conditionFloorA(), conditionFloorB(), conditionFloorC(), conditionFloorD(), conditionFloorE(), conditionFloorF(), and conditionFloorG().

Referenced by woss::SedimDeck41Db::calculateDeck41Types().

Here is the call graph for this function:



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

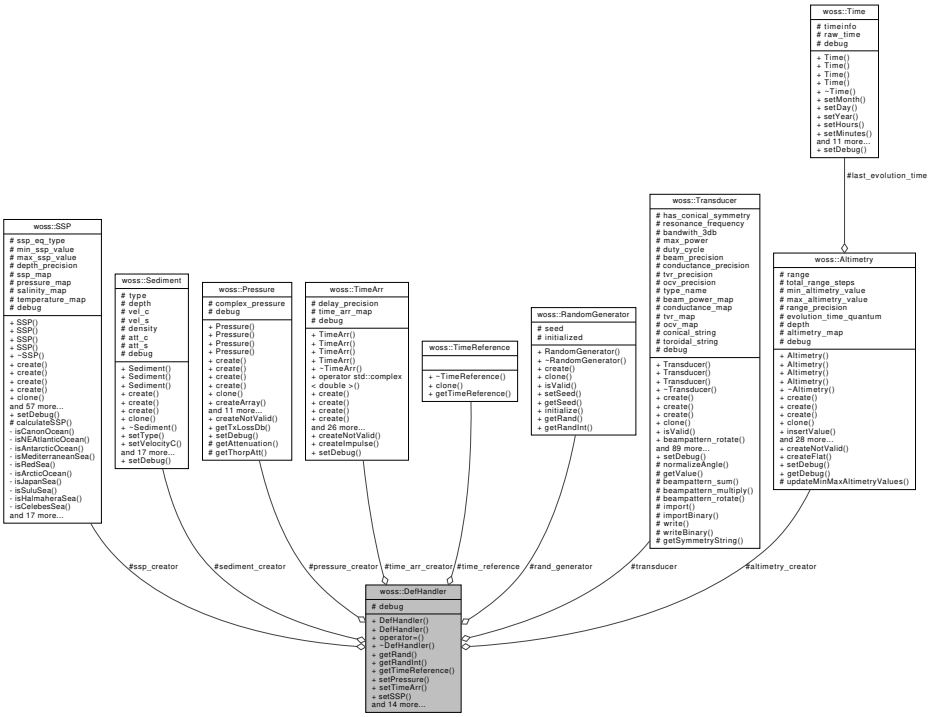
- woss/woss_db/sediment-deck41-db-logic-control.h
- woss/woss_db/sediment-deck41-db-logic-control.cpp

13.29 woss::DefHandler Class Reference

Class for managing dynamic instantiation of foundation classes.

`#include <definitions-handler.h>`

Collaboration diagram for woss::DefHandler:



Public Member Functions

- DefHandler ()
- DefHandler (const DefHandler &copy)
- DefHandler & operator= (const DefHandler &copy)
- virtual ~DefHandler ()
- double getRand () const
- int getRandInt () const
- double getTimeReference () const
- void **setPressure** (Pressure *const ptr)
- void **setTimeArr** (TimeArr *const ptr)
- void **setSSP** (SSP *const ptr)
- void **setSediment** (Sediment *const ptr)
- void **setTimeReference** (TimeReference *const ptr)
- void **setRandGenerator** (RandomGenerator *const ptr)
- void **setTransducer** (Transducer *const ptr)
- void **setAltimetry** (Altimetry *const ptr)
- const Pressure *const **getPressure** () const
- const TimeArr *const **getTimeArr** () const
- const SSP *const **getSSP** () const
- const Sediment *const **getSediment** () const
- RandomGenerator *const **getRandomGenerator** () const
- const Transducer *const **getTransducer** () const
- const Altimetry *const **getAltimetry** () const
- void **setDebug** (bool flag)
- bool **getDebug** ()

Protected Attributes

- bool debug
- SSP * **ssp_creator**
- Sediment * **sediment_creator**
- Pressure * **pressure_creator**
- TimeArr * **time_arr_creator**
- TimeReference * **time_reference**
- RandomGenerator * **rand_generator**
- Transducer * **transducer**
- Altimetry * **altimetry_creator**

13.29.1 Detailed Description

Class for managing dynamic instantiation of foundation classes.

woss::DefHandler is a class that handle dynamic instances of woss::SSP, woss::Sediment, woss::TimeArr, woss::Pressure. A client should aks for a pointer to any of them, and then call the virtual **create** method for an instance of that particular class. In this way a class that inherits from any of those mentioned above, could be "plugged-in" at run time in any WOSS class without changing the code. It should be created with singleton pattern for safety reasons (e.g. woss::Sigleton< woss::DefHandler >)

13.29.2 Constructor & Destructor Documentation

13.29.2.1 DefHandler() [1/2] DefHandler::DefHandler ( )

Default constructor

13.29.2.2 DefHandler() [2/2] DefHandler::DefHandler (const DefHandler & *copy*)

Copy constructor

13.29.2.3 ~DefHandler() DefHandler::~DefHandler ( ) [virtual]

Destructor

13.29.3 Member Function Documentation

13.29.3.1 getRand() `double woss::DefHandler::getRand ( ) const [inline]`

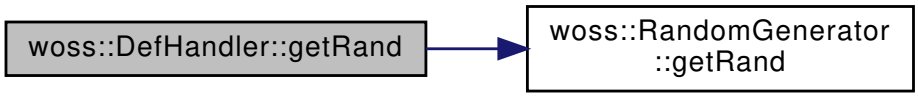
Returns a random value from the connected woss::RandomGenerator object

Returns

a random value

References woss::RandomGenerator::getRand().

Here is the call graph for this function:



13.29.3.2 getRandInt() `int woss::DefHandler::getRandInt ( ) const [inline]`

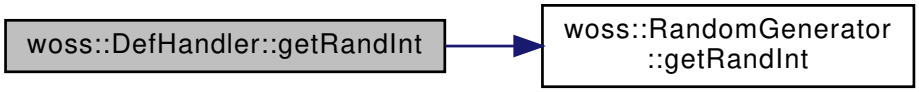
Returns a random integer from the connected woss::RandomGenerator object

Returns

a random integer value

References woss::RandomGenerator::getRandInt().

Here is the call graph for this function:



13.29.3.3 getTimeReference() `double woss::DefHandler::getTimeReference ( ) const [inline]`

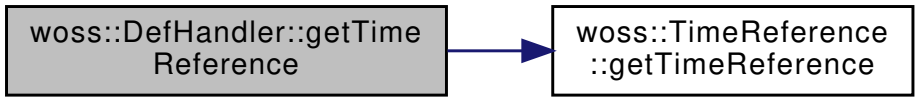
Returns the current simulation time reference from the connected woss::TimeReference object

Returns

a simulation time reference in seconds

References woss::TimeReference::getTimeReference().

Here is the call graph for this function:

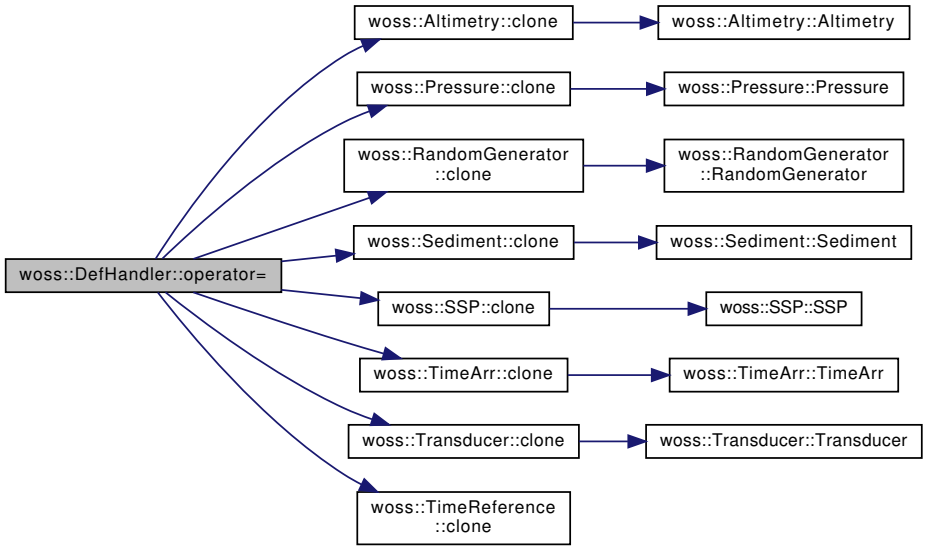


13.29.3.4 operator=() `DefHandler & DefHandler::operator= ( const DefHandler & copy )`

Assignment operator

References woss::Altimetry::clone(), woss::Pressure::clone(), woss::RandomGenerator::clone(), woss::Sediment::clone(), woss::SSP::clone(), woss::TimeArr::clone(), woss::Transducer::clone(), woss::TimeReference::clone(), and debug.

Here is the call graph for this function:



13.29.4 Member Data Documentation

13.29.4.1 debug `bool woss::DefHandler::debug [protected]`

Debug flag

Referenced by operator=().

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

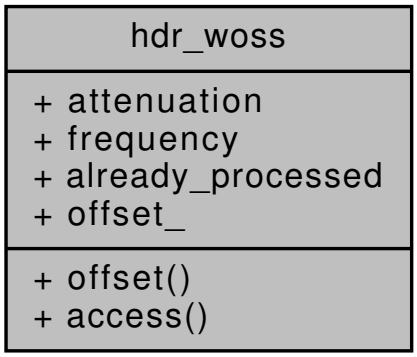
- woss/woss_def/definitions-handler.h
- woss/woss_def/definitions-handler.cpp

13.30 hdr_woss Struct Reference

WOSS packet header.

`#include <uw-woss-pkt-hdr.h>`

Collaboration diagram for hdr_woss:



Static Public Member Functions

- static int & **offset** ()
- static struct hdr_woss * **access** (const Packet *p)

Public Attributes

- std::complex< double > attenuation
- double frequency
- bool already_processed

Static Public Attributes

- static int **offset_** = 0

13.30.1 Detailed Description

WOSS packet header.

struct hdr_woss extends Miracle MPhy header adding new information

13.30.2 Member Data Documentation

13.30.2.1 already_processed `bool hdr_woss::already_processed`

bool that informs WossMPropagation that WossChannelModule has already processed this Packet

Referenced by WossMPropagation::computeGain(), and WossMPropagation::getGain().

13.30.2.2 attenuation `std::complex<double> hdr_woss::attenuation`

complex attenuation of current Packet, calculated by woss::WossManager

Referenced by WossMPropagation::computeGain(), and WossMPropagation::getGain().

13.30.2.3 frequency `double hdr_woss::frequency`

frequency of calculations

Referenced by WossMPropagation::computeGain().

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

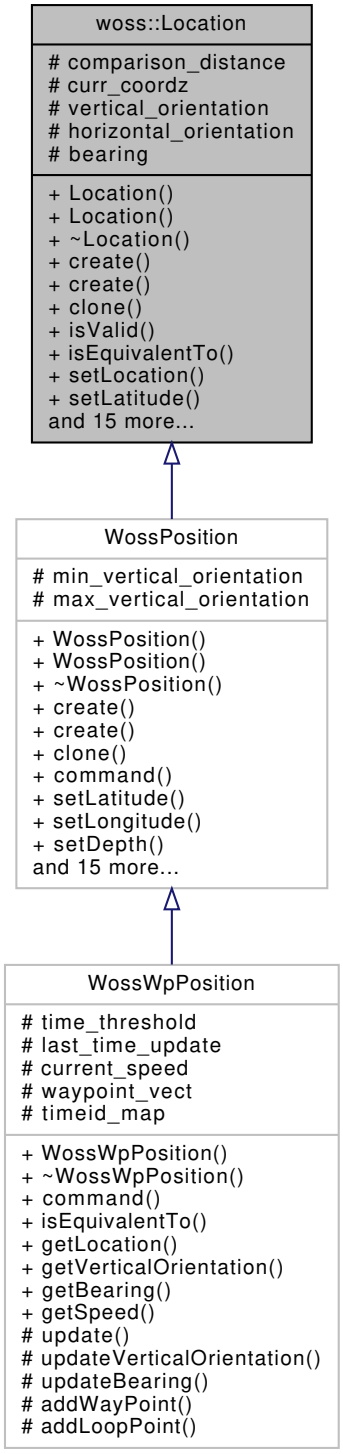
- woss_phy/uw-woss-pkt-hdr.h
- woss_phy/initlib.cc

13.31 woss::Location Class Reference

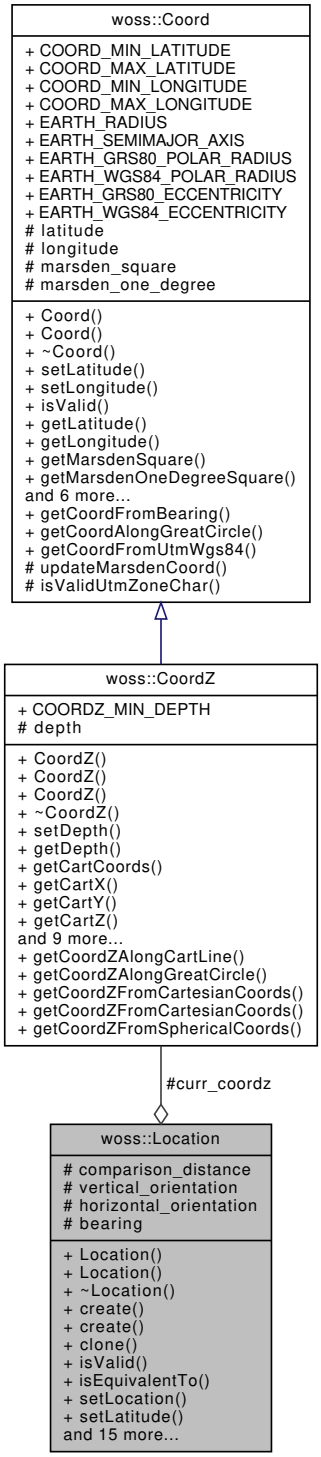
Class that stores the coordinates of moving entity.

`#include <location-definitions.h>`

Inheritance diagram for woss::Location:



Collaboration diagram for woss::Location:



Public Member Functions

- Location (const CoordZ &coordz=CoordZ(), double dist=LOCATION_COMPARISON_DISTANCE)
- Location (double latitude, double longitude, double depth=0, double dist=LOCATION_COMPARISON_DISTANCE)
- virtual ~Location ()
- virtual Location * create (double latitude, double longitude, double depth=0, double dist=LOCATION_COMPARISON_DISTANCE) const

- virtual Location * create (const CoordZ &coordz=CoordZ(), double dist=LOCATION_COMPARISON_↩ DISTANCE) const
- virtual Location * clone () const
- virtual bool isValid () const
- virtual bool isEquivalentTo (const woss::CoordZ &coordz)
- virtual void setLocation (const CoordZ &coordz)
- virtual void setLatitude (double lat)
- virtual void setLongitude (double lon)
- virtual void setDepth (double depth)
- virtual void setVerticalOrientation (double angle)
- virtual void setHorizontalOrientation (double angle)
- virtual CoordZ getLocation ()
- virtual double getLatitude ()
- virtual double getLongitude ()
- virtual double getDepth ()
- virtual double getX ()
- virtual double getY ()
- virtual double getZ ()
- virtual double getVerticalOrientation ()
- virtual double getHorizontalOrientation ()
- virtual double getBearing ()
- friend::std::ostream & operator<< (::std::ostream &os, const Location &instance)

Protected Attributes

- double comparison_distance
- CoordZ curr_coordz
- double vertical_orientation
- double horizontal_orientation
- double bearing

13.31.1 Detailed Description

Class that stores the coordinates of moving entity.

The woss::Location class has the task of storing the geographical coordinates of a moving entity

13.31.2 Constructor & Destructor Documentation

13.31.2.1 Location() [1/2]

```
Location::Location (
        const CoordZ & coordz = CoordZ(),
        double dist = LOCATION_COMPARISON_DISTANCE )
```

Location default constructor

Parameters

coordz	coordinates
dist	distance comparison precision [m]

Referenced by clone(), and create().

13.31.2.2 Location() [2/2]

```
Location::Location (
        double latitude,
        double longitude,
        double depth = 0,
        double dist = LOCATION_COMPARISON_DISTANCE )
```

Location constructor

Parameters

latitude	latitude value [decimal degrees]
longitude	longitude value [decimal degrees]
depth	depth value [m]
dist	distance comparison precision [m]

13.31.2.3 ~Location()

```
virtual woss::Location::~Location ( )  [inline], [virtual]
```

Location destructor

13.31.3 Member Function Documentation

13.31.3.1 clone()

```
virtual Location * woss::Location::clone ( ) const  [inline], [virtual]
```

Location virtual factory method

Returns

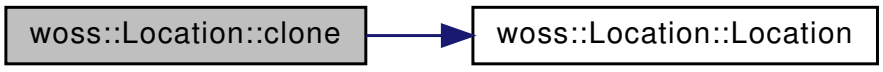
a heap-allocated copy of **this** instance

Reimplemented in WossPosition.

References Location().

Referenced by woss::WossCreatorContainer< Data * >::get().

Here is the call graph for this function:



13.31.3.2 create() [1/2]

```
virtual Location * woss::Location::create (
        const CoordZ & coordz = CoordZ(),
        double dist = LOCATION_COMPARISON_DISTANCE ) const  [inline], [virtual]
```

Location virtual factory method

Parameters

coordz	coordinates
dist	distance comparison precision [m]

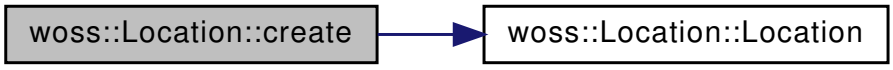
Returns

a heap-allocated Location object

Reimplemented in WossPosition.

References Location().

Here is the call graph for this function:



13.31.3.3 create() [2/2]

```
virtual Location * woss::Location::create (
        double latitude,
        double longitude,
        double depth = 0,
        double dist = LOCATION_COMPARISON_DISTANCE ) const  [inline], [virtual]
```

Location virtual factory method

Parameters

latitude	latitude value [decimal degrees]
longitude	longitude value [decimal degrees]
depth	depth value [m]
dist	distance comparison precision [m]

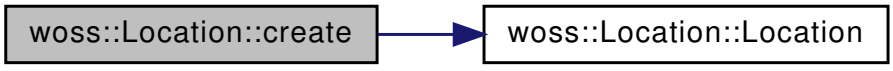
Returns

a heap-allocated Location object

Reimplemented in WossPosition.

References Location().

Here is the call graph for this function:



13.31.3.4 getBearing() `double Location::getBearing ( ) [virtual]`

Gets current bearing in [-pi,pi]

Returns

bearing [dec degrees]

Reimplemented in WossWpPosition.

References bearing.

13.31.3.5 getDepth() `double Location::getDepth ( ) [virtual]`

Gets current depth

Returns

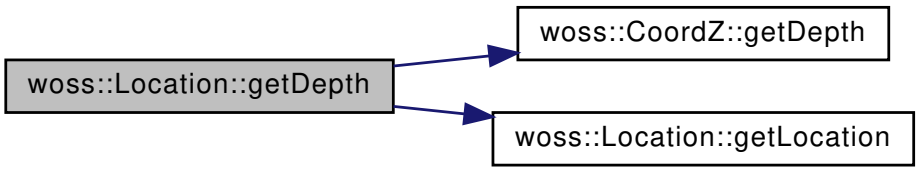
valid depth [m]

Reimplemented in WossPosition.

References woss::CoordZ::getDepth(), and getLocation().

Referenced by WossPosition::getDepth().

Here is the call graph for this function:



13.31.3.6 getHorizontalOrientation() `double Location::getHorizontalOrientation ( ) [virtual]`

Gets current horizontal orientation from reference line (0 degrees = parallel to direction of current movement) positive values are counter-clockwise rotations, while negative ones are clockwise rotations

Returns

difference angle [dec degrees]

References horizontal_orientation.

13.31.3.7 getLatitude() `double Location::getLatitude ( ) [virtual]`

Gets current latitude

Returns

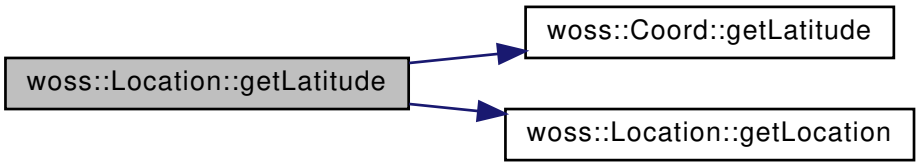
valid latitude [decimal degrees]

Reimplemented in WossPosition.

References woss::Coord::getLatitude(), and getLocation().

Referenced by WossPosition::getLatitude().

Here is the call graph for this function:



13.31.3.8 getLocation() `CoordZ Location::getLocation ( ) [virtual]`

Gets current coordinates

Returns

valid woss::CoordZ

Reimplemented in WossWpPosition.

References curr_coordz.

Referenced by getDepth(), getLatitude(), WossWpPosition::getLocation(), getLongitude(), getX(), getY(), getZ(), and isEquivalentTo().

13.31.3.9 getLongitude() `double Location::getLongitude ( ) [virtual]`

Gets current longitude

Returns

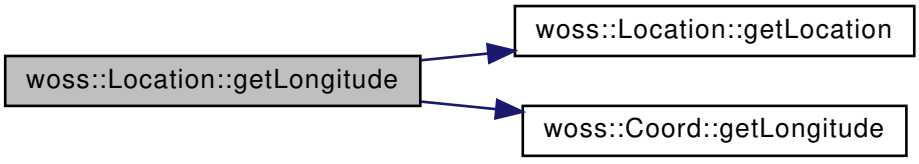
valid longitude [decimal degrees]

Reimplemented in WossPosition.

References getLocation(), and woss::Coord::getLongitude().

Referenced by WossPosition::getLongitude().

Here is the call graph for this function:



13.31.3.10 getVerticalOrientation() `double Location::getVerticalOrientation ( ) [virtual]`

Gets current vertical orientation from reference line (0 degrees = parallel to sea surface / bottom). Negative values are towards the surface, while positive ones are towards sea bottom

Returns

difference angle [dec degrees]

Reimplemented in WossWpPosition.

References vertical_orientation.

13.31.3.11 getX() `double Location::getX ( ) [virtual]`

Gets current cartesian x-axis value

Returns

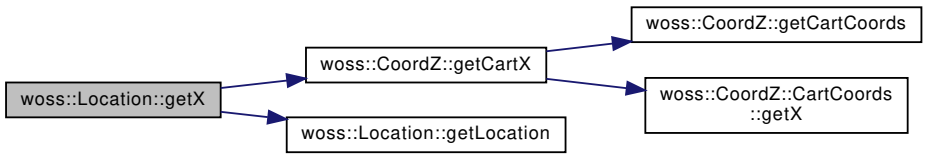
x value

Reimplemented in WossPosition.

References woss::CoordZ::getCartX(), and getLocation().

Referenced by WossPosition::getX().

Here is the call graph for this function:



13.31.3.12 getY() `double Location::getY ( ) [virtual]`

Gets current cartesian y-axis value

Returns

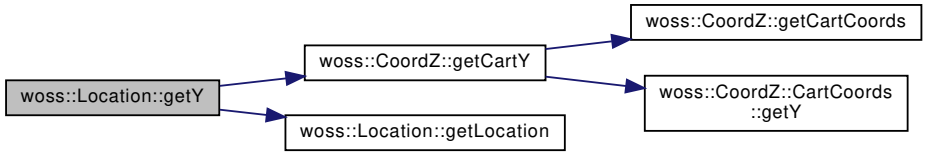
y value

Reimplemented in WossPosition.

References woss::CoordZ::getCartY(), and getLocation().

Referenced by WossPosition::getY().

Here is the call graph for this function:



13.31.3.13 getZ() `double Location::getZ ( ) [virtual]`

Gets current cartesian z-axis value

Returns

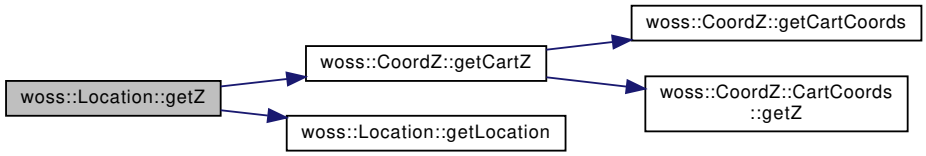
z value

Reimplemented in WossPosition.

References woss::CoordZ::getCartZ(), and getLocation().

Referenced by WossPosition::getZ().

Here is the call graph for this function:



13.31.3.14 isEquivalentTo() `bool Location::isEquivalentTo ( const woss::CoordZ & coordz ) [virtual]`

Checks if the woss::CoordZ given is equivalent to this Location

Parameters

coordz	valid woss::CoordZ to check

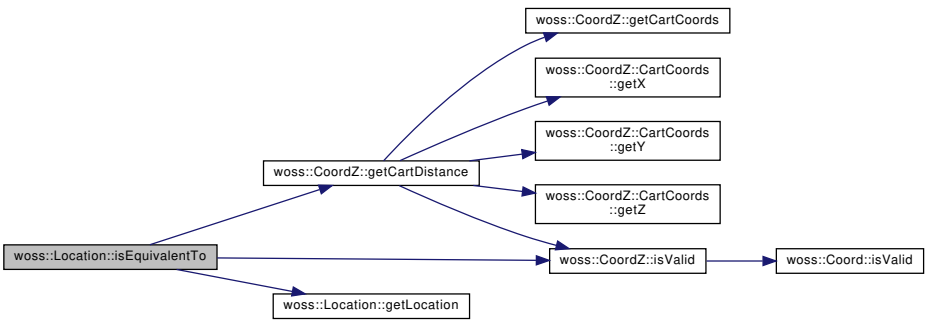
Returns

true if assumption is valid, *false* otherwise

Reimplemented in WossWpPosition.

References comparison_distance, woss::CoordZ::getCartDistance(), getLocation(), and woss::CoordZ::isValid().

Here is the call graph for this function:



13.31.3.15 isValid() `virtual bool woss::Location::isValid ( ) const [inline], [virtual]`

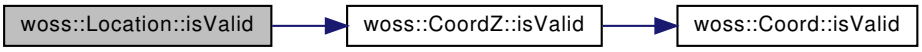
Checks the validity of Location

Returns

true if curr_coordz is valid, *false* otherwise

References curr_coordz, and woss::CoordZ::isValid().

Here is the call graph for this function:



13.31.3.16 operator<<() `friend::std::ostream & woss::Location::operator<< ( ::std::ostream & os, const Location & instance ) [inline]`

<< operator

Parameters

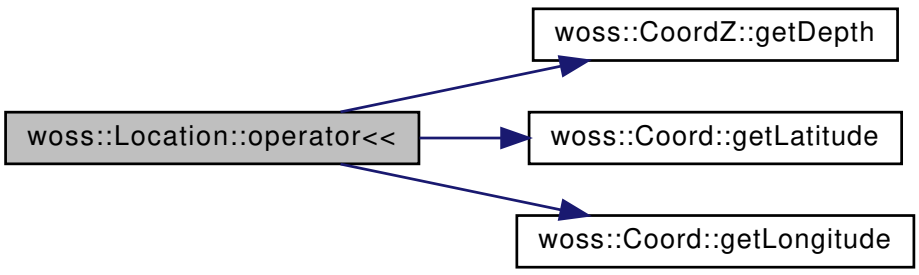
os	left operand ostream reference
instance	right operand const Location reference

Returns

os reference after the operation

References curr_coordz, woss::CoordZ::getDepth(), woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Here is the call graph for this function:



13.31.3.17 setDepth() `void Location::setDepth ( double depth ) [virtual]`

Sets initial depth

Parameters

lat	valid depth [m]

Reimplemented in WossPosition.

References curr_coordz, and woss::CoordZ::setDepth().

Referenced by WossPosition::setDepth().

Here is the call graph for this function:

13.31.3.18 setHorizontalOrientation() `void Location::setHorizontalOrientation ( double angle ) [virtual]`

Gets current horizontal orientation from reference line (0 degrees = parallel to direction of current movement) positive values are counter-clockwise rotations, while negative ones are clockwise rotations

Parameters

angle	delta angle [dec degrees]

References horizontal_orientation.

13.31.3.19 setLatitude() `void Location::setLatitude ( double lat ) [virtual]`

Sets initial latitude

Parameters

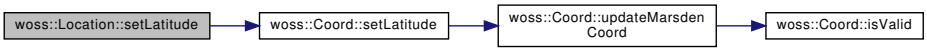
lat	valid latitude [decimal degrees]

Reimplemented in WossPosition.

References curr_coordz, and woss::Coord::setLatitude().

Referenced by WossPosition::setLatitude().

Here is the call graph for this function:



13.31.3.20 setLocation() void Location::setLocation (const CoordZ & *coordz*) [virtual]

Sets initial coordinates

Parameters

coordz	valid woss::CoordZ

References curr_coordz.

13.31.3.21 setLongitude() void Location::setLongitude (double *lon*) [virtual]

Sets initial longitude

Parameters

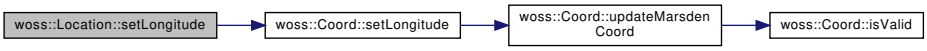
lat	valid longitude [decimal degrees]

Reimplemented in WossPosition.

References curr_coordz, and woss::Coord::setLongitude().

Referenced by WossPosition::setLongitude().

Here is the call graph for this function:



13.31.3.22 setVerticalOrientation() void Location::setVerticalOrientation (double *angle*) [virtual]

Sets current vertical orientation from reference line (0 degrees = parallel to sea surface / bottom). Negative values are towards the surface, while positive ones are towards sea bottom

Parameters

angle	delta angle [dec degrees]

References vertical_orientation.

13.31.4 Member Data Documentation

13.31.4.1 bearing `double woss::Location::bearing [protected]`

current bearing

Referenced by getBearing(), and WossWpPosition::getBearing().

13.31.4.2 comparison_distance `double woss::Location::comparison_distance [protected]`

Comparison distance [m]

Referenced by isEquivalentTo(), and WossWpPosition::isEquivalentTo().

13.31.4.3 curr_coordz `CoordZ woss::Location::curr_coordz [protected]`

current coordinates

Referenced by getLocation(), WossWpPosition::isEquivalentTo(), isValid(), operator<<(), setDepth(), setLatitude(), setLocation(), and setLongitude().

13.31.4.4 horizontal_orientation `double woss::Location::horizontal_orientation [protected]`

current horizontal orientation. It's the delta angle from reference direction (0 degrees = parallel to movement's direction)

Referenced by getHorizontalOrientation(), and setHorizontalOrientation().

13.31.4.5 vertical_orientation `double woss::Location::vertical_orientation [protected]`

current vertical orientation. It's the delta angle from reference direction (0 degrees = parallel to sea surface / bottom)

Referenced by getVerticalOrientation(), WossWpPosition::getVerticalOrientation(), and setVerticalOrientation().

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

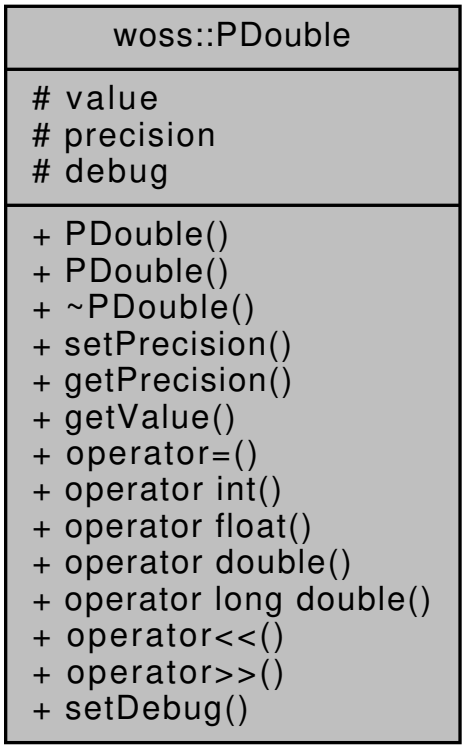
- woss/woss_def/location-definitions.h
- woss/woss_def/location-definitions.cpp

13.32 woss::PDouble Class Reference

Custom precision long double class.

`#include <custom-precision-double.h>`

Collaboration diagram for woss::PDouble:



Public Member Functions

- PDouble (const long double input=0.0, const long double precision=PDOUBLE_DEFAULT_PRECISION)
- PDouble (const PDouble &copy)
- ~PDouble ()
- void setPrecision (double value)
- long double getPrecision () const
- long double getValue () const
- PDouble & operator= (const PDouble &copy)
- operator int () const
- operator float () const
- operator double () const
- operator long double () const
- friend::std::ostream & operator<< (::std::ostream &os, const PDouble &instance)
- friend::std::istream & operator>> (::std::istream &is, PDouble &instance)

Static Public Member Functions

- static void setDebug (bool flag)

Protected Attributes

- long double value
- long double precision

Static Protected Attributes

- static bool debug = false

Friends

- const PDouble operator+ (const PDouble &left, const PDouble &right)
- const PDouble operator- (const PDouble &left, const PDouble &right)
- const PDouble operator/ (const PDouble &left, const PDouble &right)
- const PDouble operator∗ (const PDouble &left, const PDouble &right)
- const PDouble operator% (const PDouble &left, const PDouble &right)
- PDouble & operator+= (PDouble &left, const PDouble &right)
- PDouble & operator-= (PDouble &left, const PDouble &right)
- PDouble & operator/= (PDouble &left, const PDouble &right)
- PDouble & operator∗= (PDouble &left, const PDouble &right)
- PDouble & operator%= (PDouble &left, const PDouble &right)
- bool operator== (const PDouble &left, const PDouble &right)
- bool operator!= (const PDouble &left, const PDouble &right)
- bool operator> (const PDouble &left, const PDouble &right)
- bool operator< (const PDouble &left, const PDouble &right)
- bool operator>= (const PDouble &left, const PDouble &right)
- bool operator<= (const PDouble &left, const PDouble &right)

13.32.1 Detailed Description

Custom precision long double class.

The PDouble class stores a long double value and a long double precision for arithmetic computation and comparison purposes.

13.32.2 Constructor & Destructor Documentation

13.32.2.1 PDouble() [1/2]

```
PDouble::PDouble (
        const long double input = 0.0,
        const long double precision = PDOUBLE_DEFAULT_PRECISION )
```

PDouble constructor

Parameters

input	value to store
precision	custom precision

13.32.2.2 PDouble() [2/2] PDouble::PDouble (const PDouble & *copy*)

PDouble copy constructor

Parameters

copy	PDouble to be copied

References precision, and value.

13.32.2.3 ~PDouble() PDouble::~PDouble ( )

PDouble destructor. It is not **virtual**, since this class is not meant to be inherited from

13.32.3 Member Function Documentation

13.32.3.1 getPrecision() long double woss::PDouble::getPrecision ( ) const [inline]

Returns the custom precison

Returns

long double precision

References precision.

13.32.3.2 getValue() long double woss::PDouble::getValue ( ) const [inline]

Returns the custom value

Returns

long double value

References value.

13.32.3.3 operator double() `woss::PDouble::operator double ( ) const [inline]`

double cast operator

Returns

a copy of *value* casted to double

13.32.3.4 operator float() `woss::PDouble::operator float ( ) const [inline]`

float cast operator

Returns

a copy of *value* casted to float

13.32.3.5 operator int() `woss::PDouble::operator int ( ) const [inline]`

int cast operator

Returns

a copy of *value* casted to int

13.32.3.6 operator long double() `woss::PDouble::operator long double ( ) const [inline]`

long double cast operator

Returns

a copy of *value* casted to long double

13.32.3.7 operator<<() `friend::std::ostream & woss::PDouble::operator<< ( ::std::ostream & os, const PDouble & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const PDouble reference

Returns

os reference after the operation

13.32.3.8 operator=() `PDouble & PDouble::operator= ( const PDouble & copy )`

Assignment operator

Parameters

copy	const reference to a PDouble object to be copied

Returns

PDouble reference to *this*

References precision, and value.

13.32.3.9 operator>>() `friend::std::istream & woss::PDouble::operator>> ( ::std::istream & is, PDouble & instance )`

operator

Parameters

is	left operand istream reference
instance	right operand PDouble reference. It will take the value provided by *left* with default precision

Returns

is reference after the operation

13.32.3.10 setDebug() `static void woss::PDouble::setDebug ( bool flag ) [inline], [static]`

Sets debug for all instances

Parameters

flag	debug boolean

References debug.

13.32.3.11 setPrecision() void woss::PDouble::setPrecision (double *value*) [inline]

Sets a custom precison

Parameters

value	desired precision

References precision, and value.

13.32.4 Friends And Related Function Documentation

13.32.4.1 operator"!= bool operator!= (const PDouble & *left*, const PDouble & *right*) [friend]

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.32.4.2 operator% const PDouble operator% (const PDouble & *left*, const PDouble & *right*) [friend]

Modulo operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.32.4.3 operator%= `PDouble & operator%= ( PDouble & left, const PDouble & right ) [friend]`

Compound assignment modulo operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.32.4.4 operator* `const PDouble operator* ( const PDouble & left, const PDouble & right ) [friend]`

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.32.4.5 operator*= `PDouble & operator*= ( PDouble & left, const PDouble & right ) [friend]`

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.32.4.6 operator+

```
const PDouble operator+ (
    const PDouble & left,
    const PDouble & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.32.4.7 operator+=

```
PDouble & operator+= (
    PDouble & left,
    const PDouble & right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.32.4.8 operator- `const PDouble operator- ( const PDouble & left, const PDouble & right ) [friend]`

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.32.4.9 operator-= `PDouble & operator-= ( PDouble & left, const PDouble & right ) [friend]`

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.32.4.10 operator/ `const PDouble operator/ ( const PDouble & left, const PDouble & right ) [friend]`

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.32.4.11 operator/= PDouble & operator/= (PDouble & *left*, const PDouble & *right*) [friend]

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.32.4.12 operator< bool operator< (const PDouble & *left*, const PDouble & *right*) [friend]

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

13.32.4.13 operator<= bool operator<= (const PDouble & *left*, const PDouble & *right*) [friend]

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

13.32.4.14 operator== `bool operator== ( const PDouble & left, const PDouble & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

13.32.4.15 operator> `bool operator> ( const PDouble & left, const PDouble & right ) [friend]`

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

13.32.4.16 operator>= `bool operator>= ( const PDouble & left, const PDouble & right ) [friend]`

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

13.32.5 Member Data Documentation

13.32.5.1 debug `bool PDouble::debug = false [static], [protected]`

debug status

Referenced by setDebug().

13.32.5.2 precision `long double woss::PDouble::precision [protected]`

stored precision

Referenced by getPrecision(), operator=(), PDouble(), and setPrecision().

13.32.5.3 value `long double woss::PDouble::value [protected]`

stored value

Referenced by getValue(), operator=(), PDouble(), and setPrecision().

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

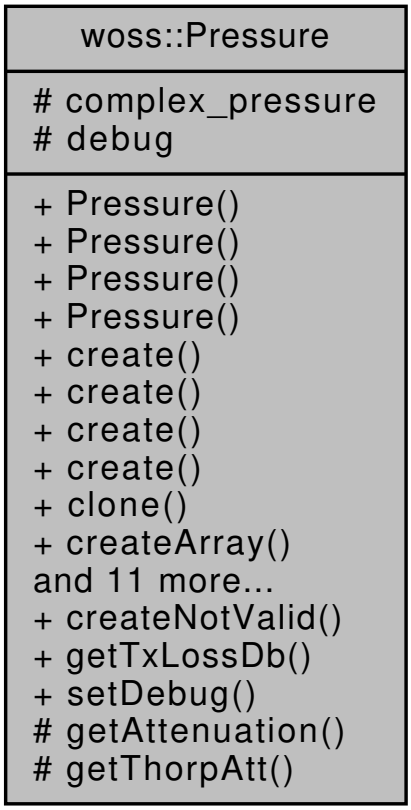
- woss/woss_def/custom-precision-double.h
- woss/woss_def/custom-precision-double.cpp

13.33 woss::Pressure Class Reference

Complex attenuated pressure class.

`#include <pressure-definitions.h>`

Collaboration diagram for woss::Pressure:



Public Member Functions

- Pressure (double real=0.0, double imag=0.0)
- Pressure (const ::std::complex< double > &complex_press)
- Pressure (const TimeArr &time_arr)
- Pressure (const Pressure &copy)
- virtual Pressure * create (double real=0.0, double imag=0.0) const
- virtual Pressure * create (const ::std::complex< double > &complex_press) const
- virtual Pressure * create (const TimeArr &time_arr) const
- virtual Pressure * create (const Pressure &copy) const
- virtual Pressure * clone () const
- virtual Pressure * createArray (unsigned int array_size) const
- operator::std::complex () const
- double real () const
- double imag () const
- double abs () const
- double phase () const
- Pressure sqrt () const
- virtual bool isValid () const
- virtual bool checkAttenuation (double distance, double frequency)
- Pressure & operator= (const Pressure &x)
- friend::std::ostream & operator<< (::std::ostream &os, const Pressure &instance)

Static Public Member Functions

- static const ::std::complex< double > createNotValid ()
- static double getTxLossDb (const ::std::complex< double > &val)
- static void setDebug (bool flag)

Protected Member Functions

- virtual double getAttenuation (double dist, double freq)
- double getThorpAtt (double frequency)

Protected Attributes

- ::std::complex< double > complex_pressure

Static Protected Attributes

- static bool debug = false

Friends

- const Pressure operator+ (const Pressure &left, const Pressure &right)
- const Pressure operator- (const Pressure &left, const Pressure &right)
- const Pressure operator/ (const Pressure &left, const Pressure &right)
- const Pressure operator* (const Pressure &left, const Pressure &right)
- Pressure & operator+= (Pressure &left, const Pressure &right)
- Pressure & operator-= (Pressure &left, const Pressure &right)
- Pressure & operator/= (Pressure &left, const Pressure &right)
- Pressure & operator*= (Pressure &left, const Pressure &right)
- bool operator== (const Pressure &left, const Pressure &right)
- bool operator!= (const Pressure &left, const Pressure &right)

13.33.1 Detailed Description

Complex attenuated pressure class.

The Pressure class stores a complex<double> acoustic pressure value for attenuation purposes. Pressure values are adimensional: complex pressure represented has to be considered in conjunction with transmitted pressure. Pressure also provides full arithmetic capability and a attenuation coherency check

13.33.2 Constructor & Destructor Documentation

13.33.2.1 Pressure() [1/4]

```
Pressure::Pressure (
        double real = 0.0,
        double imag = 0.0 )
```

Pressure constructor

Parameters

real	real part
imag	imaginary part

Referenced by clone(), and create().

13.33.2.2 Pressure() [2/4] woss::Pressure::Pressure (const ::std::complex< double > & *complex_press*)

Pressure constructor

Parameters

complex_press	complex<double> value

13.33.2.3 Pressure() [3/4] Pressure::Pressure (const TimeArr & *time_arr*)

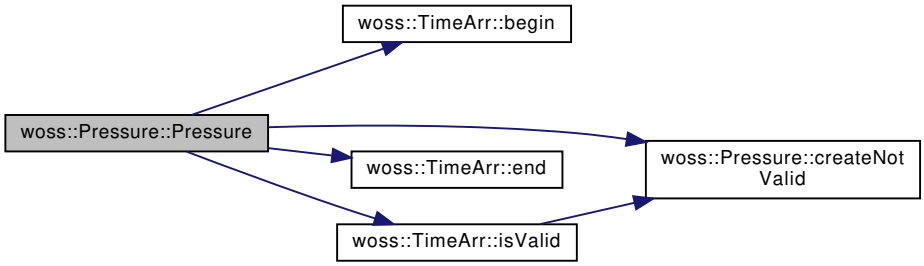
Constructs a Pressure from a TimeArr

Parameters

time_arr	valid TimeArr const reference

References woss::TimeArr::begin(), complex_pressure, createNotValid(), woss::TimeArr::end(), and woss::TimeArr::isValid().

Here is the call graph for this function:



13.33.2.4 Pressure() [4/4] Pressure::Pressure (const Pressure & *copy*)

Pressure copy constructor

Parameters

copy	Pressure to be copied

References complex_pressure.

13.33.3 Member Function Documentation

13.33.3.1 abs() `double woss::Pressure::abs ( ) const [inline]`

Gets the absolute value

Returns

double absolute value

References complex_pressure.

Referenced by WossMPropagation::computeGain(), and WossMPhyBpsk::getTxPower().

13.33.3.2 checkAttenuation() `bool Pressure::checkAttenuation ( double distance, double frequency ) [virtual]`

Checks if the attuenuation provided by the complex pressure is truly an attenuation, e.g. if abs < 1. If not, replace the pressure with the attenuation provided by Thorp absorption process at given frequency along the given distance

Parameters

distance	distance value in *meters*
frequency	frequency value in *hertz*

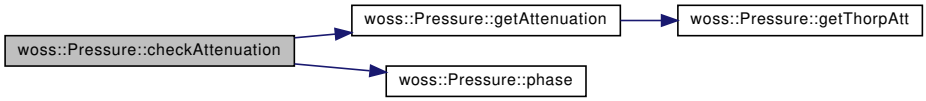
Returns

true if correction was applied, *false* otherwise

References complex_pressure, debug, getAttenuation(), and phase().

Referenced by woss::TimeArr::checkPressureAttenuation(), and WossMPropagation::computeGain().

Here is the call graph for this function:



13.33.3.3 clone() `virtual Pressure * woss::Pressure::clone ( ) const [inline], [virtual]`

Pressure virtual factory method

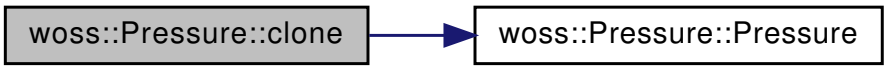
Returns

a heap-created copy of **this** instance

References Pressure().

Referenced by woss::DefHandler::operator=().

Here is the call graph for this function:



13.33.3.4 create() [1/4] `virtual Pressure * woss::Pressure::create ( const ::std::complex< double > & complex_press ) const [inline], [virtual]`

Pressure virtual factory method

Parameters

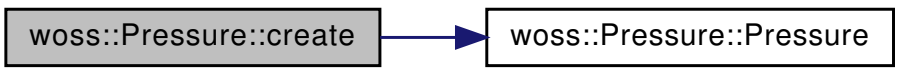
complex_press	complex<double> value

Returns

a heap-created Pressure object

References Pressure().

Here is the call graph for this function:



13.33.3.5 create() [2/4] `virtual Pressure * woss::Pressure::create ( const Pressure & copy ) const [inline], [virtual]`

Pressure virtual factory method

Parameters

copy	Pressure to be copied

Returns

a heap-created Pressure object

References Pressure().

Here is the call graph for this function:

13.33.3.6 create() [3/4]

```
virtual Pressure * woss::Pressure::create (
    const TimeArr & time_arr ) const  [inline], [virtual]
```

Pressure virtual factory method

Parameters

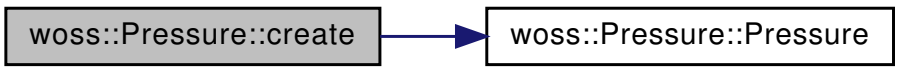
time_arr	valid TimeArr const reference

Returns

a heap-created Pressure object

References Pressure().

Here is the call graph for this function:



13.33.3.7 create() [4/4]

```
virtual Pressure * woss::Pressure::create (
    double real = 0.0,
    double imag = 0.0 ) const  [inline], [virtual]
```

Pressure virtual factory method

Parameters

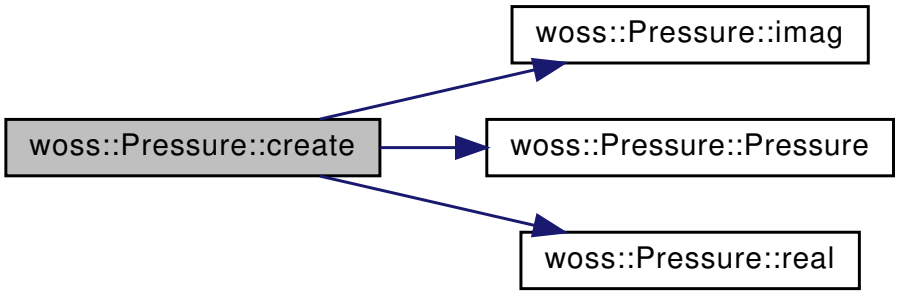
real	real part
imag	imaginary part

Returns

a heap-created Pressure object

References imag(), Pressure(), and real().

Here is the call graph for this function:



13.33.3.8 createArray()

`virtual Pressure * woss::Pressure::createArray ( unsigned int array_size ) const [inline], [virtual]`

Pressure virtual factory method

Parameters

array_size	size of array

Returns

a heap-created array of size array_size

13.33.3.9 createNotValid()

`static const ::std::complex< double > woss::Pressure::createNotValid ( ) [inline], [static]`

Creates an instance not valid

Returns

an instance not valid (e.g. (+inf, +inf))

Referenced by woss::TimeArr::createNotValid(), woss::WossManagerResDb::dbGetPressure(), woss::WossDbManager::getPressure
woss::ResPressureTxtDb::getValue(), woss::WossManager::getWossPressure(), woss::TimeArr::isValid(),
Pressure(), woss::ArrAscResReader::readAvgPressure(), woss::ArrBinResReader::readAvgPressure(), woss::ShdResReader::readA
woss::ResPressureTxtDb::readMap(), woss::ShdResReader::readPressure(), woss::ShdResReader::readTimeArr(),
and woss::TimeArr::TimeArr().

13.33.3.10 getAttenuation()

```
double Pressure::getAttenuation (
    double dist,
    double freq )  [protected], [virtual]
```

Gets the linear acoustic attenuation incurred at given frequency along the given distance

Parameters

dist	distance in *meters*
freq	frequency in *hertz*

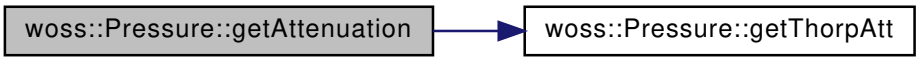
Returns

linear attenuation

References getThorpAtt().

Referenced by checkAttenuation(), and WossMPropagation::computeGain().

Here is the call graph for this function:



13.33.3.11 getThorpAtt()

```
double Pressure::getThorpAtt (
    double frequency )  [protected]
```

Gets the acoustic attenuation (in db re uPa / m) given by Thorp absorption process incurred at given frequency

Parameters

freq	frequency in *hertz*

Returns

attenuation in *db re uPa / m*

Referenced by getAttenuation().

13.33.3.12 getTxLossDb()

```
static double woss::Pressure::getTxLossDb (
    const ::std::complex< double > & val ) [inline], [static]
```

Returns the Transmission Loss measured in db

Returns

the Transmission Loss [db]

Referenced by woss::ArrBinResReader::getArrBinFile(), and woss::TimeArr::lowerBoundTxLoss().

13.33.3.13 imag()

```
double woss::Pressure::imag ( ) const [inline]
```

Gets the imaginary part

Returns

double imaginary part

References complex_pressure.

Referenced by create().

13.33.3.14 isValid()

```
virtual bool woss::Pressure::isValid ( ) const [inline], [virtual]
```

Checks the validity of complex pressure provided

Returns

true if complex pressure is valid, *false* otherwise

References complex_pressure.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::TimeArr::insertValue(), woss::TimeArr::sumValue(), and woss::TimeArr::TimeArr().

13.33.3.15 operator::std::complex()

```
woss::Pressure::operator::std::complex ( ) const [inline]
```

complex<double> operator for implicit casting

Returns

a copy of complex_pressure

13.33.3.16 operator<<()

```
friend::std::ostream & woss::Pressure::operator<< (
    ::std::ostream & os,
    const Pressure & instance )
```

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Pressure reference

Returns

os reference after the operation

13.33.3.17 operator=() `Pressure & Pressure::operator= ( const Pressure & x )`

Assignment operator

Parameters

copy	const reference to a Pressure object to be copied

Returns

Pressure reference to *this*

References complex_pressure.

13.33.3.18 phase() `double woss::Pressure::phase ( ) const [inline]`

Gets the phase (arg) value

Returns

double phase (arg) value

References complex_pressure.

Referenced by checkAttenuation().

13.33.3.19 real() `double woss::Pressure::real ( ) const [inline]`

Gets the real part

Returns

double real part

References complex_pressure.

Referenced by create().

13.33.3.20 setDebug() static void woss::Pressure::setDebug (bool *flag*) [inline], [static]

Sets debug for the whole class

Parameters

flag	debug value

References debug.

13.33.3.21 sqrt() `Pressure woss::Pressure::sqrt ( ) const [inline]`

Gets the square root of a complex number

Returns

double square root value

References complex_pressure.

13.33.4 Friends And Related Function Documentation

13.33.4.1 operator"!= `bool operator!= ( const Pressure & left, const Pressure & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.33.4.2 operator* `const Pressure operator* ( const Pressure & left, const Pressure & right ) [friend]`

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.33.4.3 operator*= Pressure & operator*= (Pressure & *left*, const Pressure & *right*) [friend]

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.33.4.4 operator+ const Pressure operator+ (const Pressure & *left*, const Pressure & *right*) [friend]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.33.4.5 operator+= Pressure & operator+= (Pressure & *left*, const Pressure & *right*) [friend]

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.33.4.6 operator- `const Pressure operator- ( const Pressure & left, const Pressure & right ) [friend]`

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.33.4.7 operator-= `Pressure & operator-= ( Pressure & left, const Pressure & right ) [friend]`

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.33.4.8 operator/

```
const Pressure operator/ (
    const Pressure & left,
    const Pressure & right )  [friend]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.33.4.9 operator/=

```
Pressure & operator/= (
    Pressure & left,
    const Pressure & right )  [friend]
```

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.33.4.10 operator==

```
bool operator== (
    const Pressure & left,
    const Pressure & right )  [friend]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.33.5 Member Data Documentation

13.33.5.1 complex_pressure ::std::complex< double > woss::Pressure::complex_pressure [protected]

Complex acoustic pressure

Referenced by abs(), checkAttenuation(), imag(), isValid(), operator=(), phase(), Pressure(), real(), and sqrt().

13.33.5.2 debug bool Pressure::debug = false [static], [protected]

Debug flag

Referenced by checkAttenuation(), and setDebug().

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

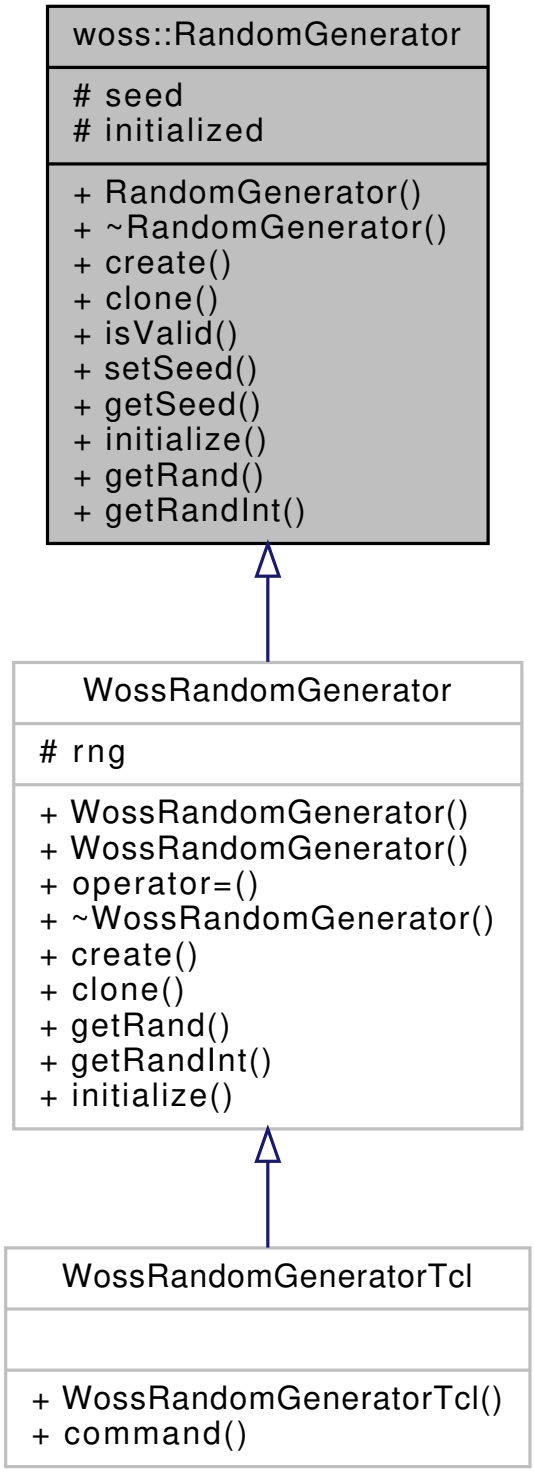
- woss/woss_def/pressure-definitions.h
- woss/woss_def/pressure-definitions.cpp

13.34 woss::RandomGenerator Class Reference

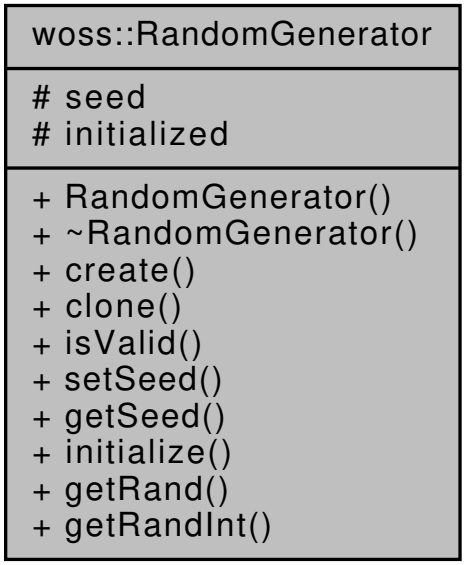
woss::RandomGenerator class

```
#include <random-generator-definitions.h>
```

Inheritance diagram for woss::RandomGenerator:



Collaboration diagram for woss::RandomGenerator:



Public Member Functions

- RandomGenerator (int s=0)
- virtual ~RandomGenerator ()
- virtual RandomGenerator * create (double seed)
- virtual RandomGenerator * clone () const
- virtual bool isValid () const
- virtual void setSeed (int s)
- virtual int getSeed () const
- virtual void initialize ()
- virtual double getRand () const
- virtual int getRandInt () const

Protected Attributes

- int seed
- bool initialized

13.34.1 Detailed Description

woss::RandomGenerator class

Class for random value generation purposes

13.34.2 Constructor & Destructor Documentation

13.34.2.1 RandomGenerator() `woss::RandomGenerator::RandomGenerator ( int s = 0 ) [inline]`

Default RandomGenerator constructor

Parameters

s	seed

Referenced by clone(), and create().

13.34.2.2 ~RandomGenerator() `virtual woss::RandomGenerator::~RandomGenerator ( ) [inline], [virtual]`

Default destructor

13.34.3 Member Function Documentation

13.34.3.1 clone() `virtual RandomGenerator * woss::RandomGenerator::clone ( ) const [inline], [virtual]`

RandomGenerator virtual factory method

Returns

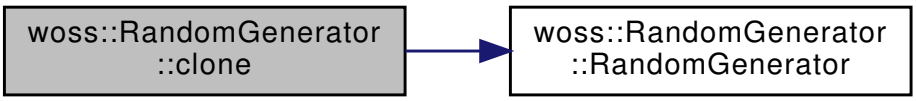
a heap-allocated RandomGenerator copy of **this** instance

Reimplemented in WossRandomGenerator.

References RandomGenerator().

Referenced by woss::DefHandler::operator=().

Here is the call graph for this function:



13.34.3.2 create() `virtual RandomGenerator * woss::RandomGenerator::create ( double seed ) [inline], [virtual]`

RandomGenerator virtual factory method

Parameters

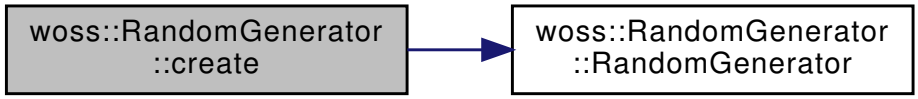
copy	RandomGenerator to be copied

Returns

a heap-allocated RandomGenerator object

References RandomGenerator(), and seed.

Here is the call graph for this function:



13.34.3.3 getRand() `double RandomGenerator::getRand ( ) const [virtual]`

Gets a random value

Returns

a random value between 0 and 1

Reimplemented in WossRandomGenerator.

References initialized.

Referenced by woss::DefHandler::getRand().

13.34.3.4 getRandInt() `int RandomGenerator::getRandInt ( ) const [virtual]`

Gets a random integer value

Returns

a random integer

Reimplemented in WossRandomGenerator.

References initialized.

Referenced by woss::DefHandler::getRandInt().

13.34.3.5 getSeed() `virtual int woss::RandomGenerator::getSeed ( ) const [inline], [virtual]`

Sets the seed

Returns

the seed

References seed.

13.34.3.6 initialize() `void RandomGenerator::initialize ( ) [virtual]`

Mandatory function to initialize the instance

Reimplemented in WossRandomGenerator.

References initialized, and seed.

13.34.3.7 isValid() `virtual bool woss::RandomGenerator::isValid ( ) const [inline], [virtual]`

Checks the validity of RandomGenerator

Returns

true if initialized, *false* otherwise

References initialized.

13.34.3.8 setSeed() `virtual void woss::RandomGenerator::setSeed ( int s ) [inline], [virtual]`

Sets the seed

Parameters

s	seed

References seed.

13.34.4 Member Data Documentation

13.34.4.1 initialized `bool woss::RandomGenerator::initialized [protected]`

true if initialize() has been called, false otherwise

Referenced by getRand(), WossRandomGenerator::getRand(), getRandInt(), WossRandomGenerator::getRandInt(), initialize(), WossRandomGenerator::initialize(), and isValid().

13.34.4.2 seed `int woss::RandomGenerator::seed [protected]`

seed value

Referenced by create(), getSeed(), initialize(), WossRandomGenerator::initialize(), and setSeed().

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

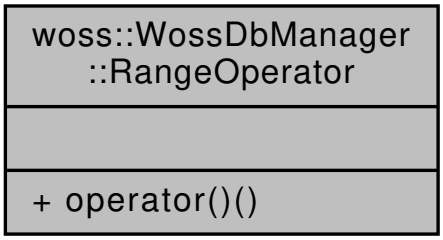
- woss/woss_def/random-generator-definitions.h
- woss/woss_def/random-generator-definitions.cpp

13.35 woss::WossDbManager::RangeOperator Class Reference

Range operator function object.

`#include <woss-db-manager.h>`

Collaboration diagram for woss::WossDbManager::RangeOperator:



Public Member Functions

- double operator() (const Coord &x, const Coord &y) const

13.35.1 Detailed Description

Range operator function object.

Function object that returns the great circle distance between two valid woss::Coord

13.35.2 Member Function Documentation

13.35.2.1 operator()() `double woss::WossDbManager::RangeOperator::operator() ( const Coord & x, const Coord & y ) const [inline]`

Function that compares to woss::Coord instances. If WossDbManager::cust_bathymetry_coord_resolution is valid (>=0) two valid CoordZ are considered equivalent if their great circle distance is less or equal to the space sampling value

Parameters

tx	const reference to a valid Coord object
rx	const reference to a valid Coord object

Returns

true if x less than y, *false* otherwise

References woss::Coord::getGreatCircleDistance().

Here is the call graph for this function:

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

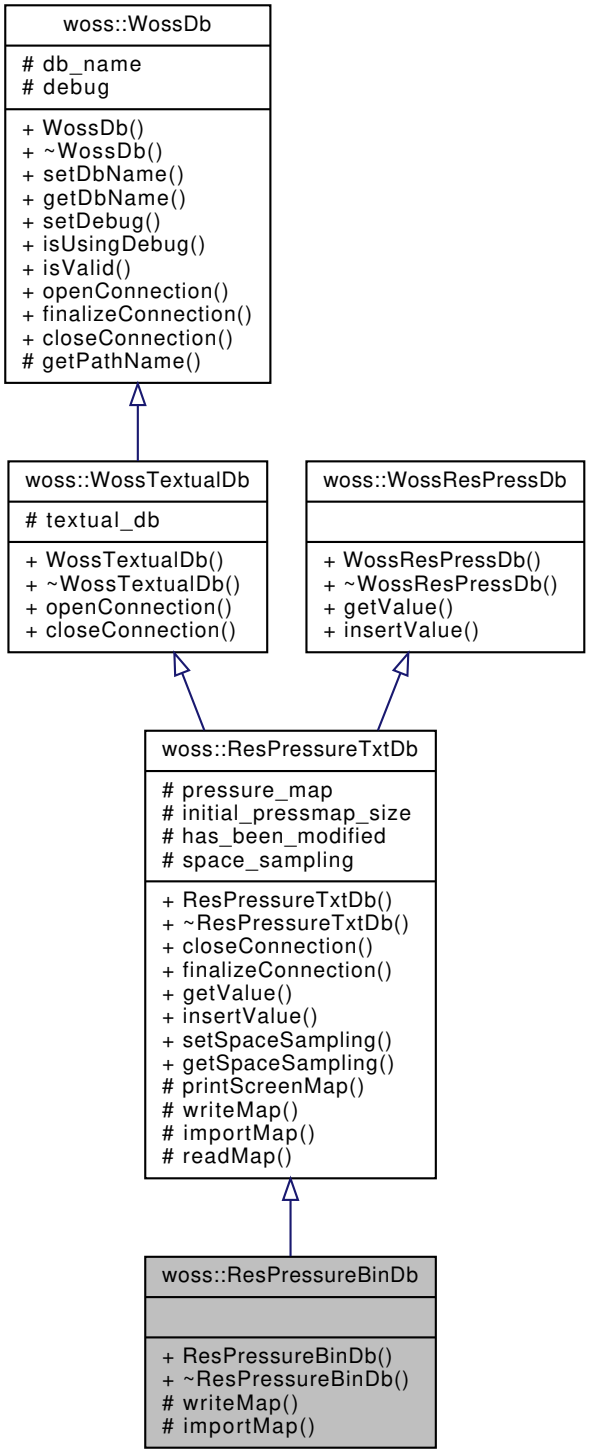
- woss/woss_db/woss-db-manager.h

13.36 woss::ResPressureBinDb Class Reference

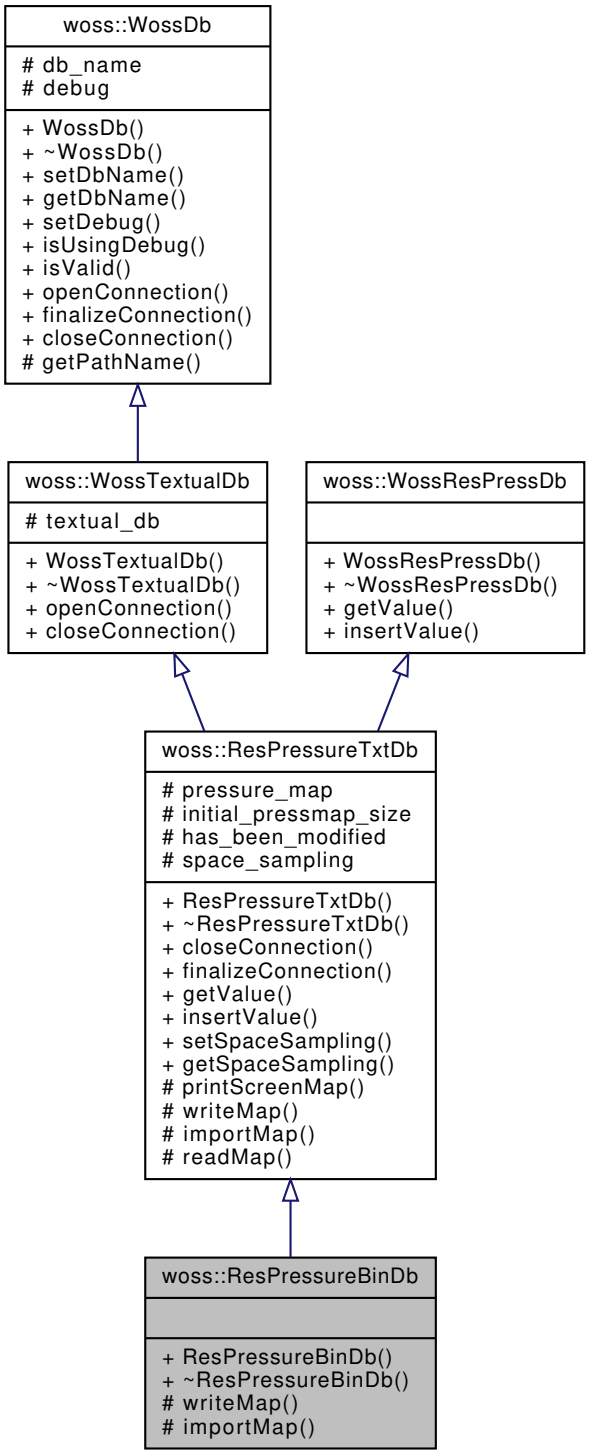
Binary WossDb for Pressure.

`#include <res-pressure-bin-db.h>`

Inheritance diagram for woss::ResPressureBinDb:



Collaboration diagram for woss::ResPressureBinDb:



Public Member Functions

- ResPressureBinDb (const ::std::string &name)

Protected Member Functions

- virtual bool writeMap ()
- virtual bool importMap ()

Additional Inherited Members

13.36.1 Detailed Description

Binary WossDb for Pressure.

ResPressureBinDb implements WossTextualDb and WossResPressDb for storing calculated Pressure into a binary file

13.36.2 Constructor & Destructor Documentation

13.36.2.1 ResPressureBinDb() `woss::ResPressureBinDb::ResPressureBinDb ( const ::std::string & name ) [inline]`

ResPressureBinDb constructor

Parameters

name	pathname of database

13.36.3 Member Function Documentation

13.36.3.1 importMap() `bool ResPressureBinDb::importMap ( ) [protected], [virtual]`

Imports the formatted binary file into pressure_map. The format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, real pressure, imag pressure

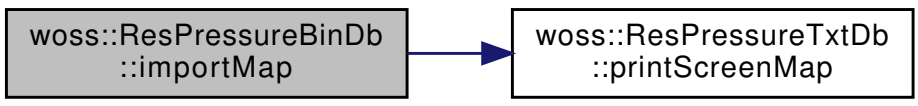
Returns

true if operation succeeds, *false* otherwise

Reimplemented from woss::ResPressureTxtDb.

References woss::WossDb::db_name, woss::WossDb::debug, woss::ResPressureTxtDb::initial_pressmap_size, woss::ResPressureTxtDb::pressure_map, woss::ResPressureTxtDb::printScreenMap(), and woss::WossTextualDb::textual_db.

Here is the call graph for this function:



13.36.3.2 writeMap() `bool ResPressureBinDb::writeMap ( ) [protected], [virtual]`

Writes pressure_map to binary file. The format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, real pressure, imag pressure

Returns

true if operation succeeds, *false* otherwise

Reimplemented from woss::ResPressureTxtDb.

References woss::WossDb::db_name, woss::ResPressureTxtDb::pressure_map, and woss::WossTextualDb::textual_db.

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

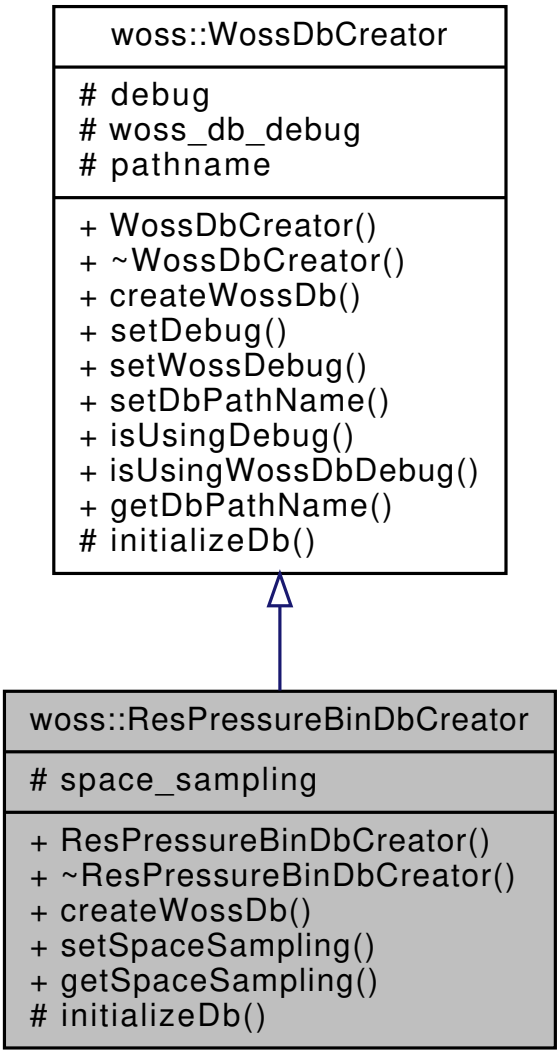
- woss/woss_db/res-pressure-bin-db.h
- woss/woss_db/res-pressure-bin-db.cpp

13.37 woss::ResPressureBinDbCreator Class Reference

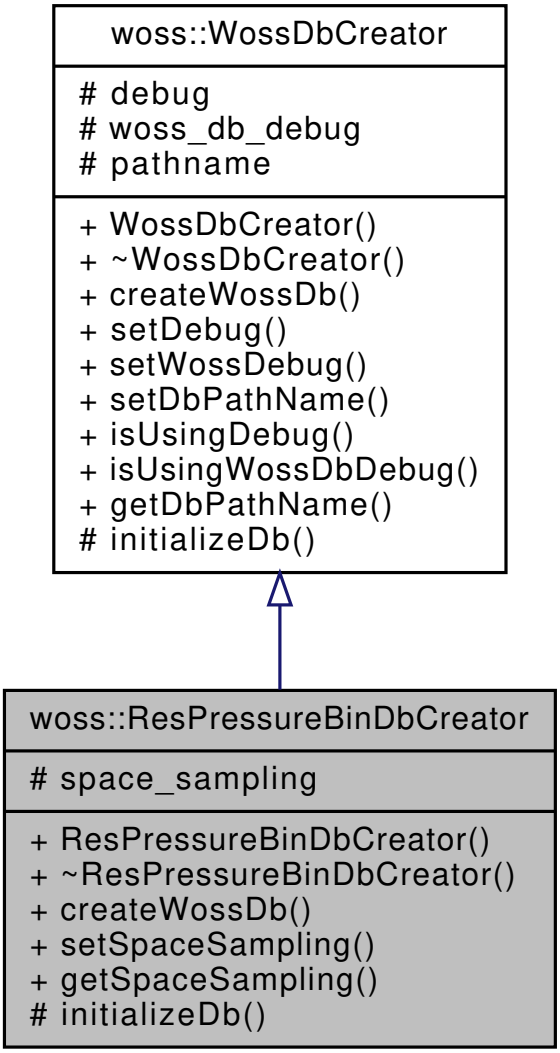
DbCreator for binary Pressure database.

`#include <res-pressure-bin-db-creator.h>`

Inheritance diagram for woss::ResPressureBinDbCreator:



Collaboration diagram for woss::ResPressureBinDbCreator:



Public Member Functions

- ResPressureBinDbCreator ()
- virtual WossDb ∗const createWossDb ()
- void **setSpaceSampling** (double value)
- double **getSpaceSampling** ()

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- double **space_sampling**

13.37.1 Detailed Description

DbCreator for binary Pressure database.

ResPressureBinDbCreator implements WossDbCreator for binary file Pressure database

13.37.2 Constructor & Destructor Documentation

13.37.2.1 ResPressureBinDbCreator() `ResPressureBinDbCreator::ResPressureBinDbCreator ( )`

ResPressureBinDbCreator default constructor

13.37.3 Member Function Documentation

13.37.3.1 createWossDb() `WossDb *const ResPressureBinDbCreator::createWossDb ( ) [virtual]`

This method is called to create and initialize a ResPressureTxtDb

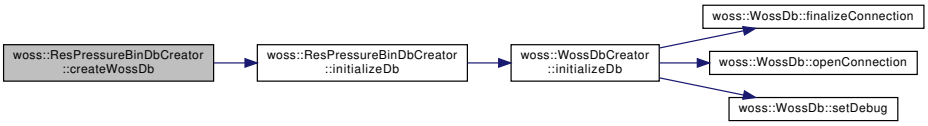
Returns

a pointer to a properly initialized ResPressureTxtDb object

Implements woss::WossDbCreator.

References initializeDb(), and woss::WossDbCreator::pathname.

Here is the call graph for this function:



13.37.3.2 initializeDb() `bool ResPressureBinDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created ResPressureTxtDb

Returns

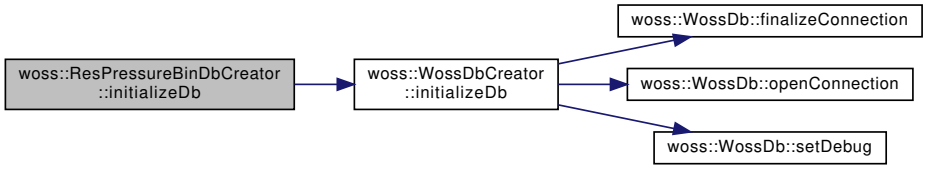
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



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

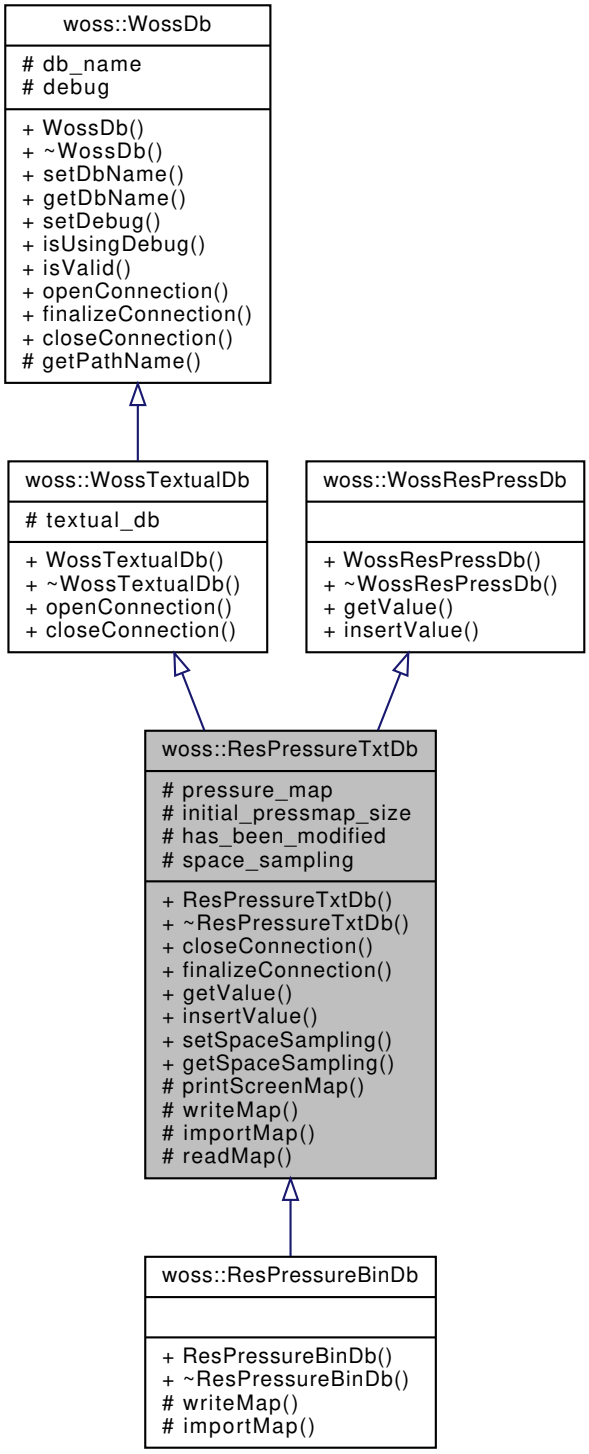
- woss/woss_db/res-pressure-bin-db-creator.h
- woss/woss_db/res-pressure-bin-db-creator.cpp

13.38 woss::ResPressureTxtDb Class Reference

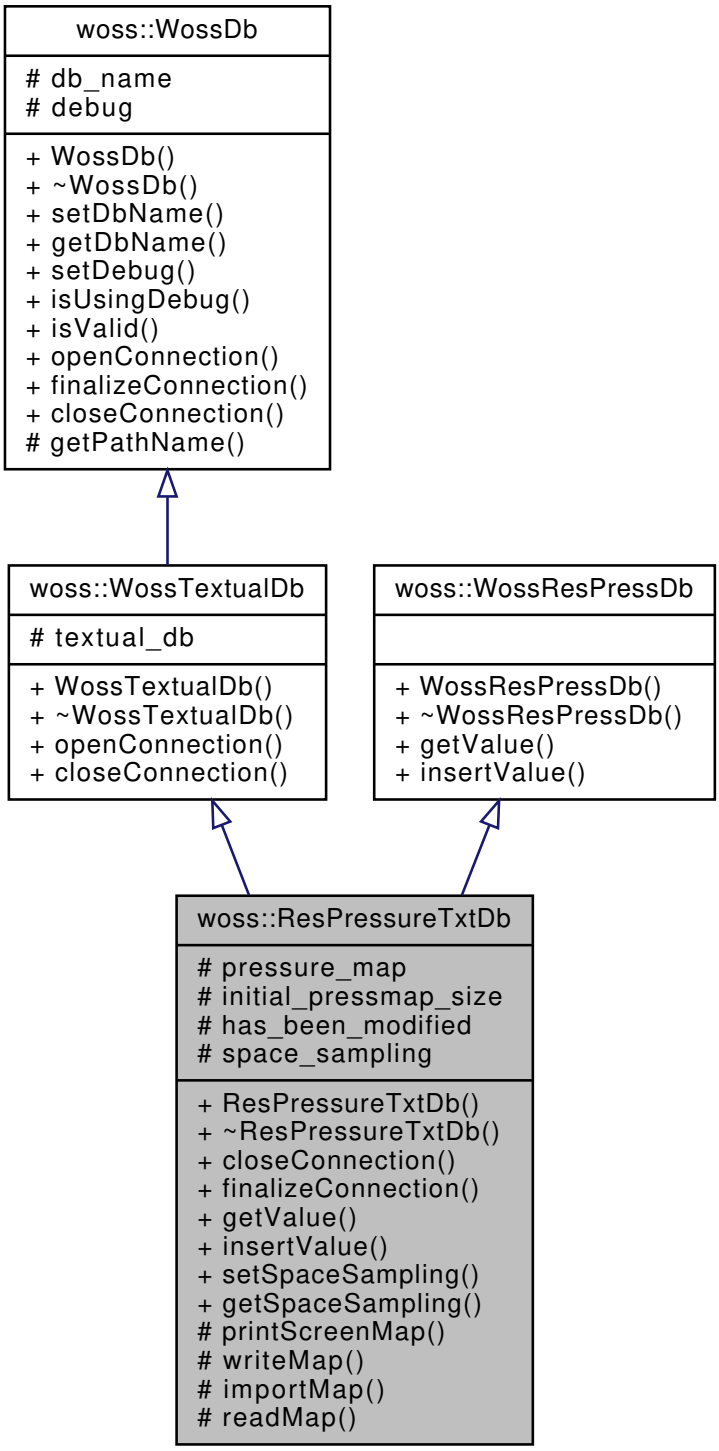
Textual WossDb for Pressure.

```
#include <res-pressure-txt-db.h>
```

Inheritance diagram for woss::ResPressureTxtDb:



Collaboration diagram for woss::ResPressureTxtDb:



Public Member Functions

- ResPressureTxtDb (const ::std::string &name)
- virtual bool closeConnection ()
- virtual bool finalizeConnection ()
- virtual Pressure * getValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const
- virtual bool insertValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const Pressure &pressure)

Static Public Member Functions

- static void **setSpaceSampling** (double value)
- static double **getSpaceSampling** ()

Protected Types

- typedef ::std::map< time_t, ::std::complex< double > > **TimeMap**
- typedef TimeMap::iterator **TMIter**
- typedef TimeMap::const_iterator **TMCIter**
- typedef TimeMap::reverse_iterator **TMRIter**
- typedef ::std::map< PDouble, TimeMap > **FreqMap**
- typedef FreqMap::iterator **FMIter**
- typedef FreqMap::reverse_iterator **FMRIter**
- typedef ::std::map< CoordZ, FreqMap, CoordComparator< ResPressureTxtDb, CoordZ > > **RxMap**
- typedef RxMap::iterator **RxMIter**
- typedef RxMap::reverse_iterator **RxMRIter**
- typedef ::std::map< CoordZ, RxMap, CoordComparator< ResPressureTxtDb, CoordZ > > PressureMatrix
- typedef PressureMatrix::iterator **PMIter**
- typedef PressureMatrix::const_iterator **PMCIter**
- typedef PressureMatrix::reverse_iterator **PMRIter**
- typedef PressureMatrix::const_reverse_iterator **PMCRIter**

Protected Member Functions

- void printScreenMap ()
- virtual bool writeMap ()
- virtual bool importMap ()
- ::std::complex< double > readMap (const CoordZ &tx, const CoordZ &rx, const double frequency, const Time &time_value) const

Protected Attributes

- PressureMatrix pressure_map
- int initial_pressmap_size
- bool **has_been_modified**

Static Protected Attributes

- static double **space_sampling** = 0.0

13.38.1 Detailed Description

Textual WossDb for Pressure.

ResPressureTxtDb implements WossTextualDb and WossResPressDb for storing calculated Pressure into a text file

13.38.2 Member Typedef Documentation

13.38.2.1 PressureMatrix `typedef ::std::map< CoordZ, RxMap, CoordComparator< ResPressureTxtDb, CoordZ > > woss::ResPressureTxtDb::PressureMatrix [protected]`

Multidimensional map that links a transmitter CoordZ to a receiver CoordZ to a frequency PDouble value and finally to a Pressure value

13.38.3 Constructor & Destructor Documentation

13.38.3.1 ResPressureTxtDb() `ResPressureTxtDb::ResPressureTxtDb ( const ::std::string & name )`

ResPressureTxtDb constructor

Parameters

name	pathname of database

13.38.4 Member Function Documentation

13.38.4.1 closeConnection() `bool ResPressureTxtDb::closeConnection ( ) [virtual]`

Closes the connection to the text file provided

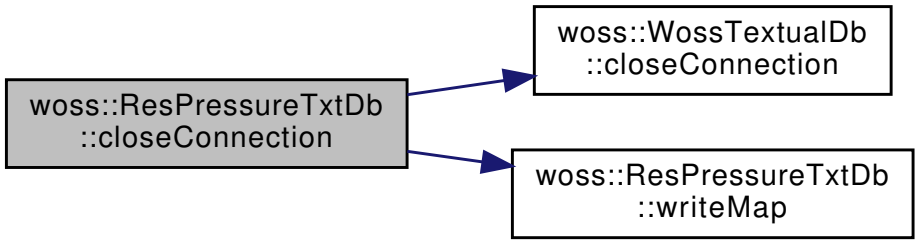
Returns

true if method was successful, *false* otherwise

Reimplemented from woss::WossTextualDb.

References woss::WossTextualDb::closeConnection(), and writeMap().

Here is the call graph for this function:



13.38.4.2 finalizeConnection() `bool ResPressureTxtDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions

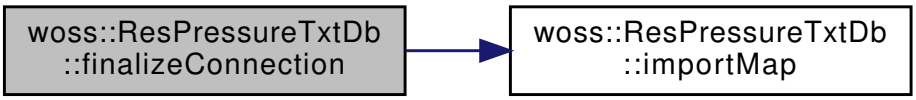
Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References importMap().

Here is the call graph for this function:



13.38.4.3 getValue()

```
Pressure * ResPressureTxtDb::getValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [virtual]
```

Returns a pointer to a heap-created Pressure for given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object

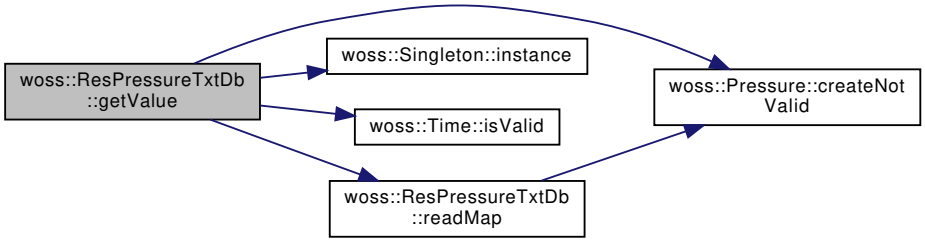
Returns

valid Pressure if parameters are found, *not valid* otherwise

Implements woss::WossResPressDb.

References woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), woss::Time::isValid(), pressure_map, and readMap().

Here is the call graph for this function:



13.38.4.4 importMap() `bool ResPressureTxtDb::importMap ( ) [protected], [virtual]`

Imports the formatted textual files into pressure_map. The column format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, real pressure, imag pressure

Returns

true if operation succeeds, *false* otherwise

Reimplemented in woss::ResPressureBinDb.

References woss::WossDb::db_name, and woss::WossTextualDb::textual_db.

Referenced by finalizeConnection().

13.38.4.5 insertValue()

```
bool ResPressureTxtDb::insertValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value,
    const Pressure & pressure ) [virtual]
```

Inserts the given Pressure value in the pressure_map at given frequency, transmitter and receiver coordinates

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object
pressure	computed Pressure

Implements woss::WossResPressDb.

References woss::WossDb::debug, and pressure_map.

13.38.4.6 printScreenMap() `void ResPressureTxtDb::printScreenMap ( ) [protected]`

Prints pressure_map to screen. The columns format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, real pressure, imag pressure

Returns

true if operation succeeds, *false* otherwise

Referenced by woss::ResPressureBinDb::importMap().

13.38.4.7 readMap() `std::complex< double > ResPressureTxtDb::readMap ( const CoordZ & tx, const CoordZ & rx, const double frequency, const Time & time_value ) const [protected]`

Reads given values from pressure_map

Parameters

tx	valid transmitter coordinates
rx	valid receiver coordinates
frequency	frequency [hz]
time_value	const reference to a valid time_value

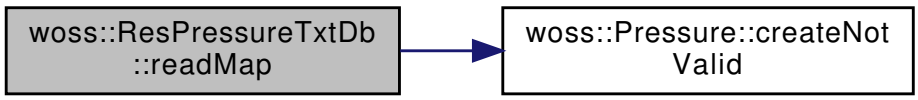
Returns

valid Pressure if parameters are found, not valid otherwise

References woss::Pressure::createNotValid(), and pressure_map.

Referenced by getValue().

Here is the call graph for this function:



13.38.4.8 writeMap() `bool ResPressureTxtDb::writeMap ( ) [protected], [virtual]`

Writes pressure_map to textual files. The columns format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, real pressure, imag pressure

Returns

true if operation succeeds, *false* otherwise

Reimplemented in woss::ResPressureBinDb.

References woss::WossDb::db_name, and woss::WossTextualDb::textual_db.

Referenced by closeConnection().

13.38.5 Member Data Documentation

13.38.5.1 initial_pressmap_size `int woss::ResPressureTxtDb::initial_pressmap_size [protected]`

pressure_map's initial size. If pressure_map's size is greater on closeConnection() the map will be written to disk

Referenced by woss::ResPressureBinDb::importMap().

13.38.5.2 pressure_map `PressureMatrix woss::ResPressureTxtDb::pressure_map [protected]`

PressureMatrix map for storing imported and user inserted Pressure values

Referenced by getValue(), woss::ResPressureBinDb::importMap(), insertValue(), readMap(), and woss::ResPressureBinDb::writeMap

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

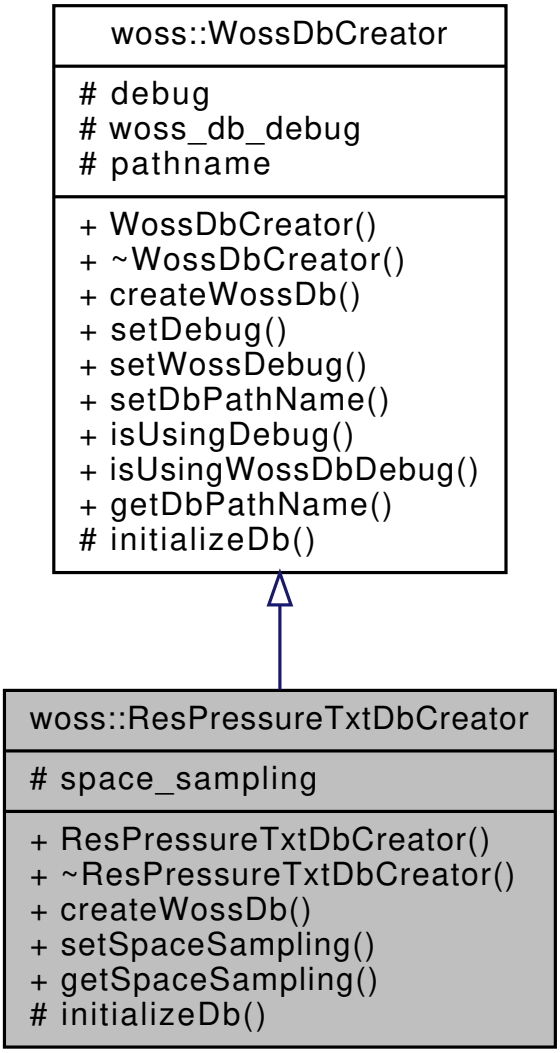
- woss/woss_db/res-pressure-txt-db.h
- woss/woss_db/res-pressure-txt-db.cpp

13.39 woss::ResPressureTxtDbCreator Class Reference

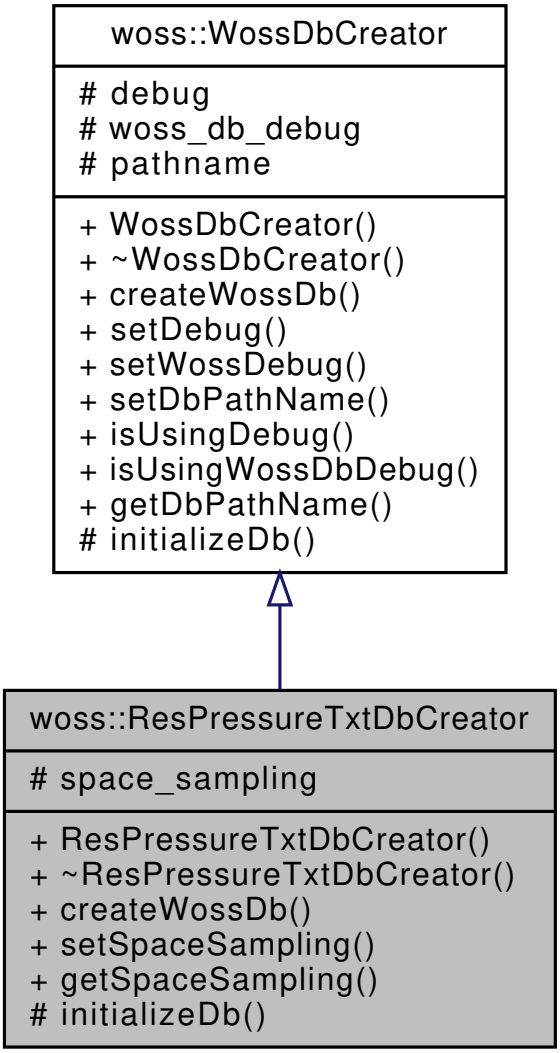
DbCreator for textual Pressure database.

`#include <res-pressure-txt-db-creator.h>`

Inheritance diagram for woss::ResPressureTxtDbCreator:



Collaboration diagram for woss::ResPressureTxtDbCreator:



Public Member Functions

- ResPressureTxtDbCreator ()
- virtual WossDb ∗const createWossDb ()
- void **setSpaceSampling** (double value)
- double **getSpaceSampling** ()

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- double **space_sampling**

13.39.1 Detailed Description

DbCreator for textual Pressure database.

ResPressureTxtDbCreator implements WossDbCreator for textual Pressure database

13.39.2 Constructor & Destructor Documentation

13.39.2.1 ResPressureTxtDbCreator() `ResPressureTxtDbCreator::ResPressureTxtDbCreator ( )`

ResPressureTxtDbCreator default constructor

13.39.3 Member Function Documentation

13.39.3.1 createWossDb() `WossDb *const ResPressureTxtDbCreator::createWossDb ( ) [virtual]`

This method is called to create and initialize a ResPressureTxtDb

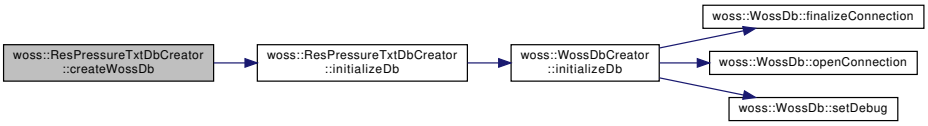
Returns

a pointer to a properly initialized ResPressureTxtDb object

Implements woss::WossDbCreator.

References initializeDb(), and woss::WossDbCreator::pathname.

Here is the call graph for this function:



13.39.3.2 initializeDb() `bool ResPressureTxtDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created ResPressureTxtDb

Returns

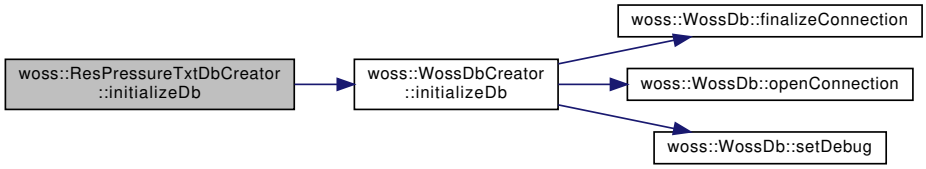
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



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

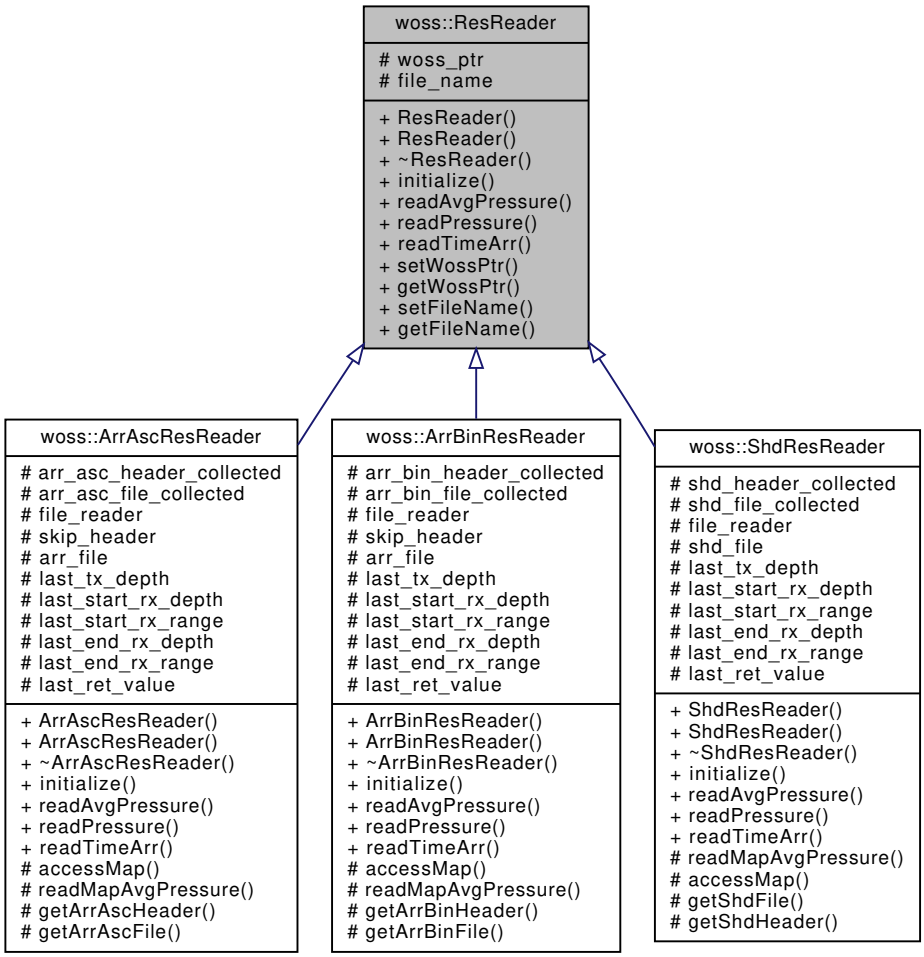
- woss/woss_db/res-pressure-txt-db-creator.h
- woss/woss_db/res-pressure-txt-db-creator.cpp

13.40 woss::ResReader Class Reference

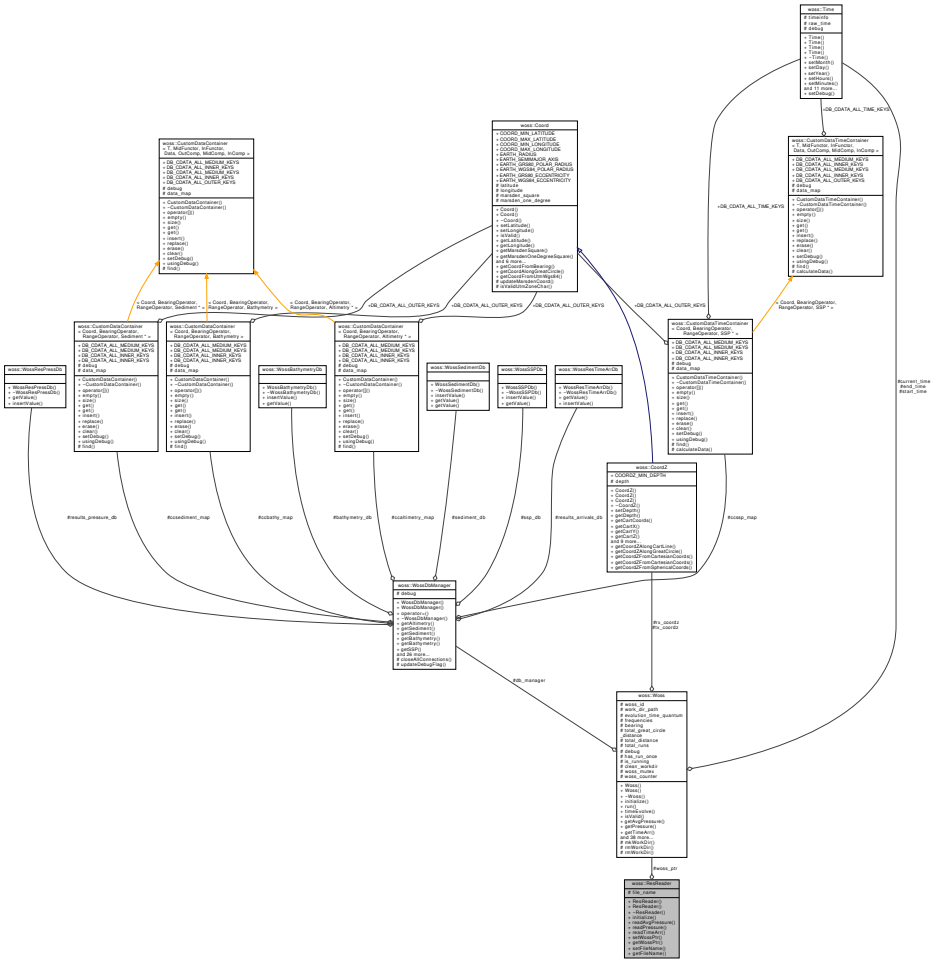
Abstract class for channel simulator result files processing.

```
#include <res-reader.h>
```

Inheritance diagram for woss::ResReader:



Collaboration diagram for woss::ResReader:



Public Member Functions

- ResReader ()
- ResReader (const Woss ∗const woss)
- virtual bool initialize ()=0
- virtual Pressure ∗ readAvgPressure (double tx_depth, double start_rx_depth, double start_rx_range, double end_rx_depth, double end_rx_range)=0
- virtual Pressure ∗ readPressure (double tx_depth, double rx_depth, double rx_range) const =0
- virtual TimeArr ∗ readTimeArr (double tx_depth, double rx_depth, double rx_range) const =0
- ResReader & setWossPtr (const Woss ∗const woss)
- const Woss ∗const getWossPtr ()
- ResReader & setFileName (const ::std::string &name)
- ::std::string getFileName ()

Protected Attributes

- const Woss ∗ woss_ptr
- ::std::string file_name

13.40.1 Detailed Description

Abstract class for channel simulator result files processing.

ResReader class has the task to read and process channel simulator result files

13.40.2 Constructor & Destructor Documentation

13.40.2.1 ResReader() [1/2] ResReader::ResReader ( )

ResReader default constructor

13.40.2.2 ResReader() [2/2] ResReader::ResReader (const Woss *const *woss*)

ResReader constructor

Parameters

woss	const pointer to a const Woss object

13.40.3 Member Function Documentation

13.40.3.1 getFileName() ::std::string woss::ResReader::getFileName ( ) [inline]

Gets the file(s) pathname

Returns

string pathname

References file_name.

13.40.3.2 getWossPtr() const Woss *const woss::ResReader::getWossPtr ( ) [inline]

Gets the Woss pointer

Returns

const pointer to linked Woss object

References woss_ptr.

13.40.3.3 initialize() `virtual bool woss::ResReader::initialize ( ) [pure virtual]`

Initializes the ResReader object

Returns

true if method was successful, *false* otherwise

Implemented in woss::ArrAscResReader, woss::ArrBinResReader, and woss::ShdResReader.

13.40.3.4 readAvgPressure()

```
virtual Pressure * woss::ResReader::readAvgPressure (
        double tx_depth,
        double start_rx_depth,
        double start_rx_range,
        double end_rx_depth,
        double end_rx_range ) [pure virtual]
```

Gets the average Pressure value in given rx range-depth box

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

Returns

a valid Pressure value

Implemented in woss::ArrAscResReader, woss::ArrBinResReader, and woss::ShdResReader.

13.40.3.5 readPressure()

```
virtual Pressure * woss::ResReader::readPressure (
        double tx_depth,
        double rx_depth,
        double rx_range ) const [pure virtual]
```

Gets a Pressure value for given range, depths

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

a valid Pressure value

Implemented in woss::ArrAscResReader, woss::ArrBinResReader, and woss::ShdResReader.

13.40.3.6 readTimeArr()

```
virtual TimeArr * woss::ResReader::readTimeArr (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [pure virtual]
```

Gets a TimeArr value for given range, depths

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

a valid Pressure value

Implemented in woss::ArrAscResReader, woss::ArrBinResReader, and woss::ShdResReader.

13.40.3.7 setFileName()

```
ResReader & woss::ResReader::setFileName (
        const ::std::string & name )  [inline]
```

Sets the file(s) pathname

Parameters

name	const reference to a valid pathname

Returns

reference to ∗**this**

References file_name.

13.40.3.8 setWossPtr()

```
ResReader & woss::ResReader::setWossPtr (
        const Woss *const woss )  [inline]
```

Sets the Woss pointer

Parameters

woss	const pointer to a const Woss object

Returns

reference to ∗**this**

References woss_ptr.

13.40.4 Member Data Documentation

13.40.4.1 file_name `::std::string woss::ResReader::file_name [protected]`

File(s) pathname

Referenced by woss::ArrAscResReader::getArrAscFile(), woss::ArrAscResReader::getArrAscHeader(), woss::ArrBinResReader::getA
woss::ArrBinResReader::getArrBinHeader(), getFileName(), woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeac
woss::ArrAscResReader::initialize(), woss::ArrBinResReader::initialize(), woss::ShdResReader::initialize(), and
setFileName().

13.40.4.2 woss_ptr `const Woss* woss::ResReader::woss_ptr [protected]`

Const pointer to Woss owner

Referenced by woss::ArrAscResReader::getArrAscHeader(), woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getA
woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), getWossPtr(), woss::ArrAscResReader::initialize(),
woss::ArrBinResReader::initialize(), woss::ShdResReader::initialize(), woss::ArrAscResReader::readMapAvgPressure(),
woss::ArrBinResReader::readMapAvgPressure(), woss::ShdResReader::readMapAvgPressure(), and setWossPtr().

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

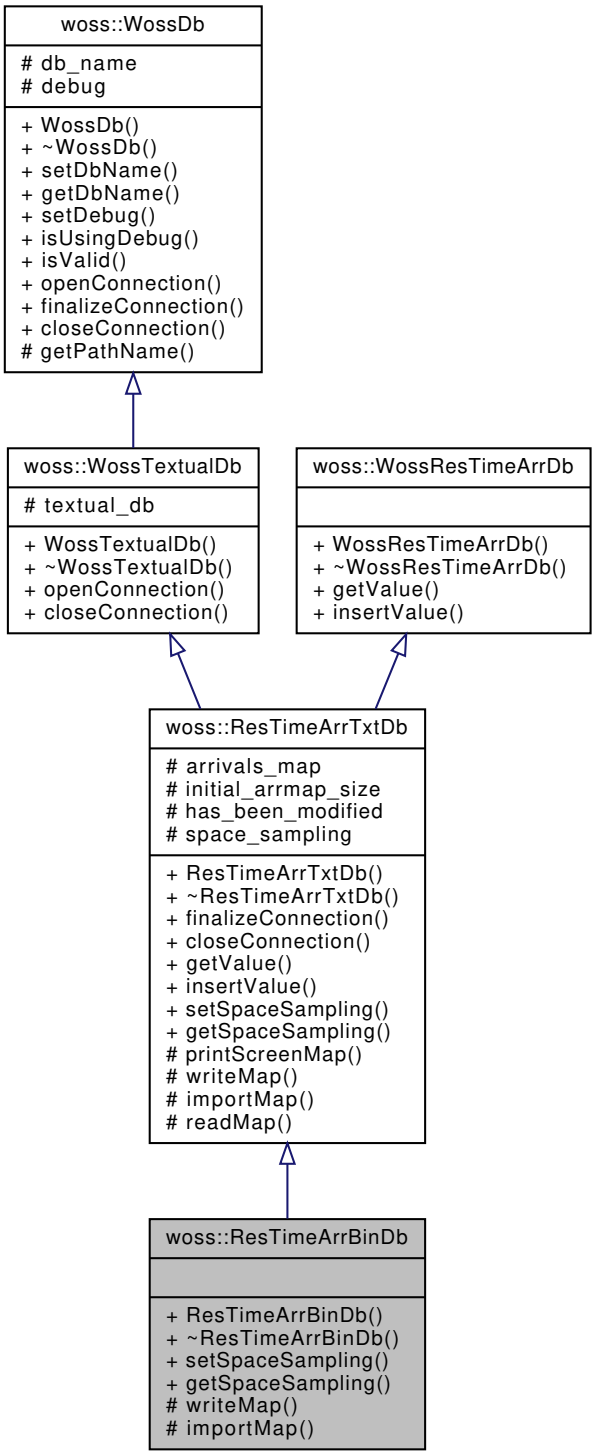
- woss/res-reader.h
- woss/res-reader.cpp

13.41 woss::ResTimeArrBinDb Class Reference

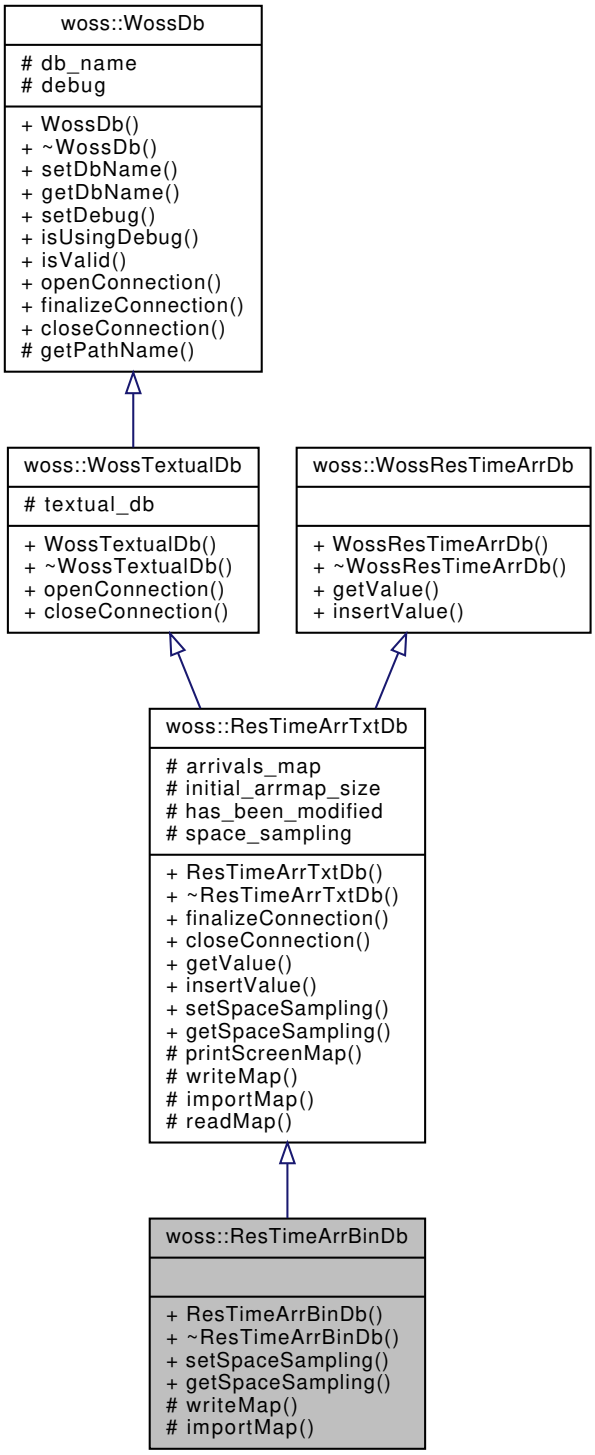
Binary WossDb for TimeArr.

`#include <res-time-arr-bin-db.h>`

Inheritance diagram for woss::ResTimeArrBinDb:



Collaboration diagram for woss::ResTimeArrBinDb:



Public Member Functions

- ResTimeArrBinDb (const ::std::string &name)

Static Public Member Functions

- static void **setSpaceSampling** (double value)
- static double **getSpaceSampling** ()

Protected Member Functions

- virtual bool writeMap ()
- virtual bool importMap ()

Additional Inherited Members

13.41.1 Detailed Description

Binary WossDb for TimeArr.

ResTimeArrBinDb implements WossTextualDb and WossResTimeArrDb for storing calculated TimeArr into a binary file

13.41.2 Constructor & Destructor Documentation

13.41.2.1 ResTimeArrBinDb() `woss::ResTimeArrBinDb::ResTimeArrBinDb ( const ::std::string & name ) [inline]`

ResTimeArrBinDb constructor

Parameters

name	pathname of database

13.41.3 Member Function Documentation

13.41.3.1 importMap() `bool ResTimeArrBinDb::importMap ( ) [protected], [virtual]`

Imports the binary file into arrivals_map. The format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, total channel taps, delay-*i-th* real pressure-*i-th*, imag pressure-*i-th*

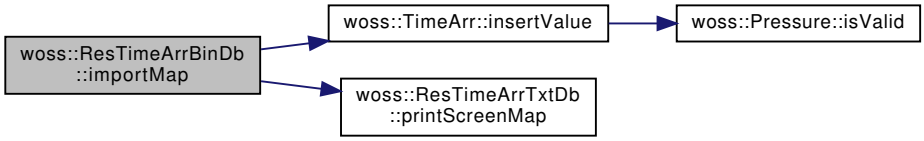
Returns

true if operation succeeds, *false* otherwise

Reimplemented from woss::ResTimeArrTxtDb.

References woss::ResTimeArrTxtDb::arrivals_map, woss::WossDb::db_name, woss::WossDb::debug, woss::ResTimeArrTxtDb::initia
woss::TimeArr::insertValue(), woss::ResTimeArrTxtDb::printScreenMap(), and woss::WossTextualDb::textual_db.

Here is the call graph for this function:



13.41.3.2 writeMap() `bool ResTimeArrBinDb::writeMap ( ) [protected], [virtual]`

Writes arrivals_map into the binary file. The format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, total channel taps, delay-*i-th* real pressure-*i-th*, imag pressure-*i-th*

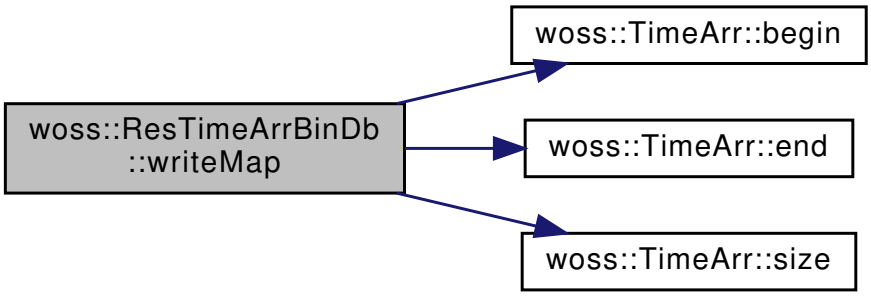
Returns

true if operation succeeds, *false* otherwise

Reimplemented from woss::ResTimeArrTxtDb.

References woss::ResTimeArrTxtDb::arrivals_map, woss::TimeArr::begin(), woss::WossDb::db_name, woss::TimeArr::end(), woss::TimeArr::size(), and woss::WossTextualDb::textual_db.

Here is the call graph for this function:



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

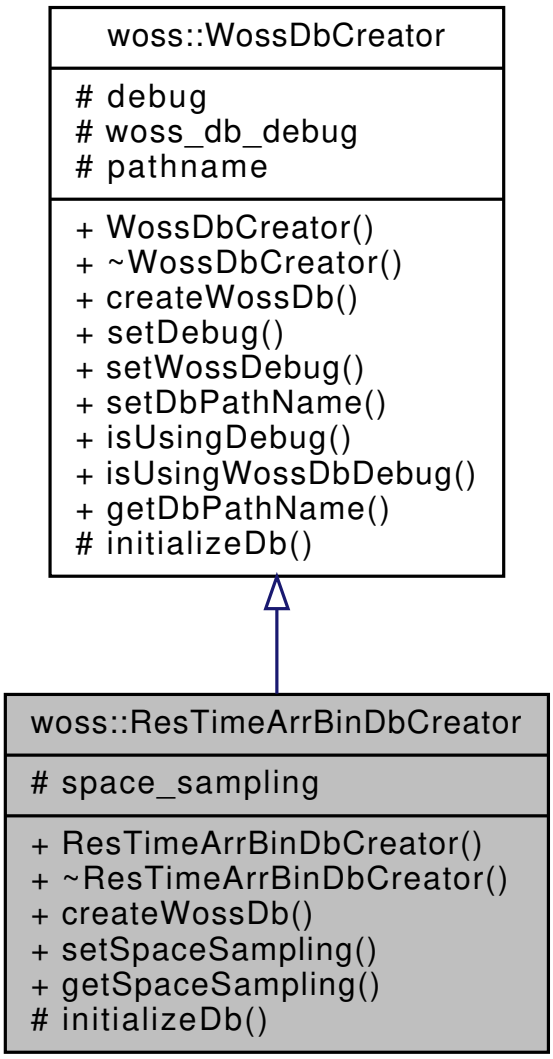
- woss/woss_db/res-time-arr-bin-db.h
- woss/woss_db/res-time-arr-bin-db.cpp

13.42 woss::ResTimeArrBinDbCreator Class Reference

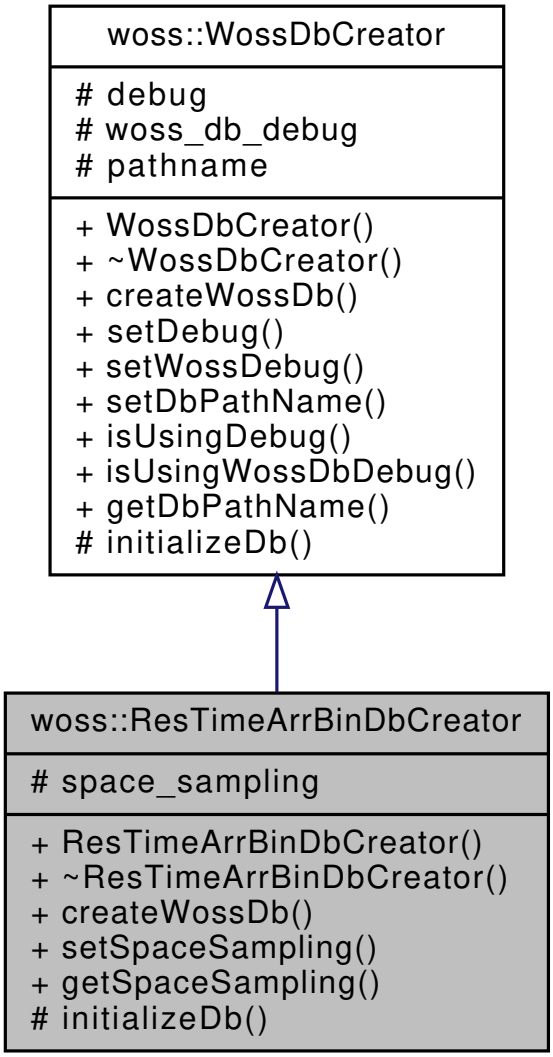
DbCreator for textual TimeArr database.

`#include <res-time-arr-bin-db-creator.h>`

Inheritance diagram for woss::ResTimeArrBinDbCreator:



Collaboration diagram for woss::ResTimeArrBinDbCreator:



Public Member Functions

- ResTimeArrBinDbCreator ()
- virtual WossDb ∗const createWossDb ()
- void **setSpaceSampling** (double value)
- double **getSpaceSampling** ()

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- double **space_sampling**

13.42.1 Detailed Description

DbCreator for textual TimeArr database.

ResTimeArrBinDbCreator implements WossDbCreator for textual TimeArr database

13.42.2 Constructor & Destructor Documentation

13.42.2.1 ResTimeArrBinDbCreator() `ResTimeArrBinDbCreator::ResTimeArrBinDbCreator ( )`

ResTimeArrBinDbCreator default constructor

13.42.3 Member Function Documentation

13.42.3.1 createWossDb() `WossDb *const ResTimeArrBinDbCreator::createWossDb ( ) [virtual]`

This method is called to create and initialize a ResTimeArrTxtDb

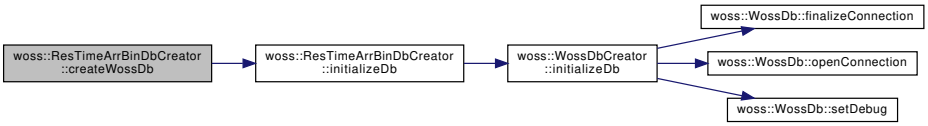
Returns

a pointer to a properly initialized ResTimeArrTxtDb object

Implements woss::WossDbCreator.

References woss::WossDbCreator::debug, initializeDb(), and woss::WossDbCreator::pathname.

Here is the call graph for this function:



13.42.3.2 initializeDb() `bool ResTimeArrBinDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created ResTimeArrTxtDb

Returns

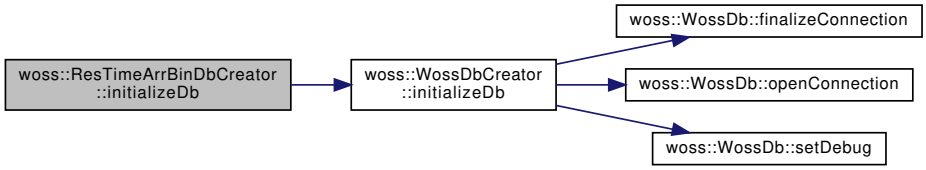
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



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

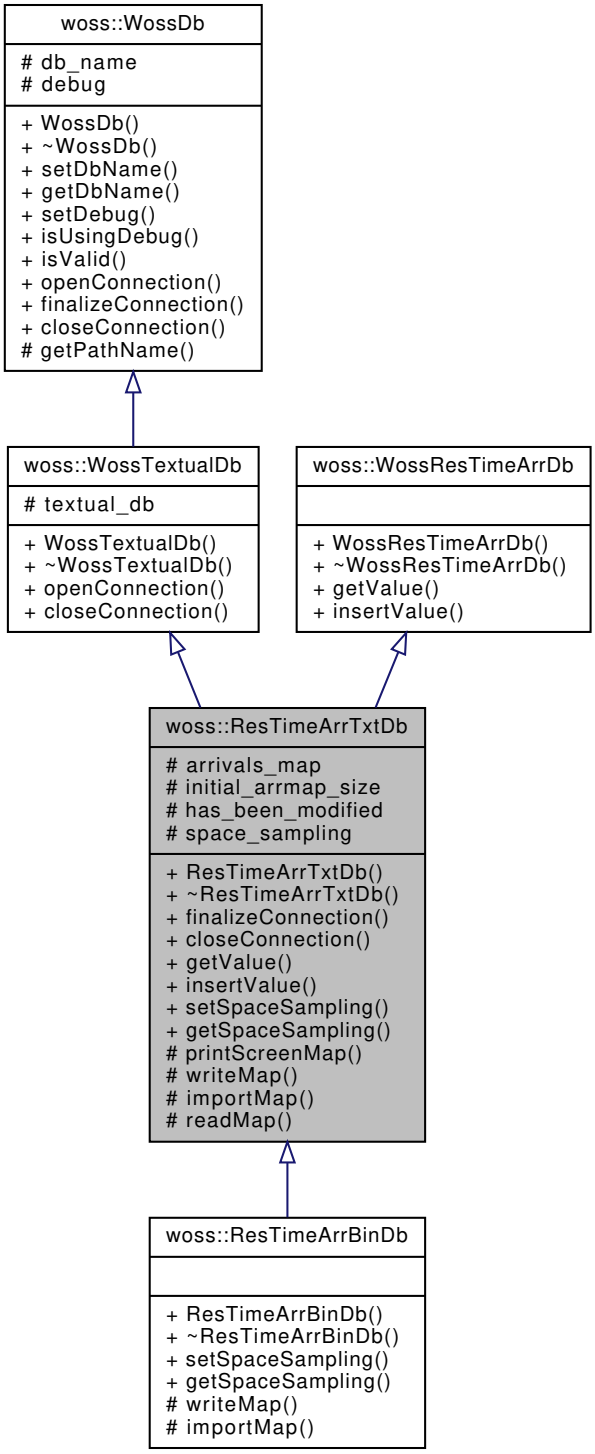
- woss/woss_db/res-time-arr-bin-db-creator.h
- woss/woss_db/res-time-arr-bin-db-creator.cpp

13.43 woss::ResTimeArrTxtDb Class Reference

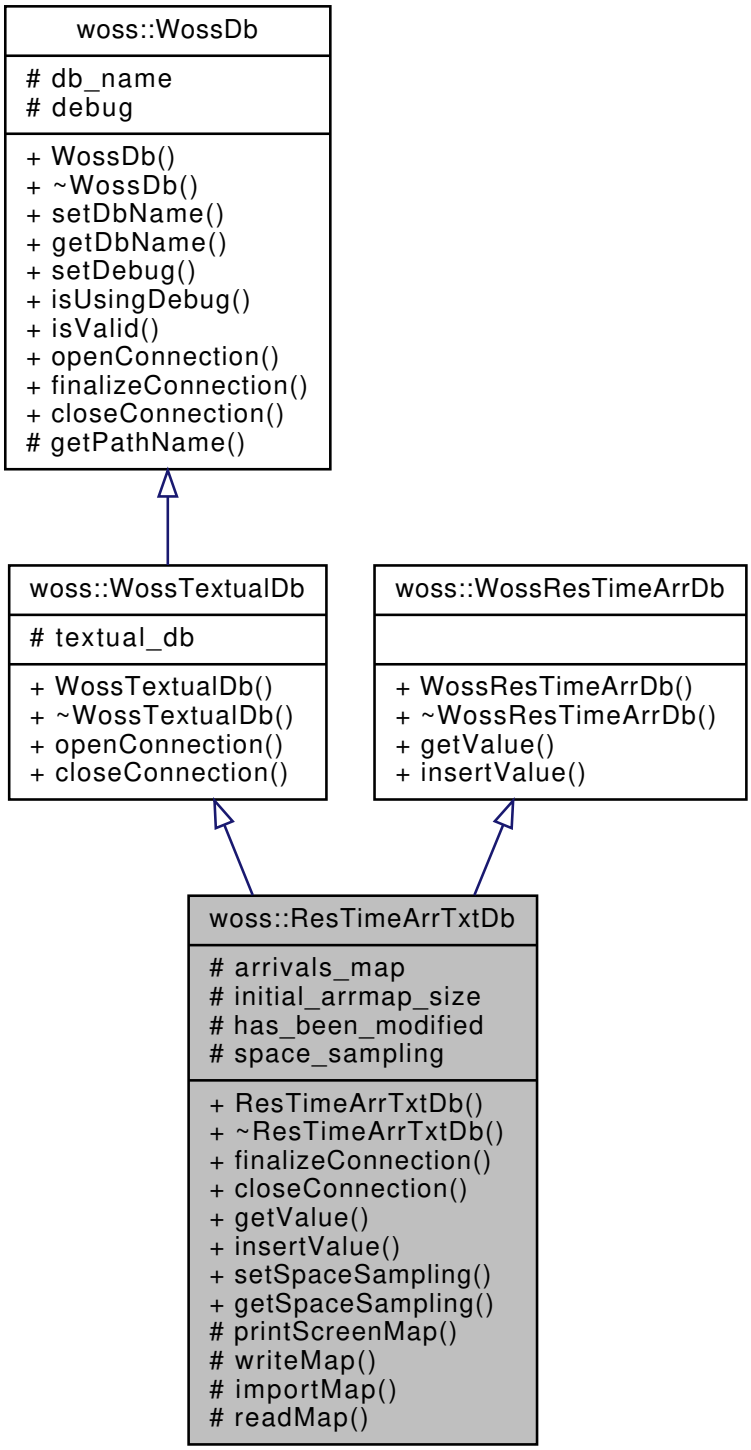
Textual WossDb for TimeArr.

```
#include <res-time-arr-txt-db.h>
```

Inheritance diagram for woss::ResTimeArrTxtDb:



Collaboration diagram for woss::ResTimeArrTxtDb:



Public Member Functions

- ResTimeArrTxtDb (const ::std::string &name)
- virtual bool finalizeConnection ()
- virtual bool closeConnection ()
- virtual TimeArr * getValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const
- virtual bool insertValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const TimeArr &channel)

Static Public Member Functions

- static void **setSpaceSampling** (double value)
- static double **getSpaceSampling** ()

Protected Types

- typedef ::std::map< time_t, TimeArr > **TimeMap**
- typedef TimeMap::iterator **TMIter**
- typedef TimeMap::const_iterator **TMCIter**
- typedef TimeMap::reverse_iterator **TMRIter**
- typedef ::std::map< PDouble, TimeMap > **FreqMap**
- typedef FreqMap::iterator **FMIter**
- typedef FreqMap::const_iterator **FMCIter**
- typedef FreqMap::reverse_iterator **FMRIter**
- typedef ::std::map< CoordZ, FreqMap, CoordComparator< ResTimeArrTxtDb, CoordZ > > **RxMap**
- typedef RxMap::iterator **RxMIter**
- typedef RxMap::const_iterator **RxMCIter**
- typedef RxMap::reverse_iterator **RxMRIter**
- typedef ::std::map< CoordZ, RxMap, CoordComparator< ResTimeArrTxtDb, CoordZ > > ArrMatrix
- typedef ArrMatrix::iterator **AMXIter**
- typedef ArrMatrix::const_iterator **AMXCIter**
- typedef ArrMatrix::reverse_iterator **AMXRIter**
- typedef ArrMatrix::const_reverse_iterator **AMXCRIter**

Protected Member Functions

- void printScreenMap ()
- virtual bool writeMap ()
- virtual bool importMap ()
- const TimeArr * readMap (const CoordZ &tx, const CoordZ &rx, const double frequency, const Time &time↩ _value) const

Protected Attributes

- ArrMatrix arrivals_map
- int initial_arrmap_size
- bool **has_been_modified**

Static Protected Attributes

- static double **space_sampling** = 0.0

13.43.1 Detailed Description

Textual WossDb for TimeArr.

ResTimeArrTxtDb implements WossTextualDb and WossResTimeArrDb for storing calculated TimeArr into a text file

13.43.2 Member Typedef Documentation

13.43.2.1 ArrMatrix `typedef ::std::map< CoordZ, RxMap, CoordComparator< ResTimeArrTxtDb, CoordZ > > woss::ResTimeArrTxtDb::ArrMatrix [protected]`

Multidimensional map that links a transmitter CoordZ to a receiver CoordZ to a frequency PDouble value and finally to a TimeArr value

13.43.3 Constructor & Destructor Documentation

13.43.3.1 ResTimeArrTxtDb() `ResTimeArrTxtDb::ResTimeArrTxtDb ( const ::std::string & name )`

ResTimeArrTxtDb constructor

Parameters

name	pathname of database

13.43.4 Member Function Documentation

13.43.4.1 closeConnection() `bool ResTimeArrTxtDb::closeConnection ( ) [virtual]`

Closes the connection to the text file provided

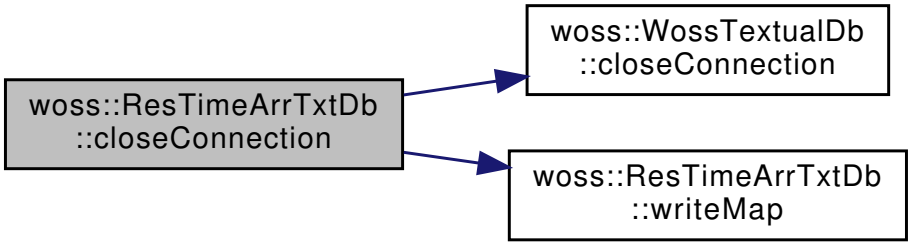
Returns

true if method was successful, *false* otherwise

Reimplemented from woss::WossTextualDb.

References woss::WossTextualDb::closeConnection(), and writeMap().

Here is the call graph for this function:



13.43.4.2 finalizeConnection() `bool ResTimeArrTxtDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions

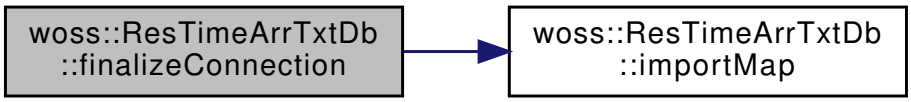
Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References importMap().

Here is the call graph for this function:



13.43.4.3 getValue()

```
TimeArr * ResTimeArrTxtDb::getValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [virtual]
```

Returns a pointer to a heap-created TimeArr value for given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object

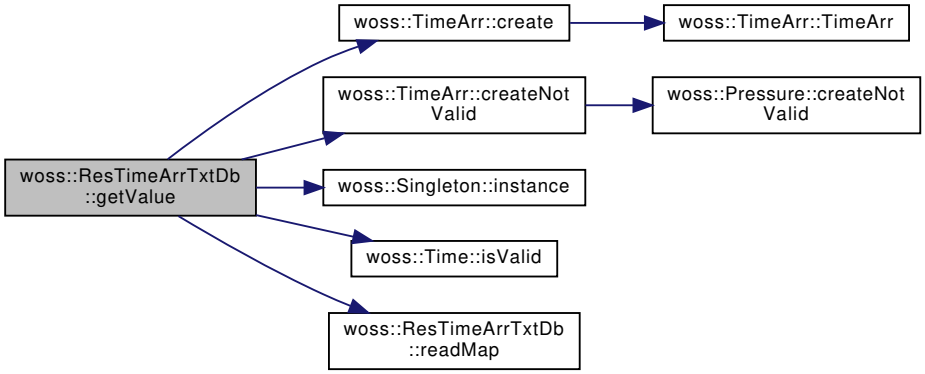
Returns

valid TimeArr if parameters are found, *not valid* otherwise

Implements woss::WossResTimeArrDb.

References arrivals_map, woss::TimeArr::create(), woss::TimeArr::createNotValid(), woss::Singleton< T >::instance(), woss::Time::isValid(), and readMap().

Here is the call graph for this function:



13.43.4.4 importMap() `bool ResTimeArrTxtDb::importMap ( ) [protected], [virtual]`

Imports the formatted textual files into arrivals_map. The column format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, total channel taps, delay-*i-th* real pressure-*i-th*, imag pressure-*i-th*

Returns

true if operation succeeds, *false* otherwise

Reimplemented in woss::ResTimeArrBinDb.

References woss::WossDb::db_name, and woss::WossTextualDb::textual_db.

Referenced by finalizeConnection().

13.43.4.5 insertValue() `bool ResTimeArrTxtDb::insertValue (`

```
const CoordZ & coord_tx,
const CoordZ & coord_rx,
const double frequency,
const Time & time_value,
const TimeArr & channel ) [virtual]
```

Inserts the given TimeArr value in the arrivals_map at given frequency, transmitter and receiver coordinates

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time& object
channel	computed TimeArr

Implements woss::WossResTimeArrDb.

References arrivals_map, and woss::WossDb::debug.

13.43.4.6 printScreenMap() `void ResTimeArrTxtDb::printScreenMap ( ) [protected]`

Prints arrivals_map to screen. The columns format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, total channel taps, delay-*i-th* real pressure-*i-th*, imag pressure-*i-th*

Returns

true if operation succeeds, *false* otherwise

Referenced by woss::ResTimeArrBinDb::importMap().

13.43.4.7 readMap()

```
const TimeArr * ResTimeArrTxtDb::readMap (
    const CoordZ & tx,
    const CoordZ & rx,
    const double frequency,
    const Time & time_value ) const  [protected]
```

Reads given values from arrivals_map

Parameters

tx	valid transmitter coordinates
rx	valid receiver coordinates
frequency	frequency [hz]
time_value	const reference to a valid Time object

Returns

valid TimeArr if parameters are found, not valid otherwise

References arrivals_map, and woss::WossDb::debug.

Referenced by getValue().

13.43.4.8 writeMap() `bool ResTimeArrTxtDb::writeMap ( ) [protected], [virtual]`

Writes arrivals_map into the textual file. The columns format is the following:
tx latitude, tx longitude, tx depth, rx latitude, rx longitude, rx depth, frequency, total channel taps, delay-*i-th* real pressure-*i-th*, imag pressure-*i-th*

Returns

true if operation succeeds, *false* otherwise

Reimplemented in woss::ResTimeArrBinDb.

References woss::WossDb::db_name, and woss::WossTextualDb::textual_db.

Referenced by closeConnection().

13.43.5 Member Data Documentation

13.43.5.1 arrivals_map `ArrMatrix woss::ResTimeArrTxtDb::arrivals_map [protected]`

ArrMatrix map for storing imported and user inserted TimeArr values

Referenced by getValue(), woss::ResTimeArrBinDb::importMap(), insertValue(), readMap(), and woss::ResTimeArrBinDb::writeMap().

13.43.5.2 initial_arrmap_size `int woss::ResTimeArrTxtDb::initial_arrmap_size [protected]`

arrivals_map's initial size. If pressure_map's size is greater on closeConnection() the map will be written to disk

Referenced by woss::ResTimeArrBinDb::importMap().

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

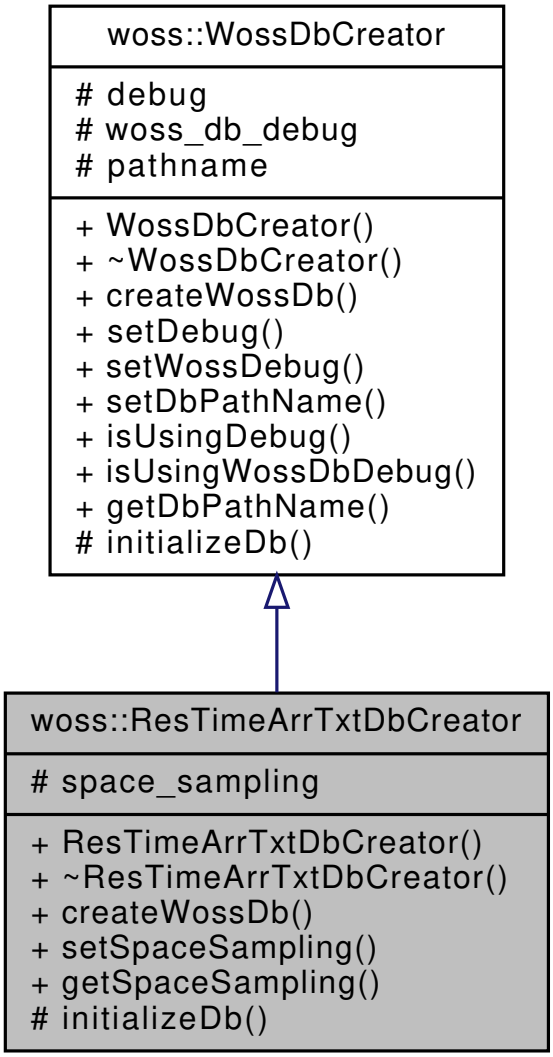
- woss/woss_db/res-time-arr-txt-db.h
- woss/woss_db/res-time-arr-txt-db.cpp

13.44 woss::ResTimeArrTxtDbCreator Class Reference

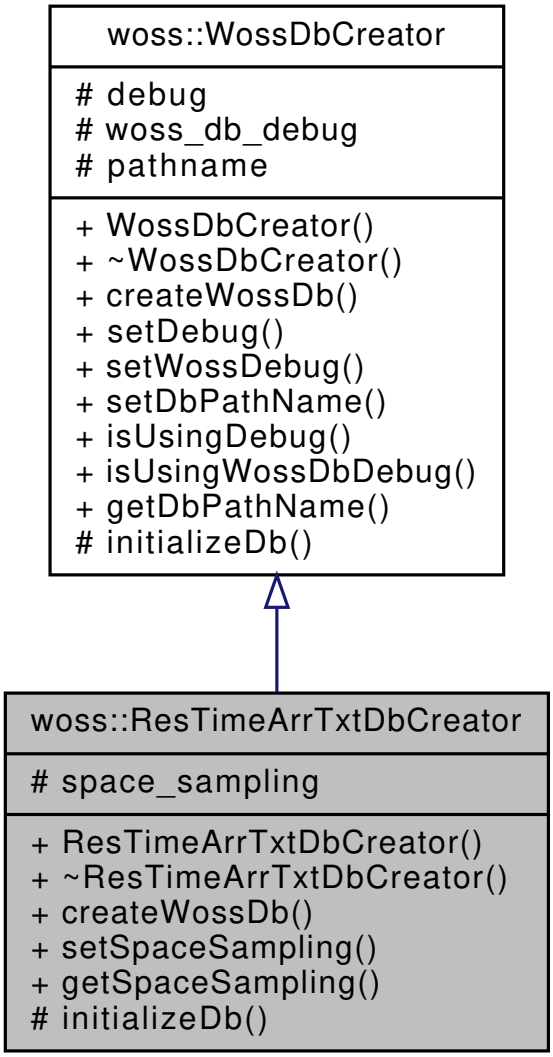
DbCreator for textual TimeArr database.

`#include <res-time-arr-txt-db-creator.h>`

Inheritance diagram for woss::ResTimeArrTxtDbCreator:



Collaboration diagram for woss::ResTimeArrTxtDbCreator:



Public Member Functions

- ResTimeArrTxtDbCreator ()
- virtual WossDb ∗const createWossDb ()
- void **setSpaceSampling** (double value)
- double **getSpaceSampling** ()

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- double **space_sampling**

13.44.1 Detailed Description

DbCreator for textual TimeArr database.

ResTimeArrTxtDbCreator implements WossDbCreator for textual TimeArr database

13.44.2 Constructor & Destructor Documentation

13.44.2.1 ResTimeArrTxtDbCreator() `ResTimeArrTxtDbCreator::ResTimeArrTxtDbCreator ( )`

ResTimeArrTxtDbCreator default constructor

13.44.3 Member Function Documentation

13.44.3.1 createWossDb() `WossDb *const ResTimeArrTxtDbCreator::createWossDb ( ) [virtual]`

This method is called to create and initialize a ResTimeArrTxtDb

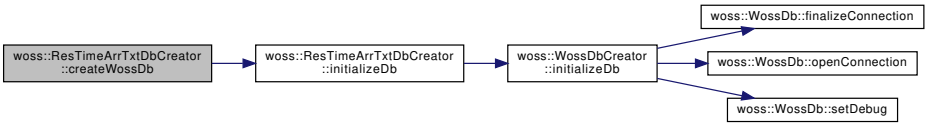
Returns

a pointer to a properly initialized ResTimeArrTxtDb object

Implements woss::WossDbCreator.

References woss::WossDbCreator::debug, initializeDb(), and woss::WossDbCreator::pathname.

Here is the call graph for this function:



13.44.3.2 initializeDb() `bool ResTimeArrTxtDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created ResTimeArrTxtDb

Returns

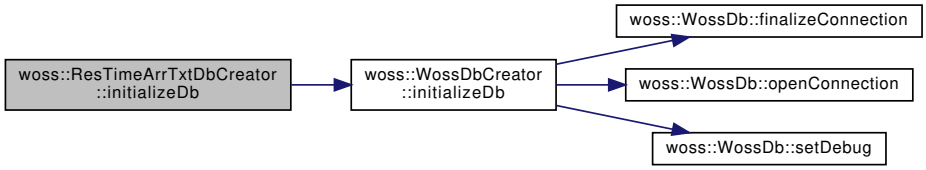
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



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

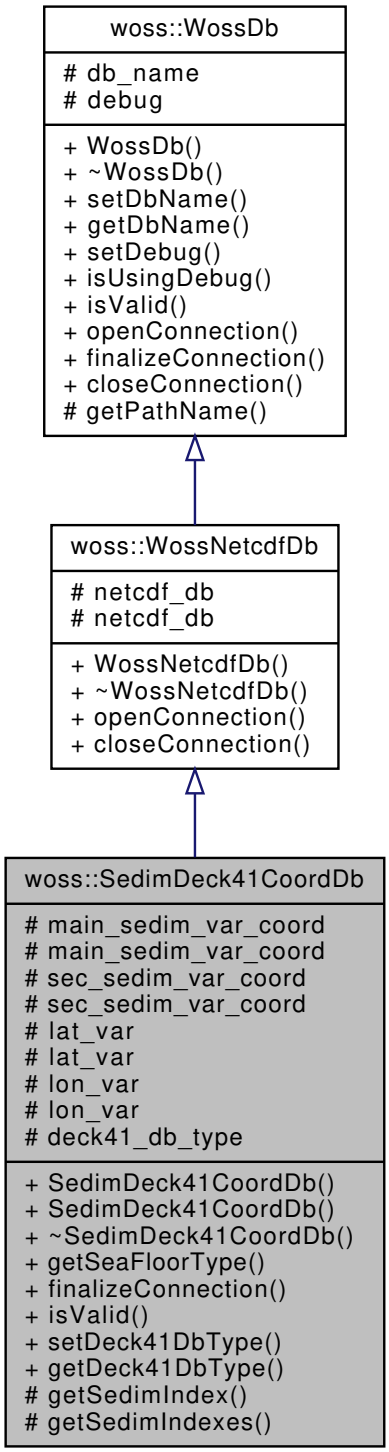
- woss/woss_db/res-time-arr-txt-db-creator.h
- woss/woss_db/res-time-arr-txt-db-creator.cpp

13.45 woss::SedimDeck41CoordDb Class Reference

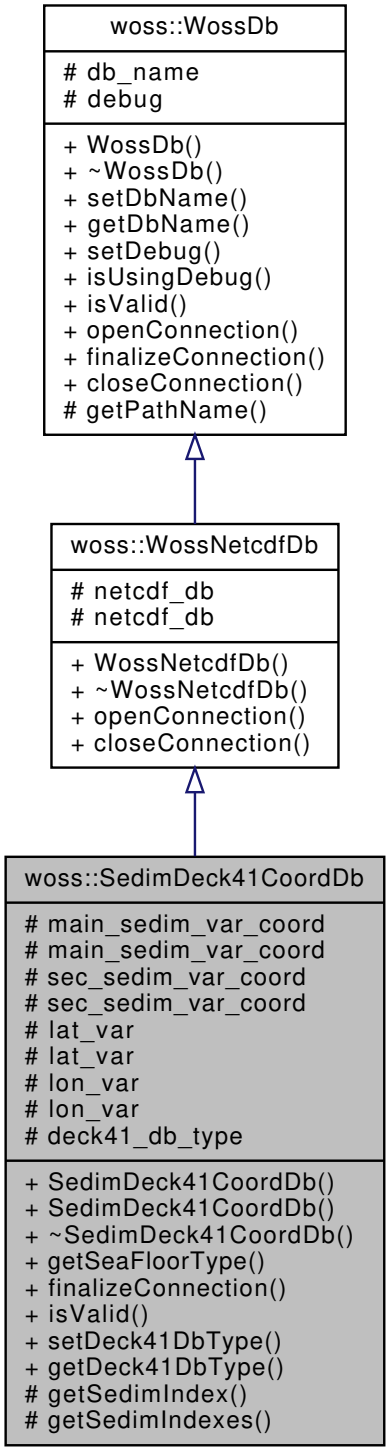
WossDb for custom made NetCDF DECK41 Sediment database.

```
#include <sediment-deck41-coord-db.h>
```

Inheritance diagram for woss::SedimDeck41CoordDb:



Collaboration diagram for woss::SedimDeck41CoordDb:



Public Member Functions

- SedimDeck41CoordDb (const ::std::string &name=DB_NAME_NOT_SET)
- SedimDeck41CoordDb (const ::std::string &name, DECK41DbType db_type)
- Deck41Types getSeaFloorType (const Coord &coordinates) const
- virtual bool finalizeConnection ()
- virtual bool isValid ()
- void setDeck41DbType (DECK41DbType db_type)
- DECK41DbType getDeck41DbType () const

Protected Member Functions

- int getSedimIndex (const Coord &coords) const
- ::std::pair< int, int > getSedimIndexes (const Coord &coordinates) const

Protected Attributes

- netCDF::NcVar main_sedim_var_coord
- NcVar * **main_sedim_var_coord**
- netCDF::NcVar sec_sedim_var_coord
- NcVar * **sec_sedim_var_coord**
- netCDF::NcVar lat_var
- NcVar * **lat_var**
- netCDF::NcVar lon_var
- NcVar * **lon_var**
- DECK41DbType deck41_db_type

13.45.1 Detailed Description

WossDb for custom made NetCDF DECK41 Sediment database.

SedimDeck41Db implements WossNetcdfDb for the custom made NetCDF DECK41 Sediment database

13.45.2 Constructor & Destructor Documentation

13.45.2.1 SedimDeck41CoordDb() [1/2]

```
SedimDeck41CoordDb::SedimDeck41CoordDb (
        const ::std::string & name = DB_NAME_NOT_SET )
```

SedimDeck41CoordDb default constructor. Default constructed object are not valid

Parameters

name	pathname of database

References woss::DECK41_DB_V1_TYPE.

13.45.2.2 SedimDeck41CoordDb() [2/2]

```
SedimDeck41CoordDb::SedimDeck41CoordDb (
        const ::std::string & name,
        DECK41DbType db_type )
```

SedimDeck41CoordDb default constructor. Default constructed object are not valid

Parameters

name	pathname of database
db_type	DECK41 db type

13.45.3 Member Function Documentation

13.45.3.1 finalizeConnection() `bool SedimDeck41CoordDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions, used to create and initialize NetCDF variables

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References woss::DECK41_DB_INVALID_TYPE, deck41_db_type, woss::DECK41_DB_V2_TYPE, lat_var, lon_var, main_sedim_var_coord, woss::WossNetcdfDb::netcdf_db, and sec_sedim_var_coord.

13.45.3.2 getDeck41DbType() `DECK41DbType woss::SedimDeck41CoordDb::getDeck41DbType ( ) const [inline]`

Returns the current DECK41 DB type

Returns

current deck41 db type

References deck41_db_type.

13.45.3.3 getSeaFloorType() `Deck41Types SedimDeck41CoordDb::getSeaFloorType ( const Coord & coordinates ) const`

Returns a Deck41Types for the given coordinates

Parameters

coordinates	const reference to a valid Coord object

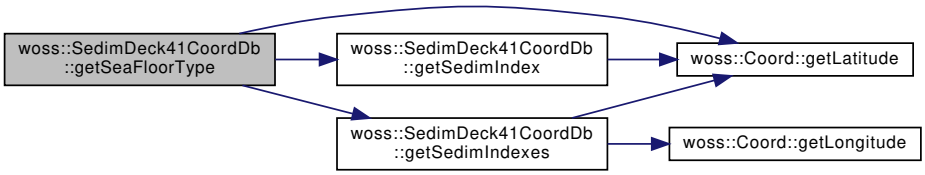
Returns

a Deck41Types value (main type value, secondary type value)

References woss::WossDb::debug, deck41_db_type, woss::DECK41_DB_V1_TYPE, woss::DECK41_DB_V2_TYPE, woss::Coord::getLatitude(), getSedimIndex(), getSedimIndexes(), lat_var, lon_var, main_sedim_var_coord, and sec_sedim_var_coord.

Referenced by woss::SedimDeck41Db::getDeck41TypesFromCoords().

Here is the call graph for this function:



13.45.3.4 getSedimIndex()

```
int SedimDeck41CoordDb::getSedimIndex (
    const Coord & coords ) const  [protected]
```

Returns the index used by a NetCDF variable to get the DECK41 floortype integer value

Parameters

coordinates	const reference to a valid Coord object

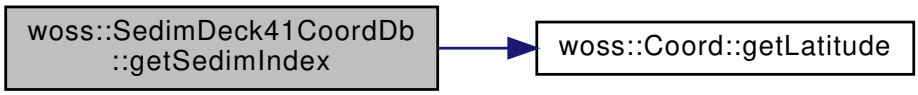
Returns

index value

References woss::Coord::getLatitude().

Referenced by getSeaFloorType().

Here is the call graph for this function:



13.45.3.5 getSedimIndexes()

```
std::pair< int, int > SedimDeck41CoordDb::getSedimIndexes (
    const Coord & coordinates ) const  [protected]
```

Returns the index pair used by a NetCDF variable to get the DECK41 floortype integer value

Parameters

coordinates	const reference to a valid Coord object

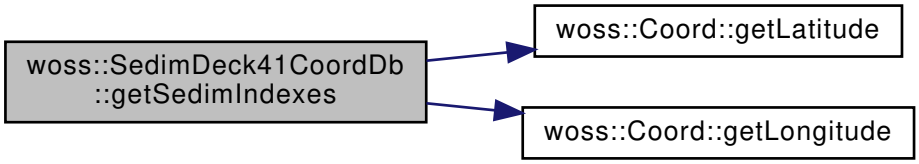
Returns

index pair

References woss::Coord::COORD_MIN_LATITUDE, woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getSeaFloorType().

Here is the call graph for this function:



13.45.3.6 isValid() `virtual bool woss::SedimDeck41CoordDb::isValid ( ) [inline], [virtual]`

Checks the validity of SedimDeck41CoordDb

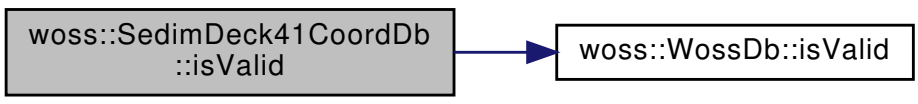
Returns

true if pathname is valid, *false* otherwise

Reimplemented from woss::WossDb.

References woss::WossDb::db_name, woss::DECK41_DB_INVALID_TYPE, deck41_db_type, and woss::WossDb::isValid().

Here is the call graph for this function:



13.45.3.7 setDeck41DbType() `void woss::SedimDeck41CoordDb::setDeck41DbType ( DECK41DbType db_type ) [inline]`

Sets the current DECK41 Db type

Parameters

db_type	DECK41DbType

References deck41_db_type.

Referenced by woss::SedimDeck41DbCreator::initializeSedimDb().

13.45.4 Member Data Documentation

13.45.4.1 deck41_db_type `DECK41DbType woss::SedimDeck41CoordDb::deck41_db_type [protected]`

DECK41 database type

Referenced by finalizeConnection(), getDeck41DbType(), getSeaFloorType(), isValid(), and setDeck41DbType().

13.45.4.2 lat_var `netCDF::NcVar woss::SedimDeck41CoordDb::lat_var [protected]`

NetCDF latitude variable

Referenced by finalizeConnection(), and getSeaFloorType().

13.45.4.3 lon_var `netCDF::NcVar woss::SedimDeck41CoordDb::lon_var [protected]`

NetCDF longitude variable

Referenced by finalizeConnection(), and getSeaFloorType().

13.45.4.4 main_sedim_var_coord `netCDF::NcVar woss::SedimDeck41CoordDb::main_sedim_var_coord [protected]`

NetCDF variable representing main DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

13.45.4.5 sec_sedim_var_coord `netCDF::NcVar woss::SedimDeck41CoordDb::sec_sedim_var_coord [protected]`

NetCDF variable representing secondary DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

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

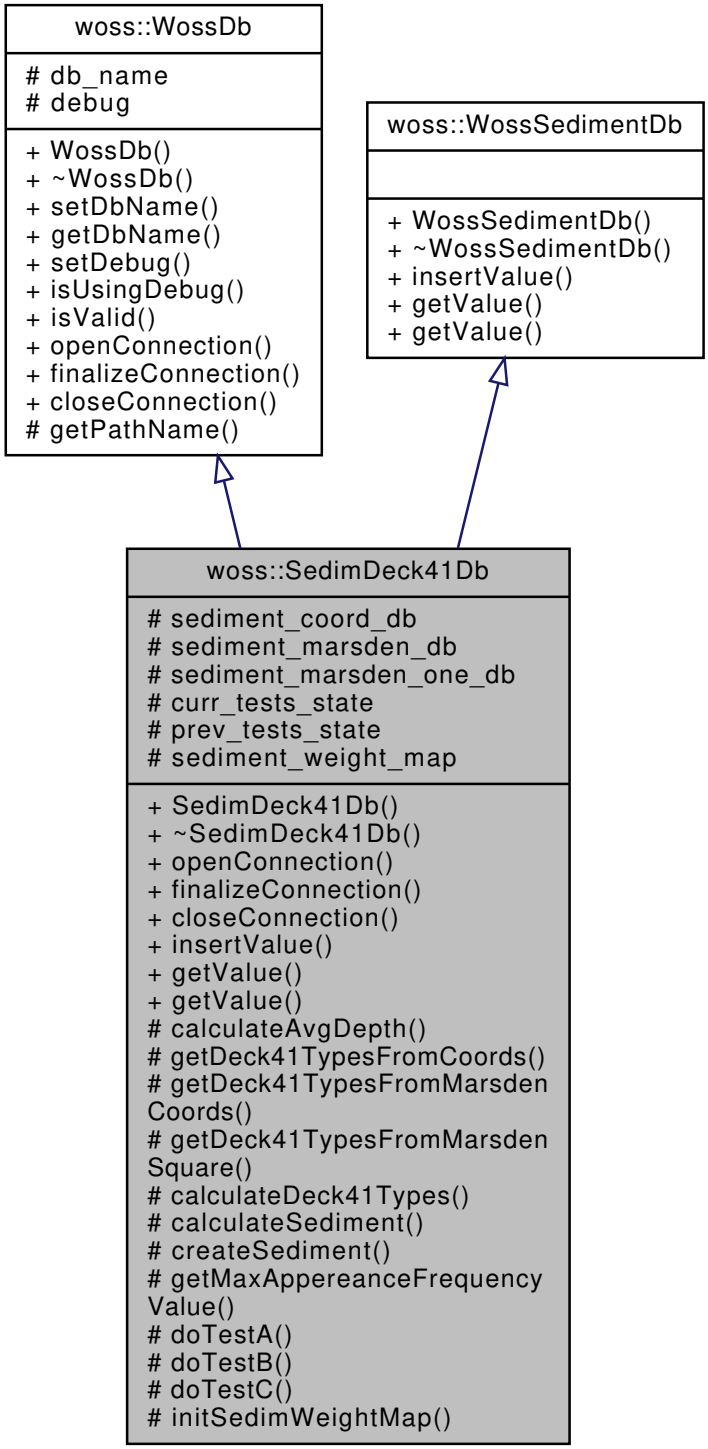
- woss/woss_db/sediment-deck41-coord-db.h
- woss/woss_db/sediment-deck41-coord-db.cpp

13.46 woss::SedimDeck41Db Class Reference

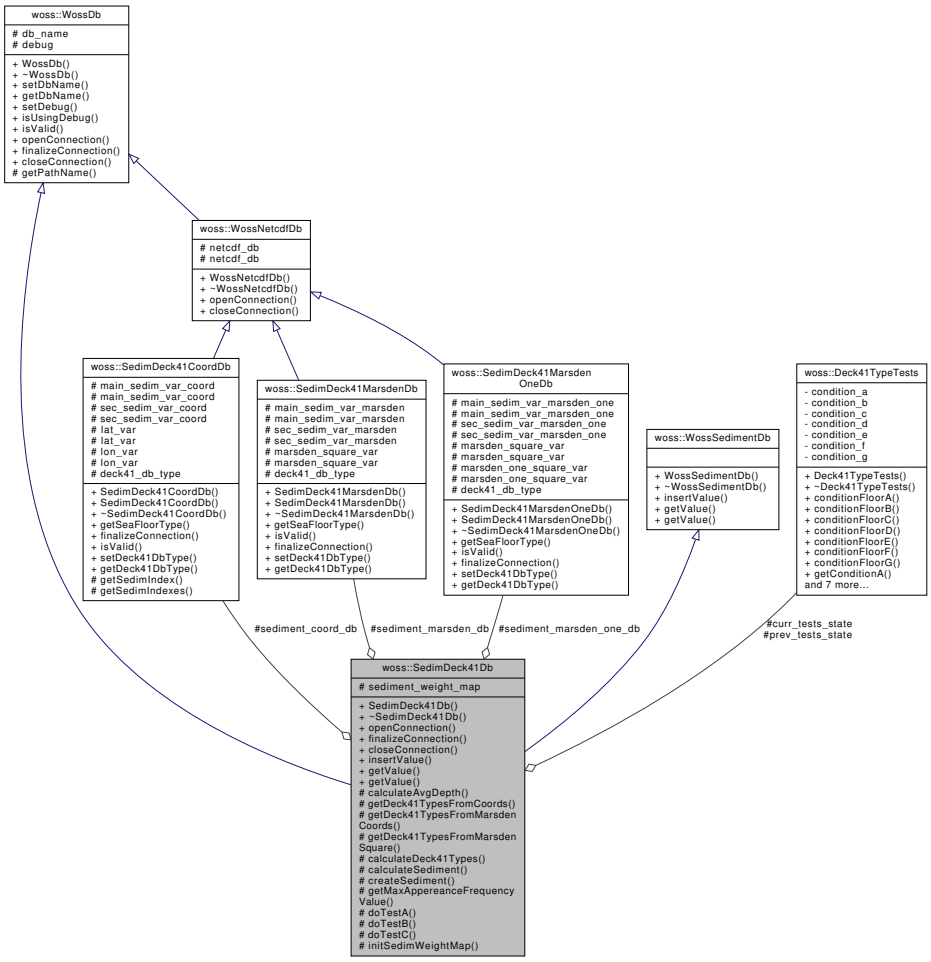
WossDb for NetCDF DECK41 Sediment database.

`#include <sediment-deck41-db.h>`

Inheritance diagram for woss::SedimDeck41Db:



Collaboration diagram for woss::SedimDeck41Db:



Public Member Functions

- SedimDeck41Db (const ::std::string &name)
- virtual bool openConnection ()
- virtual bool finalizeConnection ()
- virtual bool closeConnection ()
- virtual bool insertValue (const Coord &coordinates, const Sediment &sediment_value)
- virtual Sediment * getValue (const CoordZ &coordz) const
- virtual Sediment * getValue (const CoordZVector &coordz_vector) const

Protected Member Functions

- double calculateAvgDepth (const CoordZVector &coordz_vector) const
- Deck41Types getDeck41TypesFromCoords (const CoordZVector &coordz_vector) const
- Deck41Types getDeck41TypesFromMarsdenCoords (const CoordZVector &coordz_vector) const
- Deck41Types getDeck41TypesFromMarsdenSquare (const CoordZVector &coordz_vector) const
- Deck41Types calculateDeck41Types (const CoordZVector &coordz_vector) const
- Sediment * calculateSediment (const Deck41Types &floor_types, double avg_depth) const
- Sediment * createSediment (int deck41_type, double depth) const
- int getMaxAppereanceFrequencyValue (const FrequencyMap &frequency_map) const
- bool doTestA (const Deck41TypeTests &test) const
- bool doTestB (const Deck41TypeTests &test) const
- bool doTestC (const Deck41TypeTests &test) const

Static Protected Member Functions

- static SedimWeightMap initSedimWeightMap ()

Protected Attributes

- SedimDeck41CoordDb sediment_coord_db
- SedimDeck41MarsdenDb sediment_marsden_db
- SedimDeck41MarsdenOneDb sediment_marsden_one_db
- Deck41TypeTests curr_tests_state
- Deck41TypeTests prev_tests_state

Static Protected Attributes

- static SedimWeightMap sediment_weight_map = SedimDeck41Db::initSedimWeightMap()

Friends

- class **SedimDeck41DbCreator**

13.46.1 Detailed Description

WossDb for NetCDF DECK41 Sediment database.

SedimDeck41Db implements WossDb and WossSedimentDb for NetCDF DECK41 Sediment database. It provides logic to handle the three custom made databases: coordinates, marsden square and marsden coordinates.

13.46.2 Constructor & Destructor Documentation

13.46.2.1 SedimDeck41Db() `SedimDeck41Db::SedimDeck41Db ( const ::std::string & name )`

SedimDeck41Db constructor

Parameters

name	pathname of database

13.46.3 Member Function Documentation

13.46.3.1 calculateAvgDepth() double SedimDeck41Db::calculateAvgDepth (const CoordZVector & coordz_vector) const [protected]

Computes the average depth of a valid CoordZVector for Sediment creation

Parameters

coordz_vector	a vector of *valid* CoordZ

Returns

the average depth of the vector [m]

Referenced by getValue().

13.46.3.2 calculateDeck41Types() Deck41Types SedimDeck41Db::calculateDeck41Types (const CoordZVector & coordz_vector) const [protected]

Gets the Deck41Types of a valid CoordZVector searching in order in: coord db, marsden db, marsden one db. On each search Deck41TypeTests curr_tests_state and prev_tests_state are updated, and based on result conditions, the process stops and returns or goes on to the next database.

Parameters

coordz_vector	a vector of *valid* CoordZ

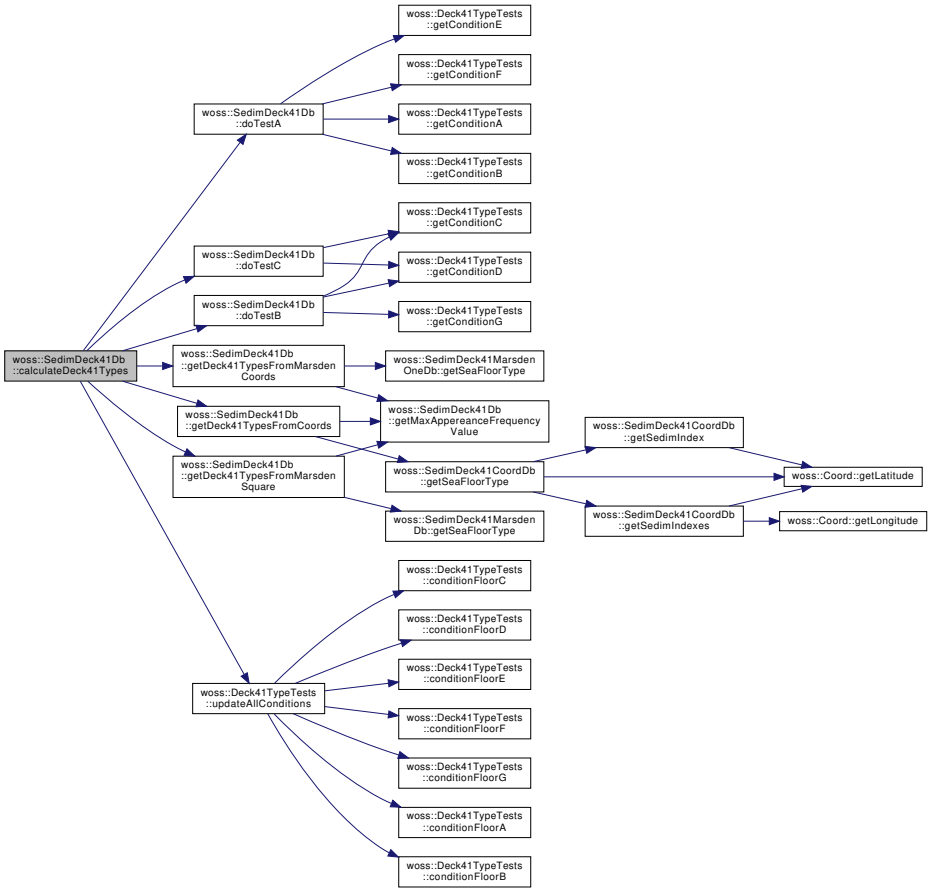
Returns

the corresponding Deck41Types (main sediment type, second sediment type) of the vector

References curr_tests_state, woss::WossDb::debug, doTestA(), doTestB(), doTestC(), getDeck41TypesFromCoords(), getDeck41TypesFromMarsdenCoords(), getDeck41TypesFromMarsdenSquare(), prev_tests_state, and woss::Deck41TypeTests::upd

Referenced by getValue().

Here is the call graph for this function:



13.46.3.3 calculateSediment()

```
Sediment * SedimDeck41Db::calculateSediment (
        const Deck41Types & floor_types,
        double avg_depth ) const  [protected]
```

Creates the corresponding Sediment from the searched Deck41Types returned by calculateDeck41Types

Parameters

floor_types	the Deck41Types pair resulted from the search process
avg_depth	depth value [m] for Sediment calculations

Returns

Sediment value

Referenced by getValue().

13.46.3.4 closeConnection() `bool SedimDeck41Db::closeConnection ( ) [virtual]`

Closes the connection to the three databases provided

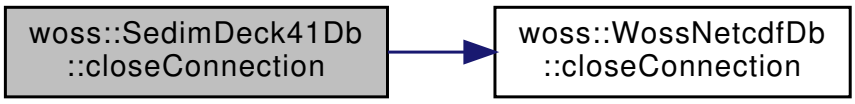
Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References woss::WossNetcdfDb::closeConnection(), sediment_coord_db, sediment_marsden_db, and sediment_marsden_one_db.

Here is the call graph for this function:



13.46.3.5 createSediment() `Sediment * SedimDeck41Db::createSediment ( int deck41_type, double depth ) const [protected]`

Creates a Sediment from the DECK41 integer type number

Parameters

deck41_type	DECK41 integer floor type number
depth	depth value [m] for Sediment calculations

Returns

Sediment value

See also

Deck41TypeTests

13.46.3.6 doTestA() `bool woss::SedimDeck41Db::doTestA ( const Deck41TypeTests & test ) const [inline], [protected]`

Does Test A on given Deck41TypeTests reference. A positive result means that the search in the databases for a valid Deck41Types is over

Parameters

test	const Deck41TypeTests reference

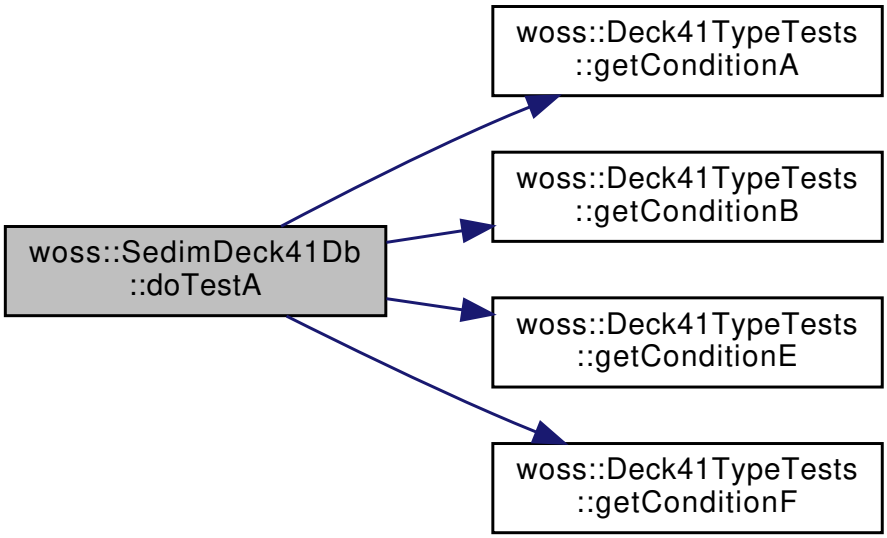
Returns

true if test succeed, *false* otherwise

References woss::Deck41TypeTests::getConditionA(), woss::Deck41TypeTests::getConditionB(), woss::Deck41TypeTests::getConditi and woss::Deck41TypeTests::getConditionF().

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.7 doTestB()

```
bool woss::SedimDeck41Db::doTestB (
        const Deck41TypeTests & test ) const  [inline], [protected]
```

Does Test B on given Deck41TypeTests reference. A positive result means that the search in the databases for a valid Deck41Types should continue. If next search step fails a Sediment not valid will be returned from the search process

Parameters

test	const Deck41TypeTests reference

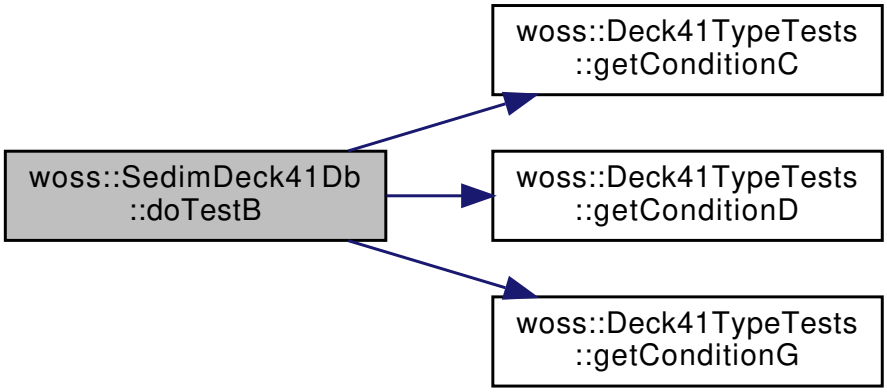
Returns

true if test succeed, *false* otherwise

References woss::Deck41TypeTests::getConditionC(), woss::Deck41TypeTests::getConditionD(), and woss::Deck41TypeTests::getCo

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.8 doTestC() `bool woss::SedimDeck41Db::doTestC ( const Deck41TypeTests & test ) const [inline], [protected]`

Does Test C on given Deck41TypeTests reference. A positive result means that the search in the databases for a valid Deck41Types is over if the next search step Test A fails

Parameters

test	const Deck41TypeTests reference

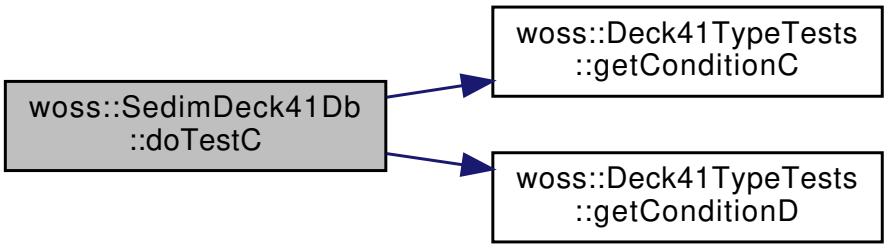
Returns

true if test succeed, *false* otherwise

References woss::Deck41TypeTests::getConditionC(), and woss::Deck41TypeTests::getConditionD().

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.9 finalizeConnection() `virtual bool woss::SedimDeck41Db::finalizeConnection ( ) [inline], [virtual]`

Not allowed

Implements woss::WossDb.

13.46.3.10 getDeck41TypesFromCoords() `Deck41Types SedimDeck41Db::getDeck41TypesFromCoords ( const CoordZVector & coordz_vector ) const [protected]`

Gets the Deck41Types of a valid CoordZVector from the SedimDeck41CoordDb database

Parameters

coordz_vector	a vector of *valid* CoordZ

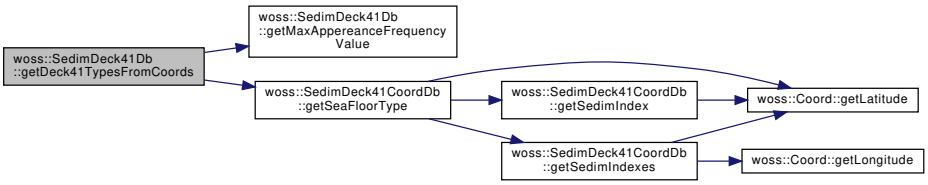
Returns

the corresponding Deck41Types (main sediment type, second sediment type) of the vector

References woss::WossDb::debug, getMaxAppereanceFrequencyValue(), woss::SedimDeck41CoordDb::getSeaFloorType(), and sediment_coord_db.

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.11 getDeck41TypesFromMarsdenCoords() `Deck41Types SedimDeck41Db::getDeck41TypesFromMarsdenCoords ( const CoordZVector & coordz_vector ) const [protected]`

Gets the Deck41Types of a valid CoordZVector from the SedimDeck41MarsdenDb database

Parameters

coordz_vector	a vector of *valid* CoordZ

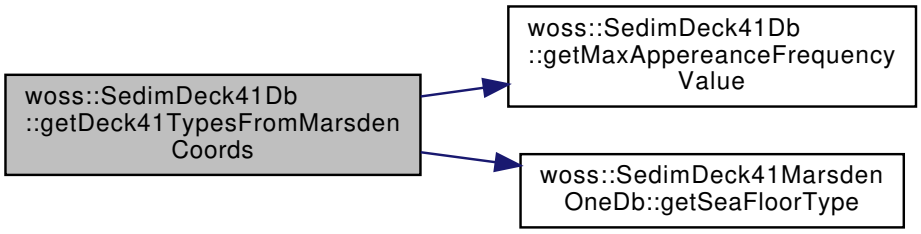
Returns

the corresponding Deck41Types (main sediment type, second sediment type) of the vector

References woss::WossDb::debug, getMaxAppereanceFrequencyValue(), woss::SedimDeck41MarsdenOneDb::getSeaFloorType(), and sediment_marsden_one_db.

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.12 getDeck41TypesFromMarsdenSquare()

```
Deck41Types SedimDeck41Db::getDeck41TypesFromMarsdenSquare (
            const CoordZVector & coordz_vector ) const  [protected]
```

Gets the Deck41Types of a valid CoordZVector from the SedimDeck41MarsdenOneDb database

Parameters

coordz_vector	a vector of *valid* CoordZ

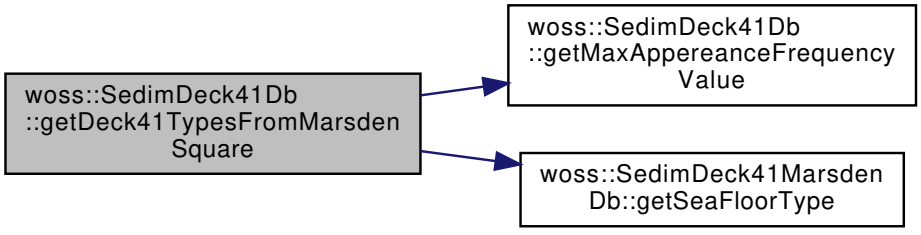
Returns

the corresponding Deck41Types (main sediment type, second sediment type) of the vector

References woss::WossDb::debug, getMaxAppereanceFrequencyValue(), woss::SedimDeck41MarsdenDb::getSeaFloorType(), and sediment_marsden_db.

Referenced by calculateDeck41Types().

Here is the call graph for this function:



13.46.3.13 getMaxAppereanceFrequencyValue() int SedimDeck41Db::getMaxAppereanceFrequencyValue (const FrequencyMap & *frequency_map*) const [protected]

Returns the type of the FrequencyMap that has the max number of times of appeareance in a database query result

Parameters

frequency_map	const FrequencyMap reference

Returns

DECK41 integer type

Referenced by getDeck41TypesFromCoords(), getDeck41TypesFromMarsdenCoords(), and getDeck41TypesFromMarsdenSquare().

13.46.3.14 getValue() [1/2] Sediment * SedimDeck41Db::getValue (const CoordZ & *coordz*) const [virtual]

Returns a pointer to a heap-based Sediment for given coordinates and depth, if present in any of the three databases. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid CoordZ object

Returns

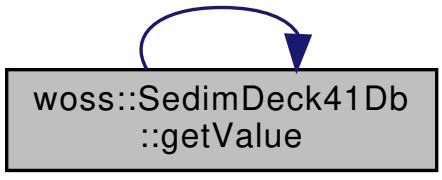
valid Sediment if coordinates are found, *not valid* otherwise

Implements woss::WossSedimentDb.

References getValue().

Referenced by getValue().

Here is the call graph for this function:



13.46.3.15 getValue() [2/2] Sediment * SedimDeck41Db::getValue (const CoordZVector & *coordz_vector*) const [virtual]

Returns a pointer to a heap-based Sediment for given coordinates and depth vector, if at least one set of coordinates is present in any of the three databases. **User is responsible of pointer's ownership**

Parameters

coordz_vector	const reference to a valid CoordZ vector

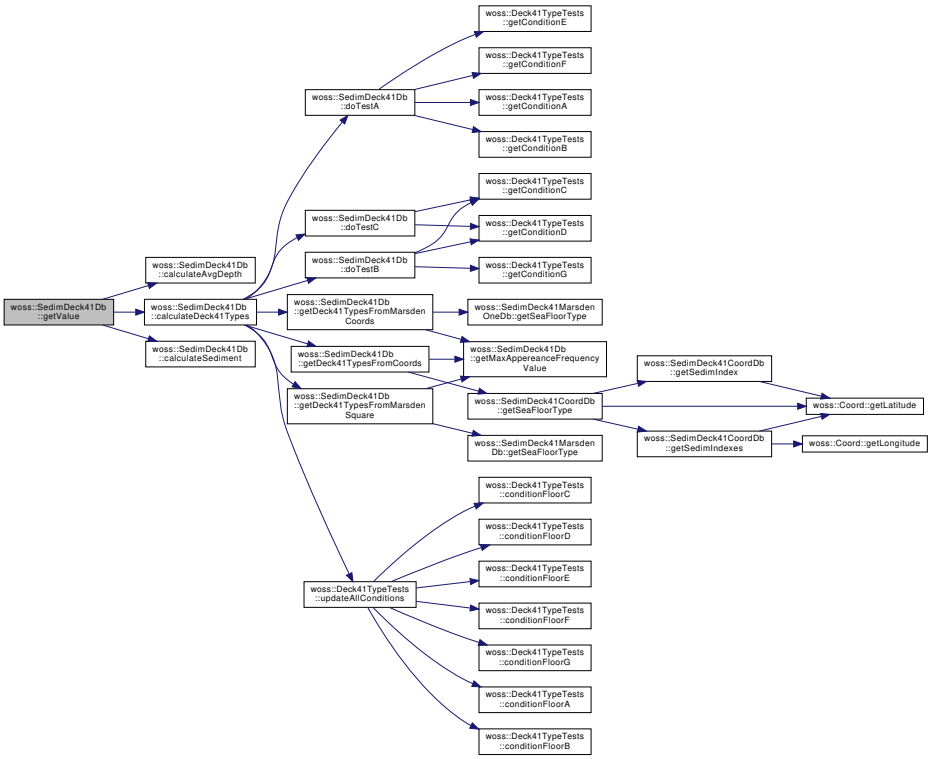
Returns

valid Sediment if at least one set of coordinates is found, *not valid* otherwise

Implements woss::WossSedimentDb.

References calculateAvgDepth(), calculateDeck41Types(), and calculateSediment().

Here is the call graph for this function:



13.46.3.16 initSedimWeightMap() `SedimWeightMap SedimDeck41Db::initSedimWeightMap ( ) [static], [protected]`

Initializes the static sediment_weight_map

Returns

a SedimWeightMap

13.46.3.17 insertValue() `bool SedimDeck41Db::insertValue ( const Coord & coordinates, const Sediment & sediment_value ) [virtual]`

Inserts the given woss::Sediment value in the database for given coordinates

Parameters

coordinates	const reference to a valid Coord object
bathymetry_value	const Reference to Sediment value to be inserted

Returns

true if method was successful, *false* otherwise

Implements woss::WossSedimentDb.

13.46.3.18 openConnection() `virtual bool woss::SedimDeck41Db::openConnection ( ) [inline], [virtual]`

Not allowed

Implements woss::WossDb.

13.46.4 Member Data Documentation

13.46.4.1 curr_tests_state `Deck41TypeTests woss::SedimDeck41Db::curr_tests_state [mutable], [protected]`

Current iteration Deck41TypeTests

Referenced by calculateDeck41Types().

13.46.4.2 prev_tests_state `Deck41TypeTests woss::SedimDeck41Db::prev_tests_state [mutable], [protected]`

Previous iteration Deck41TypeTests

Referenced by calculateDeck41Types().

13.46.4.3 sediment_coord_db `SedimDeck41CoordDb woss::SedimDeck41Db::sediment_coord_db [protected]`

WossDb handling custom made NetCDF DECK41 database

Referenced by closeConnection(), getDeck41TypesFromCoords(), and woss::SedimDeck41DbCreator::initializeSedimDb().

13.46.4.4 sediment_marsden_db

`SedimDeck41MarsdenDb woss::SedimDeck41Db::sediment_marsden_db [protected]`

WossDb handling custom made NetCDF DECK41 marsden square database

Referenced by closeConnection(), getDeck41TypesFromMarsdenSquare(), and woss::SedimDeck41DbCreator::initializeSedimDb().

13.46.4.5 sediment_marsden_one_db

`SedimDeck41MarsdenOneDb woss::SedimDeck41Db::sediment_marsden_one_db [protected]`

WossDb handling custom made NetCDF DECK41 marsden coordinates database

Referenced by closeConnection(), getDeck41TypesFromMarsdenCoords(), and woss::SedimDeck41DbCreator::initializeSedimDb().

13.46.4.6 sediment_weight_map

`SedimWeightMap SedimDeck41Db::sediment_weight_map = SedimDeck41Db::initSedimWe [static], [protected]`

Weight map for DECK41 floor type to Sediment conversion

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

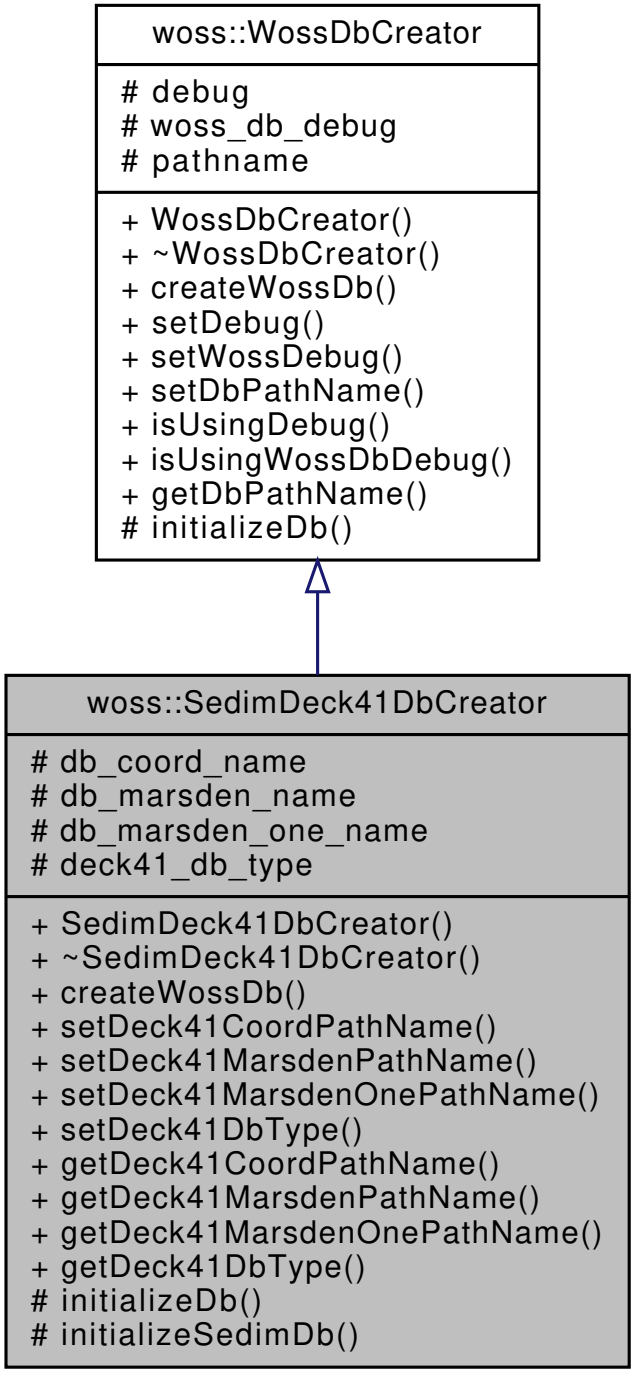
- woss/woss_db/sediment-deck41-db.h
- woss/woss_db/sediment-deck41-db.cpp

13.47 woss::SedimDeck41DbCreator Class Reference

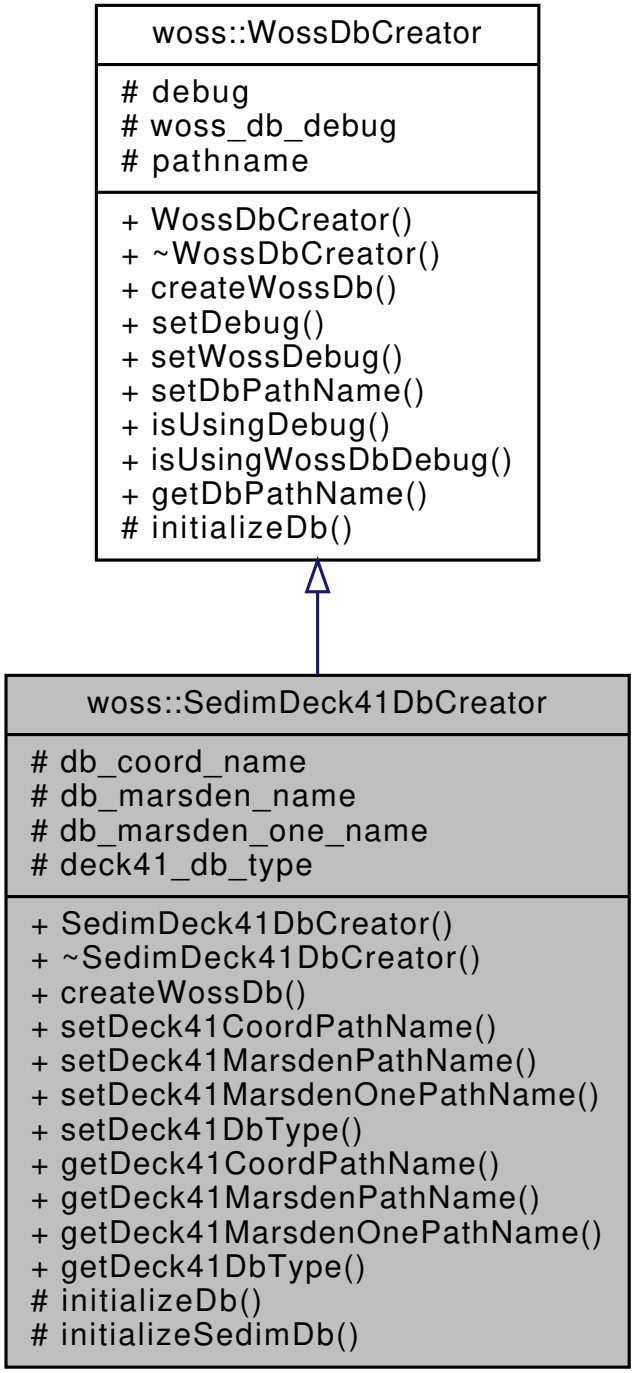
DbCreator for NetCDF Deck41 Sediment database.

`#include <sediment-deck41-db-creator.h>`

Inheritance diagram for woss::SedimDeck41DbCreator:



Collaboration diagram for woss::SedimDeck41DbCreator:



Public Member Functions

- SedimDeck41DbCreator ()
- virtual WossDb ∗const createWossDb ()
- void **setDeck41CoordPathName** (const ::std::string &name)
- void **setDeck41MarsdenPathName** (const ::std::string &name)
- void **setDeck41MarsdenOnePathName** (const ::std::string &name)
- void **setDeck41DbType** (DECK41DbType db_type)
- ::std::string **getDeck41CoordPathName** () const

- ::std::string **getDeck41MarsdenPathName** () const
- ::std::string **getDeck41MarsdenOnePathName** () const
- DECK41DbType **getDeck41DbType** () const

Protected Member Functions

- virtual bool initializeDb (WossDb ∗woss_db)
- bool initializeSedimDb (SedimDeck41Db ∗const woss_db)

Protected Attributes

- ::std::string **db_coord_name**
- ::std::string **db_marsden_name**
- ::std::string **db_marsden_one_name**
- DECK41DbType deck41_db_type

13.47.1 Detailed Description

DbCreator for NetCDF Deck41 Sediment database.

SedimDeck41DbCreator implements WossDbCreator for NetCDF Deck41 Sediment database

13.47.2 Constructor & Destructor Documentation

13.47.2.1 SedimDeck41DbCreator() `SedimDeck41DbCreator::SedimDeck41DbCreator ( )`

SedimDeck41DbCreator default constructor

References woss::DECK41_DB_V1_TYPE.

13.47.3 Member Function Documentation

13.47.3.1 createWossDb() `WossDb *const SedimDeck41DbCreator::createWossDb ( ) [virtual]`

This method is called to create and initialize a SedimDeck41Db

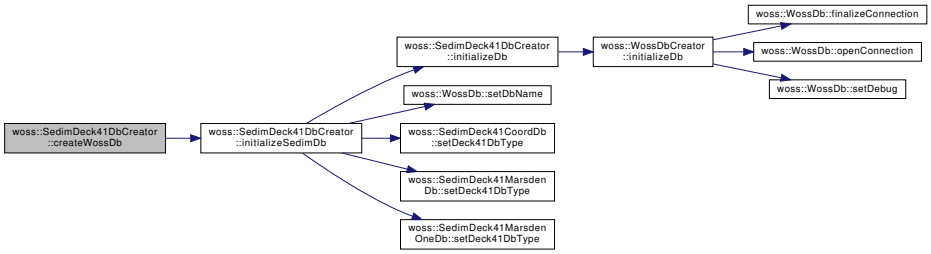
Returns

a pointer to a properly initialized SedimDeck41Db object

Implements woss::WossDbCreator.

References initializeSedimDb().

Here is the call graph for this function:



13.47.3.2 initializeDb() `bool SedimDeck41DbCreator::initializeDb ( WossDb * woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created SedimDeck41Db

Returns

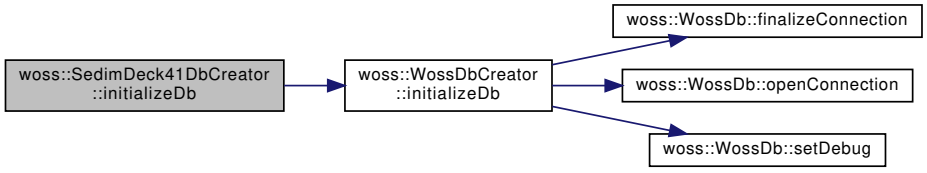
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by initializeSedimDb().

Here is the call graph for this function:



13.47.3.3 initializeSedimDb() `bool SedimDeck41DbCreator::initializeSedimDb ( SedimDeck41Db *const woss_db ) [protected]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created SedimDeck41Db

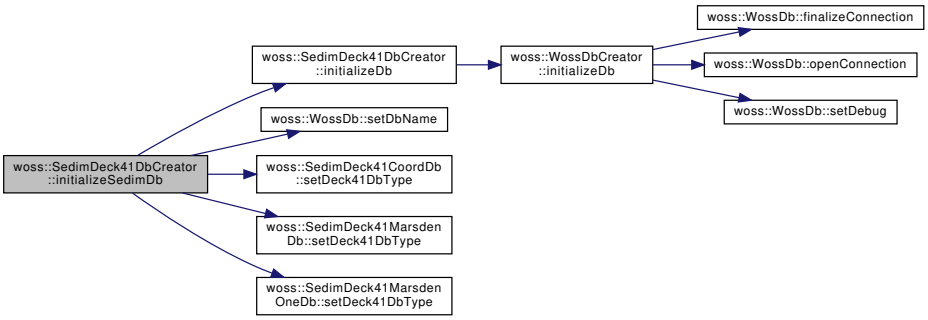
Returns

true if the method succeed, *false* otherwise

References deck41_db_type, initializeDb(), woss::SedimDeck41Db::sediment_coord_db, woss::SedimDeck41Db::sediment_marsden
woss::SedimDeck41Db::sediment_marsden_one_db, woss::WossDb::setDbName(), woss::SedimDeck41CoordDb::setDeck41DbTyp
woss::SedimDeck41MarsdenDb::setDeck41DbType(), and woss::SedimDeck41MarsdenOneDb::setDeck41DbType().

Referenced by createWossDb().

Here is the call graph for this function:



13.47.4 Member Data Documentation

13.47.4.1 deck41_db_type `DECK41DbType woss::SedimDeck41DbCreator::deck41_db_type [protected]`

DECK41 database type

Referenced by initializeSedimDb().

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

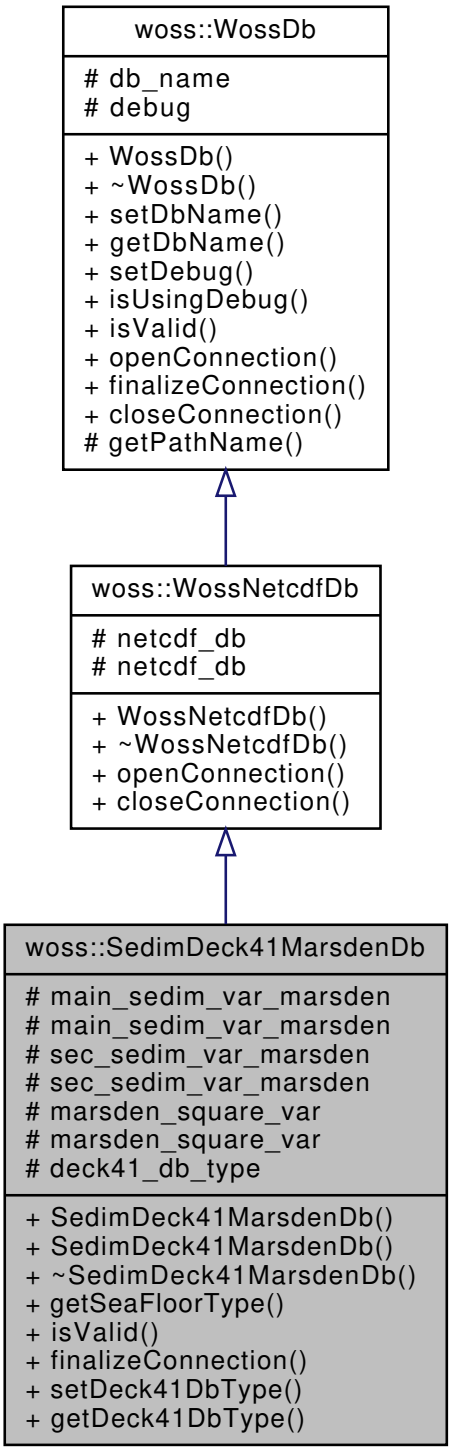
- woss/woss_db/sediment-deck41-db-creator.h
- woss/woss_db/sediment-deck41-db-creator.cpp

13.48 woss::SedimDeck41MarsdenDb Class Reference

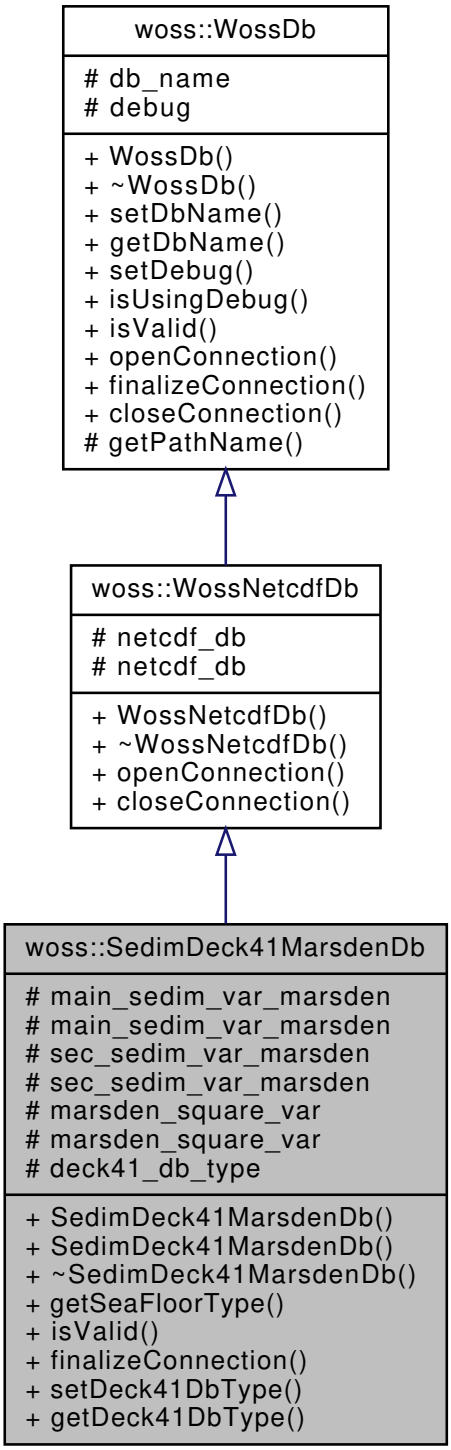
WossDb for custom made NetCDF marsden square DECK41 Sediment database.

`#include <sediment-deck41-marsden-db.h>`

Inheritance diagram for woss::SedimDeck41MarsdenDb:



Collaboration diagram for woss::SedimDeck41MarsdenDb:



Public Member Functions

- SedimDeck41MarsdenDb (const ::std::string &name=DB_NAME_NOT_SET)
- SedimDeck41MarsdenDb (const ::std::string &name, DECK41DbType db_type)
- Deck41Types getSeaFloorType (const Marsden &marsden_square) const
- virtual bool isValid ()
- virtual bool finalizeConnection ()
- void setDeck41DbType (DECK41DbType db_type)
- DECK41DbType getDeck41DbType () const

Protected Attributes

- netCDF::NcVar main_sedim_var_marsden
- NcVar * **main_sedim_var_marsden**
- netCDF::NcVar sec_sedim_var_marsden
- NcVar * **sec_sedim_var_marsden**
- netCDF::NcVar marsden_square_var
- NcVar * **marsden_square_var**
- DECK41DbType deck41_db_type

Additional Inherited Members

13.48.1 Detailed Description

WossDb for custom made NetCDF marsden square DECK41 Sediment database.

SedimDeck41MarsdenDb implements WossNetcdfDb for the custom made NetCDF marsden square DECK41 Sediment database

13.48.2 Constructor & Destructor Documentation

13.48.2.1 SedimDeck41MarsdenDb() [1/2]

```
SedimDeck41MarsdenDb::SedimDeck41MarsdenDb (
    const ::std::string & name = DB_NAME_NOT_SET )
```

SedimDeck41MarsdenDb constructor.

Parameters

name	pathname of database

References woss::DECK41_DB_V1_TYPE.

13.48.2.2 SedimDeck41MarsdenDb() [2/2]

```
SedimDeck41MarsdenDb::SedimDeck41MarsdenDb (
    const ::std::string & name,
    DECK41DbType db_type )
```

SedimDeck41MarsdenDb constructor.

Parameters

name	pathname of database
db_type	DECK41 db type

13.48.3 Member Function Documentation

13.48.3.1 finalizeConnection() `bool SedimDeck41MarsdenDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions, used to create and initialize NetCDF variables

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References deck41_db_type, woss::DECK41_DB_V2_TYPE, main_sedim_var_marsden, marsden_square_var, woss::WossNetcdfDb::netcdf_db, and sec_sedim_var_marsden.

13.48.3.2 getDeck41DbType() `DECK41DbType woss::SedimDeck41MarsdenDb::getDeck41DbType ( ) const [inline]`

Returns the current DECK41 DB type

Returns

current deck41 db type

References deck41_db_type.

13.48.3.3 getSeaFloorType() `Deck41Types SedimDeck41MarsdenDb::getSeaFloorType ( const Marsden & marsden_square ) const`

Returns a Deck41Types for the given coordinates

Parameters

coordinates	const reference to a valid Marsden object

Returns

a Deck41Types value (main type value, secondary type value)

References woss::WossDb::debug, deck41_db_type, woss::DECK41_DB_V1_TYPE, woss::DECK41_DB_V2_TYPE, main_sedim_var_marsden, marsden_square_var, and sec_sedim_var_marsden.

Referenced by woss::SedimDeck41Db::getDeck41TypesFromMarsdenSquare().

13.48.3.4 isValid() `virtual bool woss::SedimDeck41MarsdenDb::isValid ( ) [inline], [virtual]`

Checks the validity of SedimDeck41MarsdenDb

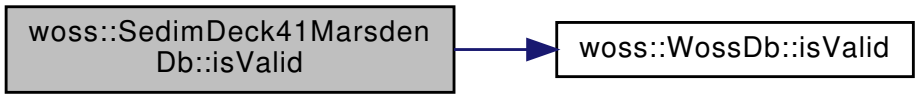
Returns

true if pathname is valid, *false* otherwise

Reimplemented from woss::WossDb.

References woss::WossDb::db_name, woss::DECK41_DB_INVALID_TYPE, deck41_db_type, and woss::WossDb::isValid().

Here is the call graph for this function:



13.48.3.5 setDeck41DbType() `void woss::SedimDeck41MarsdenDb::setDeck41DbType ( DECK41DbType db_type ) [inline]`

Sets the current DECK41 Db type

Parameters

db_type	DECK41DbType

References deck41_db_type.

Referenced by woss::SedimDeck41DbCreator::initializeSedimDb().

13.48.4 Member Data Documentation

13.48.4.1 deck41_db_type `DECK41DbType woss::SedimDeck41MarsdenDb::deck41_db_type [protected]`

DECK41 database type

Referenced by finalizeConnection(), getDeck41DbType(), getSeaFloorType(), isValid(), and setDeck41DbType().

13.48.4.2 main_sedim_var_marsden `netCDF::NcVar woss::SedimDeck41MarsdenDb::main_sedim_var_↩ marsden [protected]`

NetCDF variable representing main DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

13.48.4.3 marsden_square_var `netCDF::NcVar woss::SedimDeck41MarsdenDb::marsden_square_var [protected]`

NetCDF marsden square variable

Referenced by finalizeConnection(), and getSeaFloorType().

13.48.4.4 sec_sedim_var_marsden `netCDF::NcVar woss::SedimDeck41MarsdenDb::sec_sedim_var_↩ marsden [protected]`

NetCDF variable representing secondary DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

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

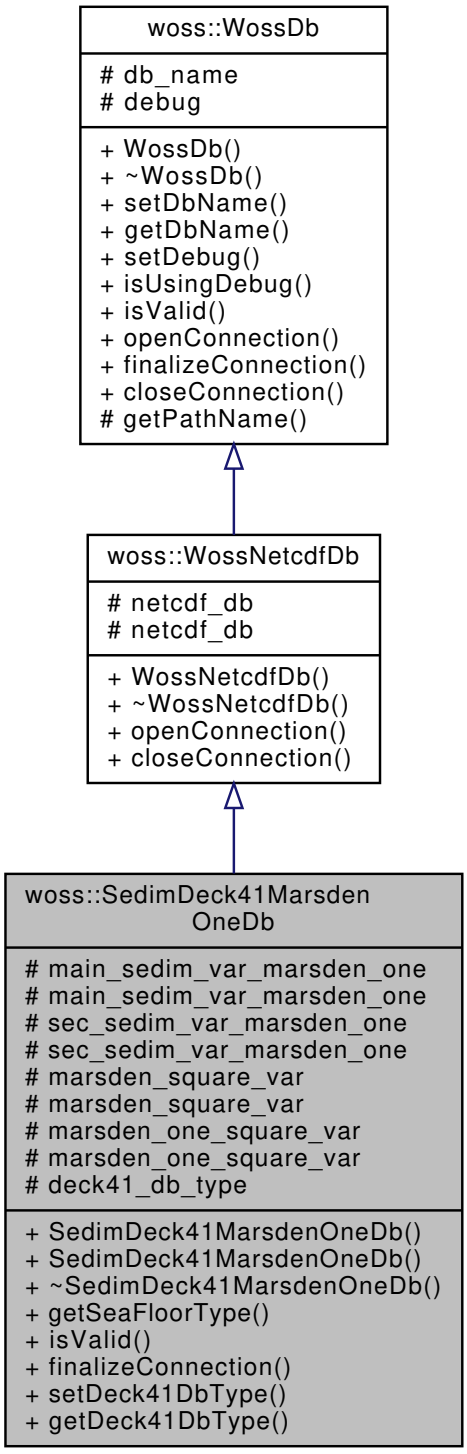
- woss/woss_db/sediment-deck41-marsden-db.h
- woss/woss_db/sediment-deck41-marsden-db.cpp

13.49 woss::SedimDeck41MarsdenOneDb Class Reference

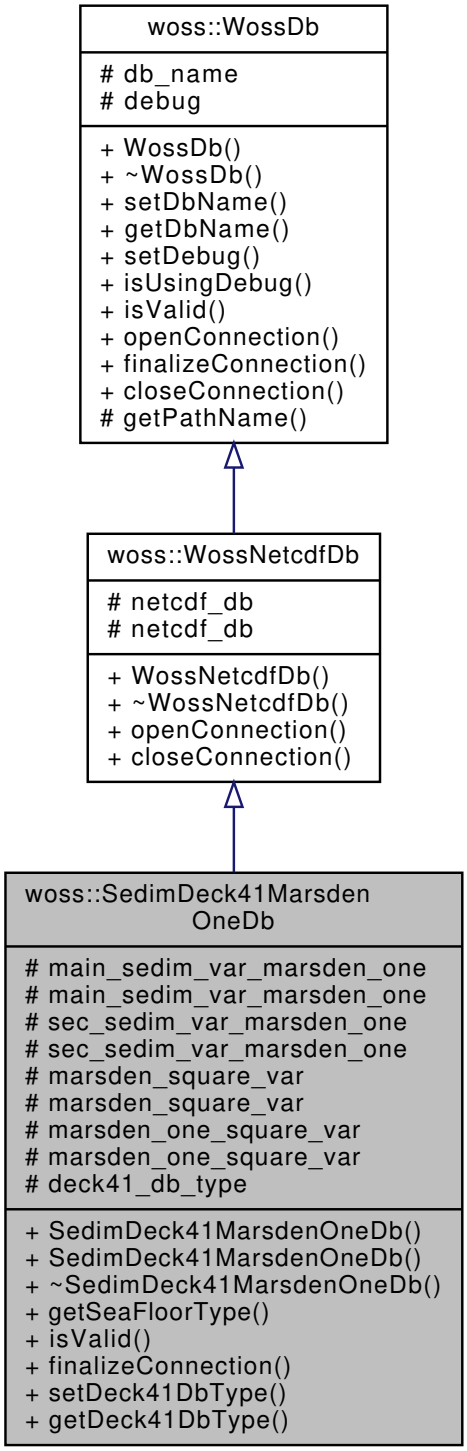
WossDb for custom made NetCDF marsden coordinates DECK41 Sediment database.

`#include <sediment-deck41-marsden-one-db.h>`

Inheritance diagram for woss::SedimDeck41MarsdenOneDb:



Collaboration diagram for woss::SedimDeck41MarsdenOneDb:



Public Member Functions

- SedimDeck41MarsdenOneDb (const ::std::string &name=DB_NAME_NOT_SET)
- SedimDeck41MarsdenOneDb (const ::std::string &name, DECK41DbType db_type)
- Deck41Types getSeaFloorType (const MarsdenCoord &marsden_coord) const
- virtual bool isValid ()
- virtual bool finalizeConnection ()
- void setDeck41DbType (DECK41DbType db_type)
- DECK41DbType getDeck41DbType () const

Protected Attributes

- netCDF::NcVar main_sedim_var_marsden_one
- NcVar * **main_sedim_var_marsden_one**
- netCDF::NcVar sec_sedim_var_marsden_one
- NcVar * **sec_sedim_var_marsden_one**
- netCDF::NcVar marsden_square_var
- NcVar * **marsden_square_var**
- netCDF::NcVar marsden_one_square_var
- NcVar * **marsden_one_square_var**
- DECK41DbType deck41_db_type

Additional Inherited Members

13.49.1 Detailed Description

WossDb for custom made NetCDF marsden coordinates DECK41 Sediment database.

SedimDeck41MarsdenOneDb implements WossNetcdfDb for the custom made NetCDF marsden coordinates DECK41 Sediment database

13.49.2 Constructor & Destructor Documentation

13.49.2.1 SedimDeck41MarsdenOneDb() [1/2]

```
SedimDeck41MarsdenOneDb::SedimDeck41MarsdenOneDb (
    const ::std::string & name = DB_NAME_NOT_SET )
```

SedimDeck41MarsdenOneDb default constructor. Default constructed object are not valid

Parameters

name	pathname of database
db_type	DECK41 db type

References woss::DECK41_DB_V1_TYPE.

13.49.2.2 SedimDeck41MarsdenOneDb() [2/2]

```
SedimDeck41MarsdenOneDb::SedimDeck41MarsdenOneDb (
    const ::std::string & name,
    DECK41DbType db_type )
```

SedimDeck41MarsdenOneDb default constructor. Default constructed object are not valid

Parameters

name	pathname of database
db_type	DECK41 db type

13.49.3 Member Function Documentation

13.49.3.1 finalizeConnection() `bool SedimDeck41MarsdenOneDb::finalizeConnection ( ) [virtual]`

Post openConnection() actions, used to create and initialize NetCDF variables

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References deck41_db_type, woss::DECK41_DB_V2_TYPE, main_sedim_var_marsden_one, marsden_one_square_var, marsden_square_var, woss::WossNetcdfDb::netcdf_db, and sec_sedim_var_marsden_one.

13.49.3.2 getDeck41DbType() `DECK41DbType woss::SedimDeck41MarsdenOneDb::getDeck41DbType ( ) const [inline]`

Returns the current DECK41 DB type

Returns

current deck41 db type

References deck41_db_type.

13.49.3.3 getSeaFloorType() `Deck41Types SedimDeck41MarsdenOneDb::getSeaFloorType ( const MarsdenCoord & marsden_coord ) const`

Returns a Deck41Types for the given coordinates

Parameters

coordinates	const reference to a valid MarsdenCoord object

Returns

a Deck41Types value (main type value, secondary type value)

References woss::WossDb::debug, deck41_db_type, woss::DECK41_DB_V1_TYPE, woss::DECK41_DB_V2_TYPE, main_sedim_var_marsden_one, marsden_one_square_var, marsden_square_var, and sec_sedim_var_marsden_one.

Referenced by woss::SedimDeck41Db::getDeck41TypesFromMarsdenCoords().

13.49.3.4 isValid() `virtual bool woss::SedimDeck41MarsdenOneDb::isValid ( ) [inline], [virtual]`

Checks the validity of SedimDeck41MarsdenOneDb

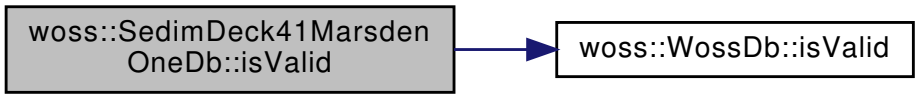
Returns

true if pathname is valid, *false* otherwise

Reimplemented from woss::WossDb.

References woss::WossDb::db_name, woss::DECK41_DB_INVALID_TYPE, deck41_db_type, and woss::WossDb::isValid().

Here is the call graph for this function:



13.49.3.5 setDeck41DbType() `void woss::SedimDeck41MarsdenOneDb::setDeck41DbType ( DECK41DbType db_type ) [inline]`

Sets the current DECK41 Db type

Parameters

db_type	DECK41DbType

References deck41_db_type.

Referenced by woss::SedimDeck41DbCreator::initializeSedimDb().

13.49.4 Member Data Documentation

13.49.4.1 deck41_db_type `DECK41DbType woss::SedimDeck41MarsdenOneDb::deck41_db_type [protected]`

DECK41 database type

Referenced by finalizeConnection(), getDeck41DbType(), getSeaFloorType(), isValid(), and setDeck41DbType().

13.49.4.2 main_sedim_var_marsden_one `netCDF::NcVar woss::SedimDeck41MarsdenOneDb::main_↩ sedim_var_marsden_one [protected]`

NetCDF variable representing main DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

13.49.4.3 marsden_one_square_var `netCDF::NcVar woss::SedimDeck41MarsdenOneDb::marsden_one_↩ square_var [protected]`

NetCDF marsden one degree square variable

Referenced by finalizeConnection(), and getSeaFloorType().

13.49.4.4 marsden_square_var `netCDF::NcVar woss::SedimDeck41MarsdenOneDb::marsden_square_var [protected]`

NetCDF marsden square variable

Referenced by finalizeConnection(), and getSeaFloorType().

13.49.4.5 sec_sedim_var_marsden_one `netCDF::NcVar woss::SedimDeck41MarsdenOneDb::sec_sedim↩ _var_marsden_one [protected]`

NetCDF variable representing secondary DECK41 floortype

Referenced by finalizeConnection(), and getSeaFloorType().

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

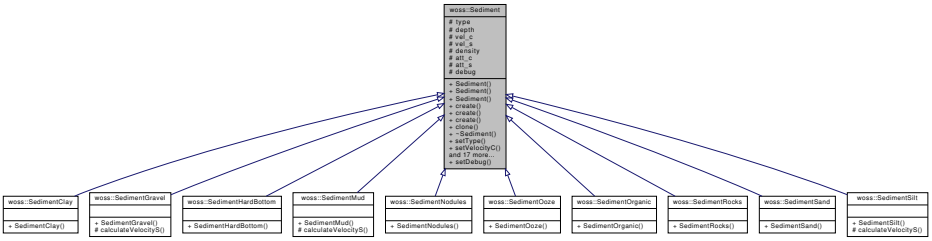
- woss/woss_db/sediment-deck41-marsden-one-db.h
- woss/woss_db/sediment-deck41-marsden-one-db.cpp

13.50 woss::Sediment Class Reference

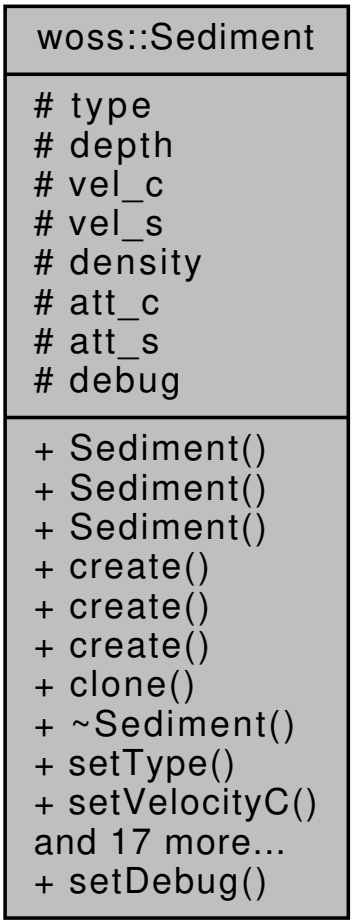
Surficial sediment geoacoustic parameters definitions.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::Sediment:



Collaboration diagram for woss::Sediment:



Public Member Functions

- Sediment ()
- Sediment (const ::std::string &name, double velc, double vels, double dens, double attc, double atts, double bottom_depth=1.0)

- Sediment (const Sediment &copy)
- virtual Sediment * create () const
- virtual Sediment * create (const ::std::string &name, double velc, double vels, double dens, double attc, double atts, double bottom_depth=1.0) const
- virtual Sediment * create (const Sediment &copy) const
- virtual Sediment * clone () const
- Sediment & setType (const ::std::string &name)
- Sediment & setVelocityC (double vel)
- Sediment & setVelocityS (double vel)
- Sediment & setDensity (double dens)
- Sediment & setAttenuationC (double att)
- Sediment & setAttenuationS (double att)
- Sediment & setDepth (double bottom_depth)
- Sediment & set (const ::std::string &name, double velc, double vels, double dens, double attc, double atts, double bottom_depth)
- double getVelocityC () const
- double getVelocityS () const
- double getDensity () const
- double getAttenuationC () const
- double getAttenuationS () const
- double getDepth () const
- ::std::string getType () const
- virtual bool isValid () const
- virtual const ::std::string getStringValues () const
- Sediment & operator= (const Sediment &time)
- friend::std::ostream & operator<< (::std::ostream &os, const Sediment &instance)

Static Public Member Functions

- static void setDebug (bool flag)

Protected Attributes

- ::std::string type
- double depth
- double vel_c
- double vel_s
- double density
- double att_c
- double att_s

Static Protected Attributes

- static bool debug = false

Friends

- const Sediment operator+ (const Sediment &left, const Sediment &right)
- const Sediment operator- (const Sediment &left, const Sediment &right)
- const Sediment operator/ (const Sediment &left, const Sediment &right)
- const Sediment operator∗ (const Sediment &left, const Sediment &right)
- Sediment & operator+= (Sediment &left, const Sediment &right)
- Sediment & operator-= (Sediment &left, const Sediment &right)
- Sediment & operator/= (Sediment &left, const Sediment &right)
- Sediment & operator∗= (Sediment &left, const Sediment &right)
- Sediment & operator+= (Sediment &left, double right)
- Sediment & operator-= (Sediment &left, double right)
- Sediment & operator/= (Sediment &left, double right)
- Sediment & operator∗= (Sediment &left, double right)
- bool operator== (const Sediment &left, const Sediment &right)
- bool operator!= (const Sediment &left, const Sediment &right)
- const Sediment operator+ (const double left, const Sediment &right)
- const Sediment operator- (const double left, const Sediment &right)
- const Sediment operator/ (const double left, const Sediment &right)
- const Sediment operator∗ (const double left, const Sediment &right)
- const Sediment operator+ (const Sediment &left, double right)
- const Sediment operator- (const Sediment &left, double right)
- const Sediment operator/ (const Sediment &left, double right)
- const Sediment operator∗ (const Sediment &left, double right)

13.50.1 Detailed Description

Surficial sediment geoacoustic parameters definitions.

The Sediment class provide an interface for creating and manipulating surficialg eoacoustic parameters.

13.50.2 Constructor & Destructor Documentation

13.50.2.1 Sediment() [1/3] Sediment::Sediment ( )

Sediment default constructor. The object created is not valid

Referenced by clone(), and create().

13.50.2.2 Sediment() [2/3] Sediment::Sediment (

```
const ::std::string & name,
double velc,
double vels,
double dens,
double attc,
double atts,
double bottom_depth = 1.0 )
```

Sediment constructor

Parameters

name	textual name
velc	compressional wave velocity [m/s]
vels	shear wave velocity [m/s]
dens	sediment density [g/cm^3 or user defined]
attc	compressional wave attenuation [db/wavelength or user defined]
atts	shear wave attenuation [db/wavelength or user defined]
bottom_depth	bottom depth [m]

13.50.2.3 Sediment() [3/3] Sediment::Sediment (const Sediment & *copy*)

Sediment copy constructor

Parameters

copy	Sediment to be copied

References att_c, att_s, density, depth, type, vel_c, and vel_s.

13.50.3 Member Function Documentation

13.50.3.1 clone() virtual Sediment * woss::Sediment::clone ( ) const [inline], [virtual]

Sediment virtual factory method

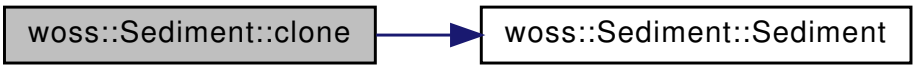
Returns

a heap-created copy of **this** instance

References Sediment().

Referenced by woss::DefHandler::operator=().

Here is the call graph for this function:



13.50.3.2 create() [1/3] virtual Sediment * woss::Sediment::create ( ) const [inline], [virtual]

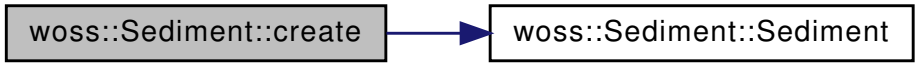
Sediment virtual factory method

Returns

a heap-created Sediment object

References Sediment().

Here is the call graph for this function:



13.50.3.3 create() [2/3] virtual Sediment * woss::Sediment::create (

```
const ::std::string & name,
double velc,
double vels,
double dens,
double attc,
double atts,
double bottom_depth = 1.0 ) const [inline], [virtual]
```

Sediment virtual factory method

Parameters

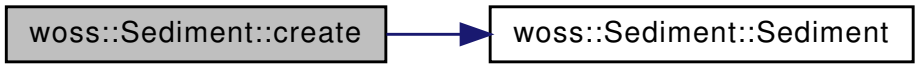
name	textual name
velc	compressional wave velocity [m/s]
vels	shear wave velocity [m/s]
dens	sediment density [g/cm^3 or user defined]
attc	compressional wave attenuation [db/wavelength or user defined]
atts	shear wave attenuation [db/wavelength or user defined]
bottom_depth	bottom depth [m]

Returns

a heap-created Sediment object

References Sediment().

Here is the call graph for this function:



13.50.3.4 create() [3/3] virtual Sediment * woss::Sediment::create (const Sediment & *copy*) const [inline], [virtual]

Sediment virtual factory method

Parameters

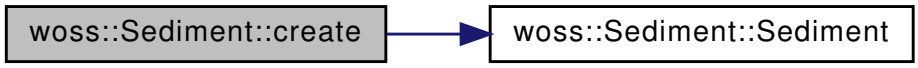
copy	Sediment to be copied

Returns

a heap-created Sediment object

References Sediment().

Here is the call graph for this function:



13.50.3.5 getAttenuationC() double woss::Sediment::getAttenuationC ( ) const [inline]

Gets compressional wave attenuation

Returns

attenuation [db/wavelength or user defined]

References att_c.

13.50.3.6 getAttenuationS() double woss::Sediment::getAttenuationS ( ) const [inline]

Gets shear wave attenuation

Returns

attenuation [db/wavelength or user defined]

References att_s.

13.50.3.7 getDensity() `double woss::Sediment::getDensity ( ) const [inline]`

Gets sediment density

Returns

density [g/cm^3 or user defined]

References density.

13.50.3.8 getDepth() `double woss::Sediment::getDepth ( ) const [inline]`

Gets bottom depth

Returns

depth [m]

References depth.

13.50.3.9 getStringValues() `const ::std::string Sediment::getStringValues ( ) const [virtual]`

Gets geoacoustic parameters in a formatted fashion

Returns

formatted string

References att_c, att_s, density, vel_c, and vel_s.

Referenced by woss::BellhopWoss::writeSediment().

13.50.3.10 getType() `::std::string woss::Sediment::getType ( ) const [inline]`

Gets sediment type name

Returns

type

References type.

Referenced by woss::WossDbManager::getSediment(), and woss::BellhopWoss::writeSediment().

13.50.3.11 getVelocityC() `double woss::Sediment::getVelocityC ( ) const [inline]`

Gets compressional wave velocity

Returns

velocity [m/s]

References vel_c.

13.50.3.12 getVelocityS() `double woss::Sediment::getVelocityS ( ) const [inline]`

Gets shear wave velocity

Returns

velocity [m/s]

References vel_s.

13.50.3.13 isValid() `virtual bool woss::Sediment::isValid ( ) const [inline], [virtual]`

Checks the validity of geoacoustic parameters provided

Returns

true if Sediment is valid, *false* otherwise

References att_c, att_s, density, vel_c, and vel_s.

Referenced by woss::ACToolboxWoss::initSediment().

13.50.3.14 operator<<() `friend::std::ostream & woss::Sediment::operator<< ( ::std::ostream & os, const Sediment & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Sediment reference

Returns

os reference after the operation

13.50.3.15 operator=() `Sediment & Sediment::operator= ( const Sediment & time )`

Assignment operator

Parameters

copy	const reference to a Sediment object to be copied

Returns

Sediment reference to *this*

References att_c, att_s, density, depth, type, vel_c, and vel_s.

13.50.3.16 set() `Sediment & woss::Sediment::set ( const ::std::string & name, double velc, double vels, double dens, double attc, double atts, double bottom_depth ) [inline]`

Sets all parameters at one.

Parameters

name	textual name
velc	compressional wave velocity [m/s]
vels	shear wave velocity [m/s]
dens	sediment density [g/cm^3 or user defined]
attc	compressional wave attenuation [db/wavelength or user defined]
atts	shear wave attenuation [db/wavelength or user defined]
bottom_depth	bottom depth ($>$ 0) [m]

Returns

reference to $*$**this**

References att_c, att_s, density, depth, type, vel_c, and vel_s.

13.50.3.17 setAttenuationC() Sediment & woss::Sediment::setAttenuationC (double *att*) [inline]

Sets compressional wave attenuation

Parameters

att	attenuation [db/wavelength or user defined]

Returns

reference to ∗**this**

References att_c.

13.50.3.18 setAttenuationS() Sediment & woss::Sediment::setAttenuationS (double *att*) [inline]

Sets shear wave attenuation

Parameters

att	attenuation [db/wavelength or user defined]

Returns

reference to ∗**this**

References att_s.

13.50.3.19 setDebug() static void woss::Sediment::setDebug (bool *flag*) [inline], [static]

Sets debug for the whole class

Parameters

flag	debug value

References debug.

13.50.3.20 setDensity() Sediment & woss::Sediment::setDensity (double *dens*) [inline]

Sets sediment density

Parameters

dens	density [g/cm^3 or user defined]

Returns

reference to ∗**this**

References density.

13.50.3.21 setDepth() `Sediment & woss::Sediment::setDepth ( double bottom_depth ) [inline]`

Sets bottom depth

Parameters

bottom_depth	positive depth [m/s]

Returns

reference to ∗**this**

References depth.

13.50.3.22 setType() `Sediment & woss::Sediment::setType ( const ::std::string & name ) [inline]`

Sets textual type name

Parameters

name	string name

Returns

reference to ∗**this**

References type.

13.50.3.23 setVelocityC() `Sediment & woss::Sediment::setVelocityC ( double vel ) [inline]`

Sets compressional wave velocity

Parameters

vel	velocity [m/s]

Returns

reference to ∗**this**

References vel_c.

13.50.3.24 setVelocityS() `Sediment & woss::Sediment::setVelocityS ( double vel ) [inline]`

Sets shear wave velocity

Parameters

vel	velocity [m/s]

Returns

reference to ∗**this**

References vel_s.

13.50.4 Friends And Related Function Documentation

13.50.4.1 operator"!= `bool operator!= ( const Sediment & left, const Sediment & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.50.4.2 operator* [1/3] const Sediment operator* (const double *left*, const Sediment & *right*) [friend]

Scalar multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.3 operator* [2/3] const Sediment operator* (const Sediment & *left*, const Sediment & *right*) [friend]

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.4 operator* [3/3] const Sediment operator* (const Sediment & *left*, double *right*) [friend]

Scalar multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.5 operator*= [1/2]

```
Sediment & operator*= (
        Sediment & left,
        const Sediment & right )  [friend]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.6 operator*= [2/2]

```
Sediment & operator*= (
        Sediment & left,
        double right )  [friend]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.7 operator+ [1/3]

```
const Sediment operator+ (
        const double left,
        const Sediment & right )  [friend]
```

Scalar sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.8 operator+ [2/3] const Sediment operator+ (

```
const Sediment & left,
const Sediment & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.9 operator+ [3/3] const Sediment operator+ (

```
const Sediment & left,
double right )  [friend]
```

Scalar sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.10 operator+= [1/2] Sediment & operator+= (

```
Sediment & left,
const Sediment & right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.11 operator+= [2/2] Sediment & operator+= (Sediment & *left*, double *right*) [friend]

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.12 operator- [1/3] const Sediment operator- (const double *left*, const Sediment & *right*) [friend]

Scalar subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.13 operator- [2/3]

```
const Sediment operator- (
    const Sediment & left,
    const Sediment & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.14 operator- [3/3]

```
const Sediment operator- (
    const Sediment & left,
    double right )  [friend]
```

Scalar subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.15 operator-= [1/2]

```
Sediment & operator-= (
    Sediment & left,
    const Sediment & right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.16 operator-= [2/2]

```
Sediment & operator-= (
    Sediment & left,
    double right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.17 operator/ [1/3]

```
const Sediment operator/ (
    const double left,
    const Sediment & right )  [friend]
```

Scalar division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.18 operator/ [2/3]

```
const Sediment operator/ (
    const Sediment & left,
    const Sediment & right )  [friend]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.19 operator/ [3/3] const Sediment operator/ (const Sediment & *left*, double *right*) [friend]

Scalar division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.50.4.20 operator/= [1/2] Sediment & operator/= (Sediment & *left*, const Sediment & *right*) [friend]

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.21 operator/= [2/2] Sediment & operator/= (Sediment & *left*, double *right*) [friend]

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.50.4.22 operator== `bool operator== ( const Sediment & left, const Sediment & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.50.5 Member Data Documentation

13.50.5.1 att_c `double woss::Sediment::att_c [protected]`

Compressional wave attenuation [db/wavelength]

Referenced by getAttenuationC(), getStringValues(), isValid(), woss::operator∗(), woss::operator∗=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setAttenuationC().

13.50.5.2 att_s `double woss::Sediment::att_s [protected]`

Shear wave attenuation [db/wavelength]

Referenced by getAttenuationS(), getStringValues(), isValid(), woss::operator∗(), woss::operator∗=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setAttenuationS().

13.50.5.3 debug `bool Sediment::debug = false [static], [protected]`

Debug flag

Referenced by setDebug().

13.50.5.4 density `double woss::Sediment::density [protected]`

Sediment density [g/cm^3]

Referenced by getDensity(), getStringValues(), isValid(), woss::operator*(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setDensity().

13.50.5.5 depth `double woss::Sediment::depth [protected]`

Bottom depth. Used for shear wave velocity calculations [m]

Referenced by getDepth(), operator=(), Sediment(), set(), and setDepth().

13.50.5.6 type `::std::string woss::Sediment::type [protected]`

Sediment type name

Referenced by getType(), woss::operator*(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setType().

13.50.5.7 vel_c `double woss::Sediment::vel_c [protected]`

Compressional wave velocity [m/s]

Referenced by getStringValues(), getVelocityC(), isValid(), woss::operator*(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setVelocityC().

13.50.5.8 vel_s `double woss::Sediment::vel_s [protected]`

Shear wave velocity [m/s]

Referenced by getStringValues(), getVelocityS(), isValid(), woss::operator*(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/(), woss::operator/=(), operator=(), Sediment(), set(), and setVelocityS().

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

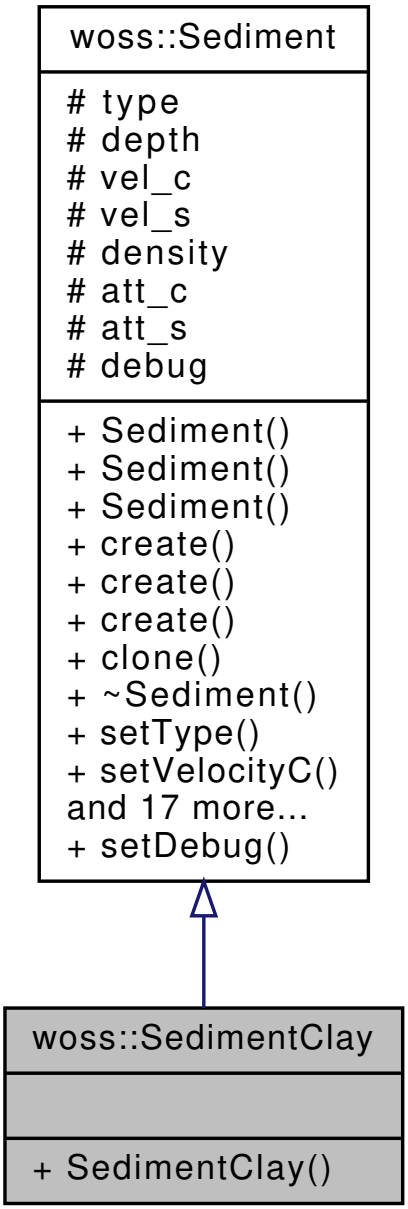
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.51 woss::SedimentClay Class Reference

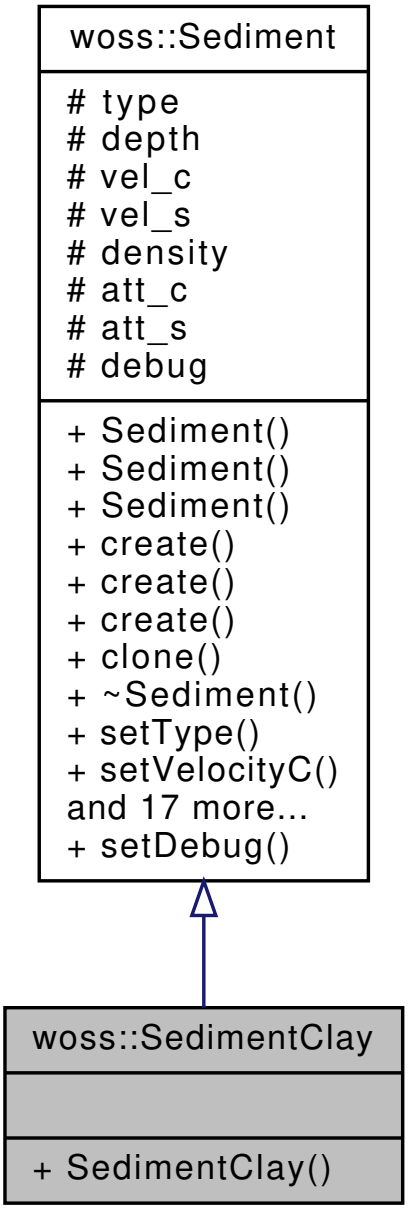
Clay type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentClay:



Collaboration diagram for woss::SedimentClay:



Additional Inherited Members

13.51.1 Detailed Description

Clay type implementation.

Literature clay implementation

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

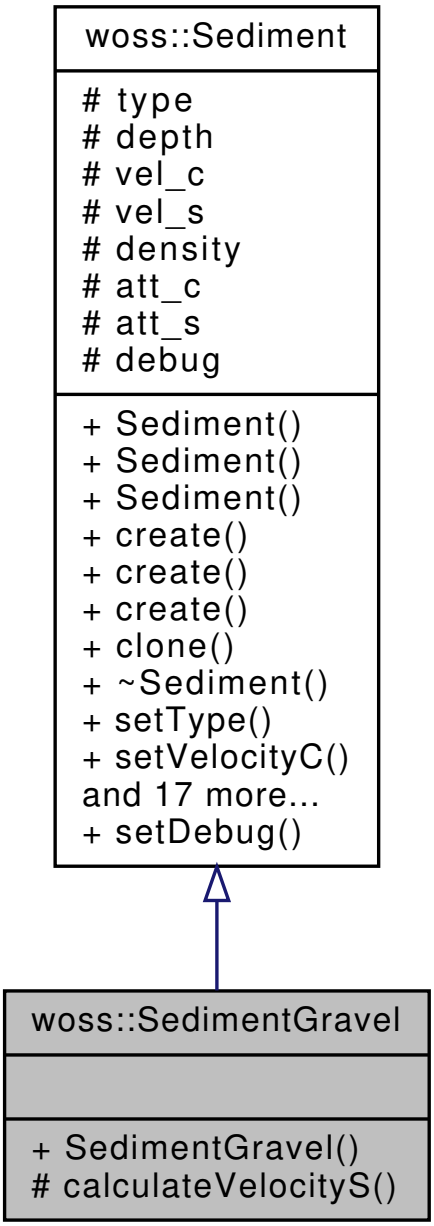
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.52 woss::SedimentGravel Class Reference

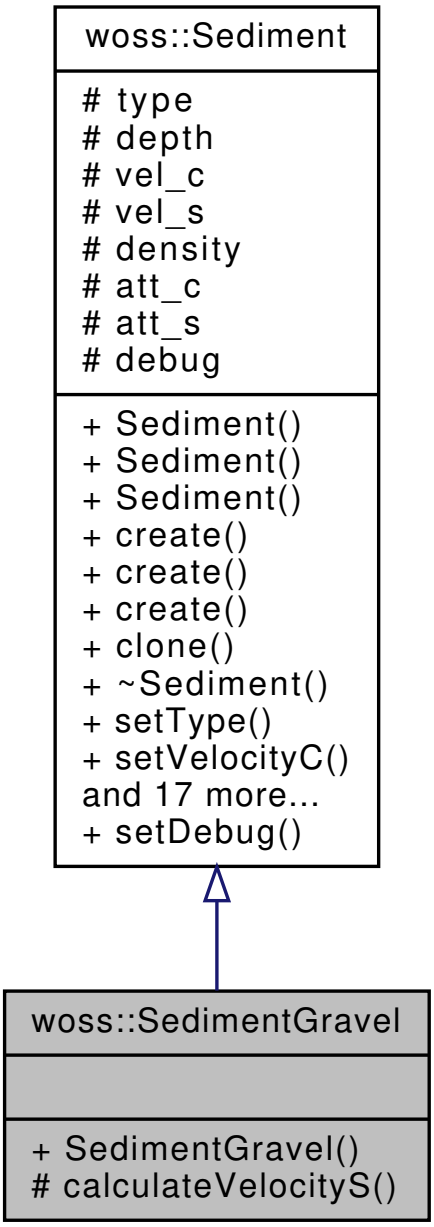
Gravel type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentGravel:



Collaboration diagram for woss::SedimentGravel:



Public Member Functions

- **SedimentGravel** (double depth=1.0)

Protected Member Functions

- double calculateVelocityS (double vels, double bottom_depth)

Additional Inherited Members

13.52.1 Detailed Description

Gravel type implementation.

Literature gravel implementation

13.52.2 Member Function Documentation

13.52.2.1 calculateVelocityS() `double SedimentGravel::calculateVelocityS ( double vels, double bottom_depth ) [protected]`

Shear wave velocity formula taken from the literature

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

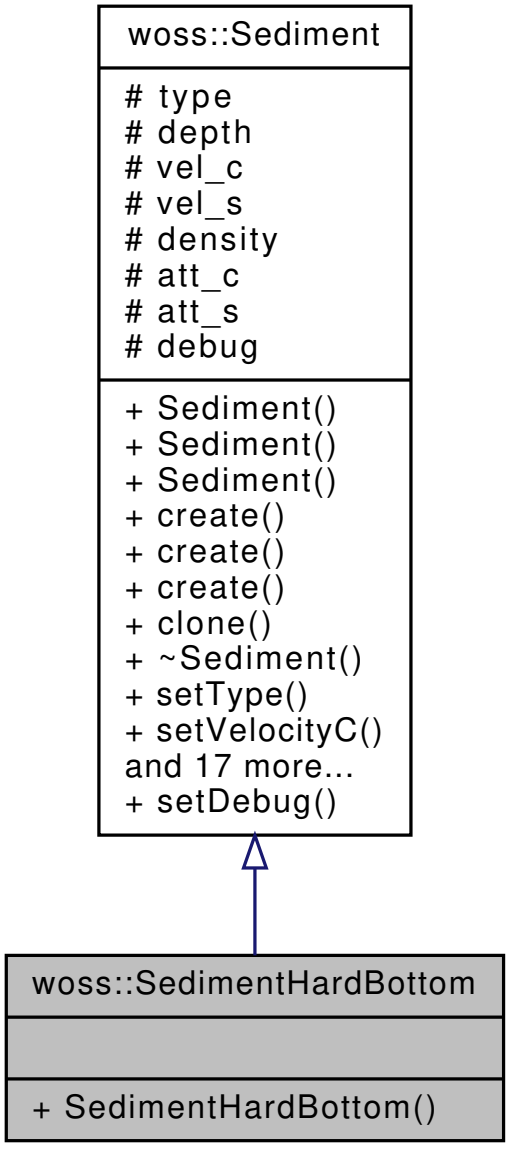
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.53 woss::SedimentHardBottom Class Reference

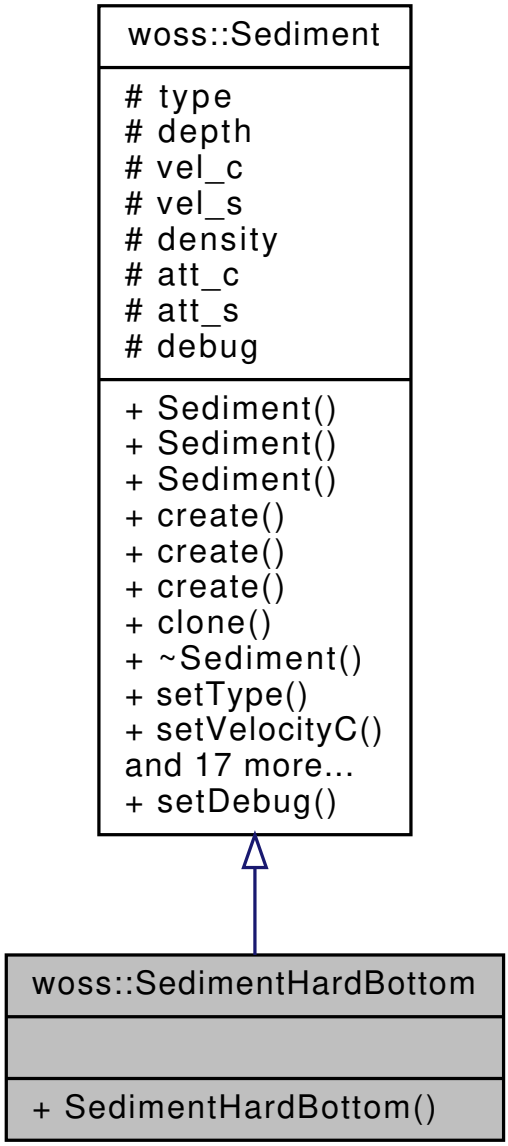
Hard bottom type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentHardBottom:



Collaboration diagram for woss::SedimentHardBottom:



Additional Inherited Members

13.53.1 Detailed Description

Hard bottom type implementation.

Literature hard bottom implementation

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

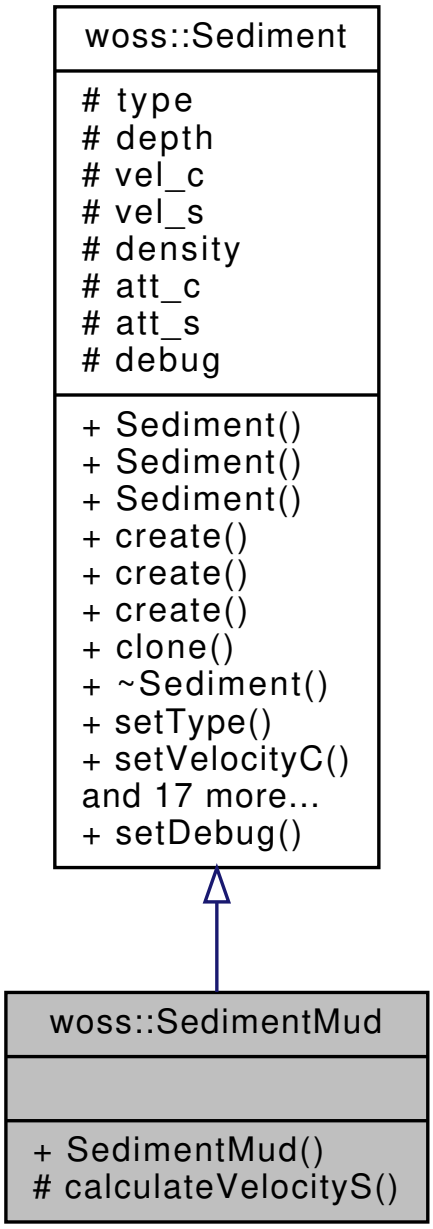
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.54 woss::SedimentMud Class Reference

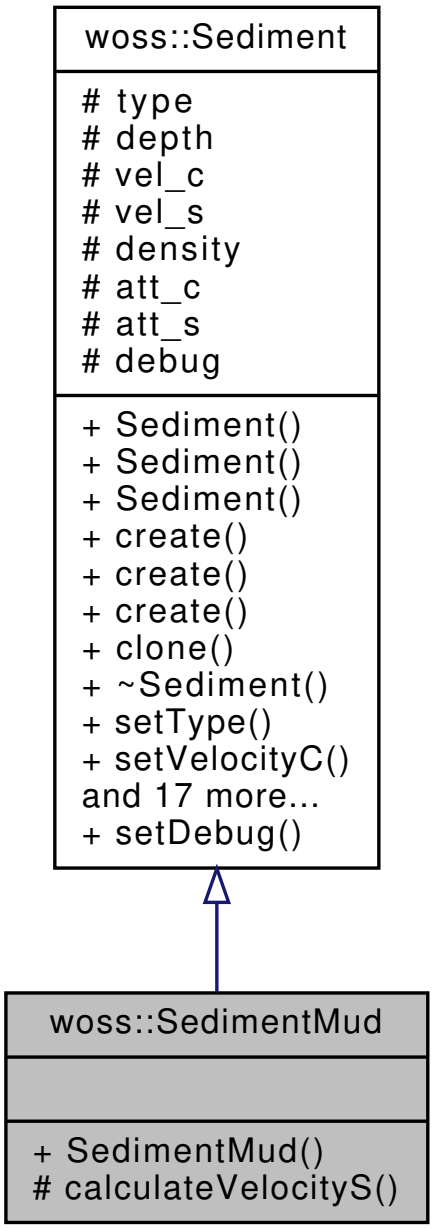
Mud type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentMud:



Collaboration diagram for woss::SedimentMud:



Public Member Functions

- **SedimentMud** (double depth=1.0)

Protected Member Functions

- double calculateVelocityS (double vels, double bottom_depth)

Additional Inherited Members

13.54.1 Detailed Description

Mud type implementation.

Literature mud implementation

13.54.2 Member Function Documentation

13.54.2.1 calculateVelocityS() `double SedimentMud::calculateVelocityS (`

```
double vels,
double bottom_depth )  [protected]
```

Shear wave velocity formula taken from the literature

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

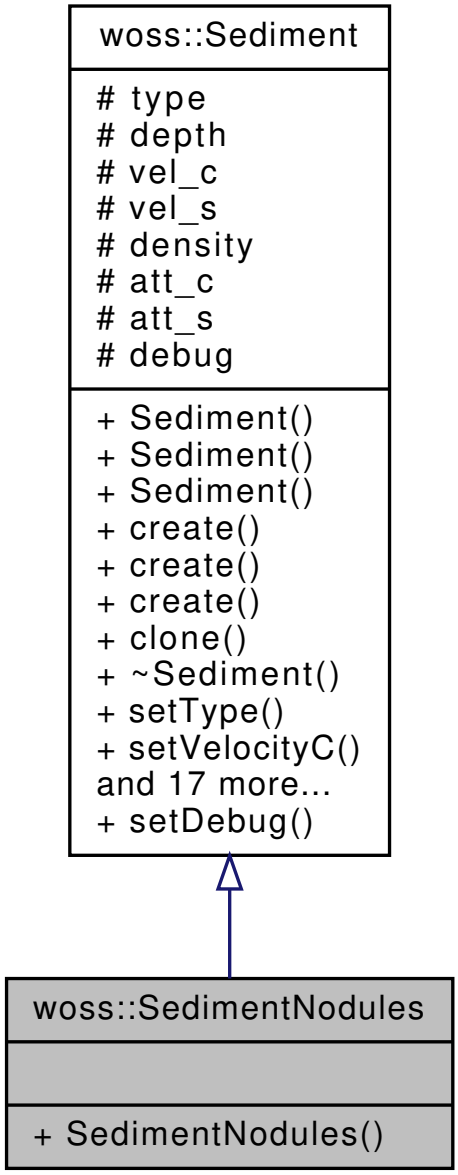
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.55 woss::SedimentNodules Class Reference

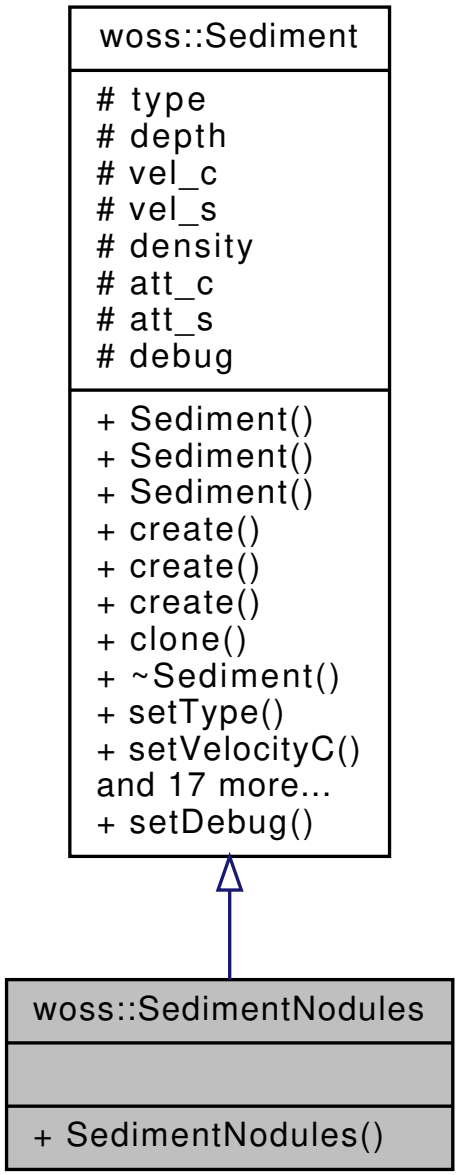
Deck41 nodules type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentNodules:



Collaboration diagram for woss::SedimentNodules:



Additional Inherited Members

13.55.1 Detailed Description

Deck41 nodules type implementation.

Literature Deck41 nodules implementation

Bug Needs a better geoacoustic representation

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

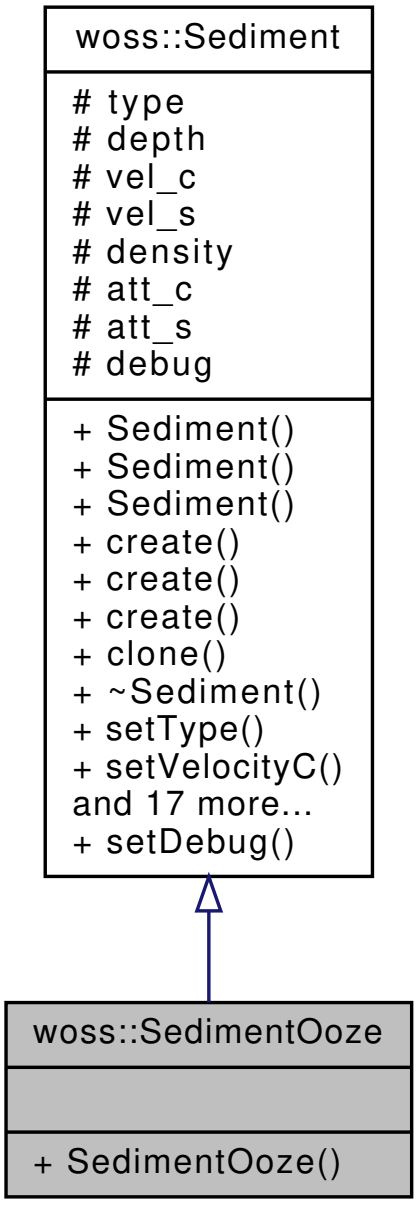
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.56 woss::SedimentOoze Class Reference

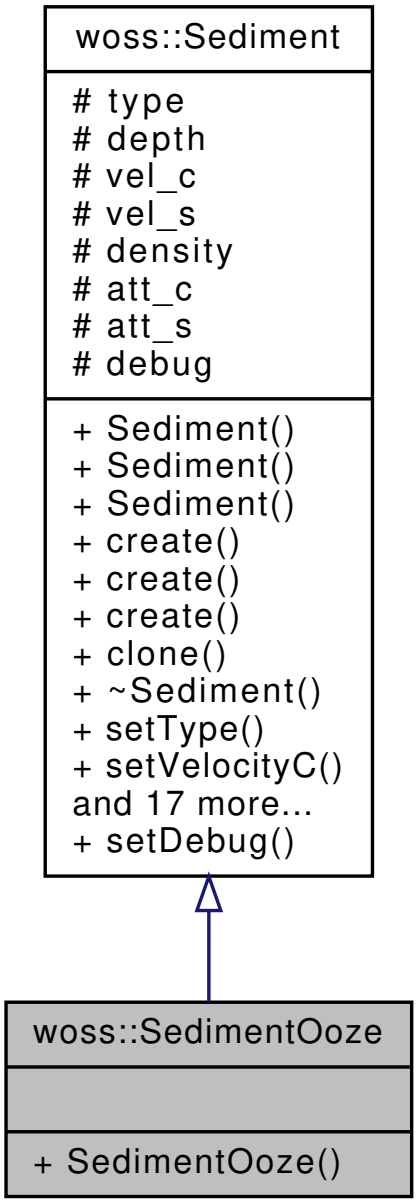
Ooze type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentOoze:



Collaboration diagram for woss::SedimentOoze:



Additional Inherited Members

13.56.1 Detailed Description

Ooze type implementation.

Literature ooze implementation

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

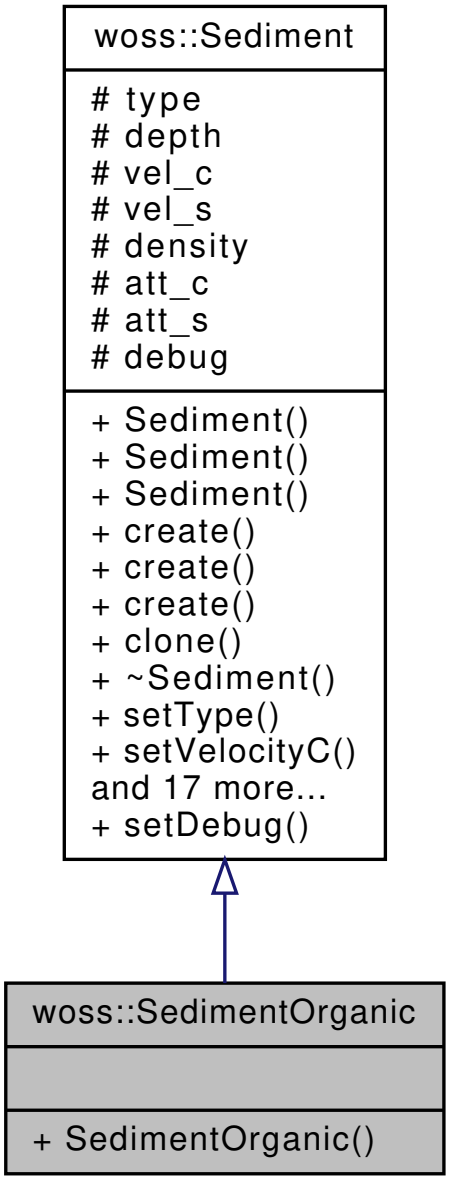
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.57 woss::SedimentOrganic Class Reference

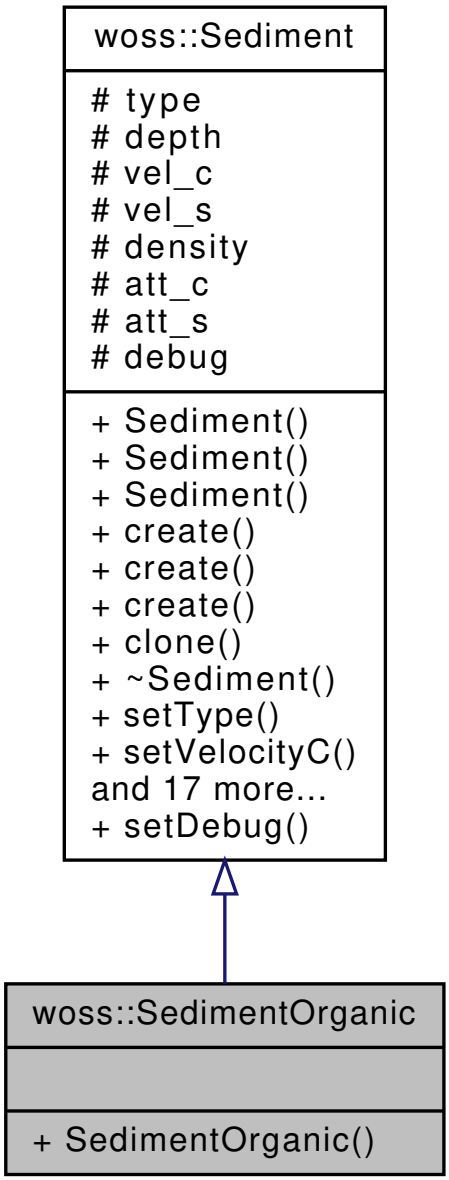
Organic type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentOrganic:



Collaboration diagram for woss::SedimentOrganic:



Additional Inherited Members

13.57.1 Detailed Description

Organic type implementation.

Literature organic implementation

Bug No geoacoustic parameters avalaible

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

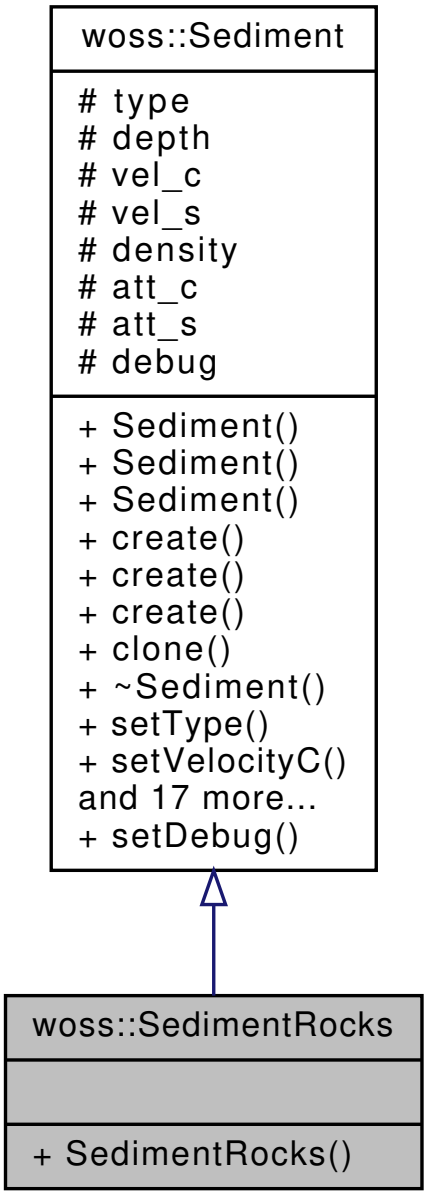
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.58 woss::SedimentRocks Class Reference

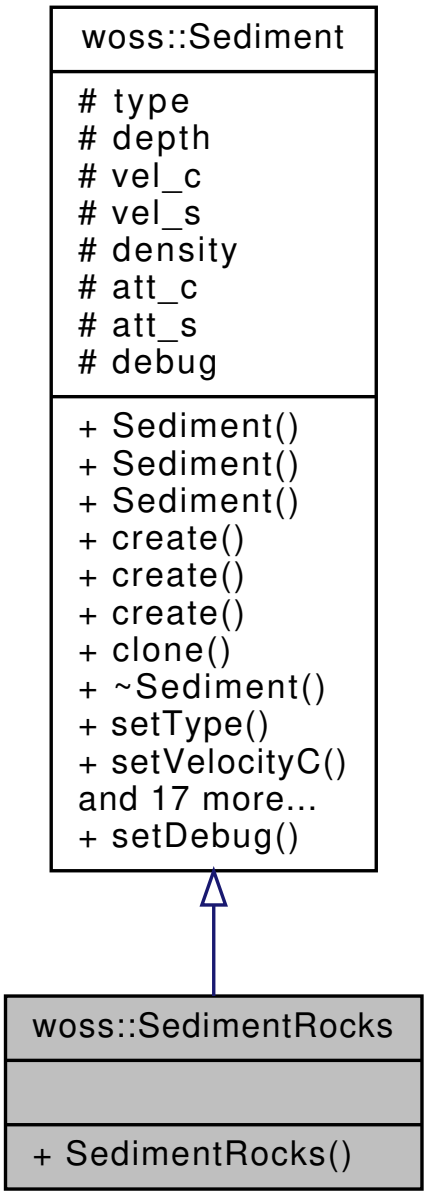
Rocks type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentRocks:



Collaboration diagram for woss::SedimentRocks:



Additional Inherited Members

13.58.1 Detailed Description

Rocks type implementation.

Literature rocks implementation

Bug Needs a better geoacoustic representation

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

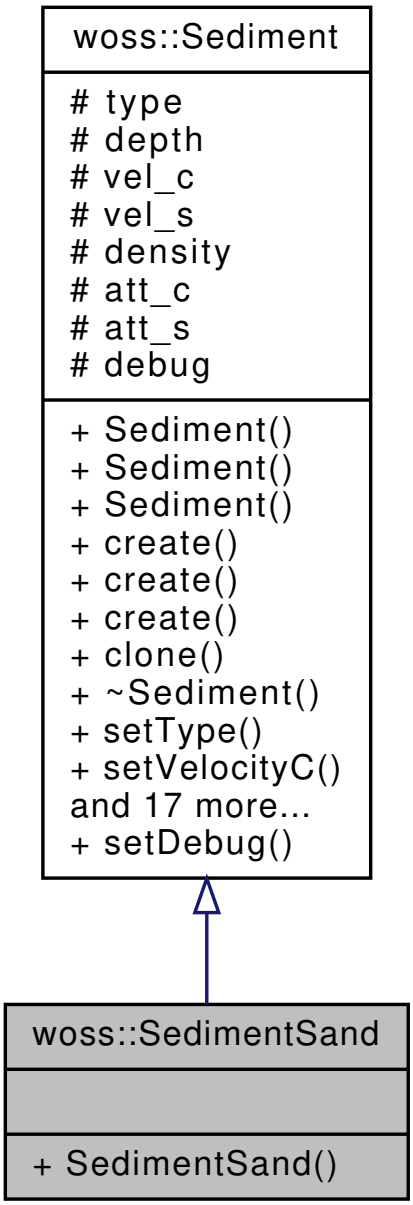
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.59 woss::SedimentSand Class Reference

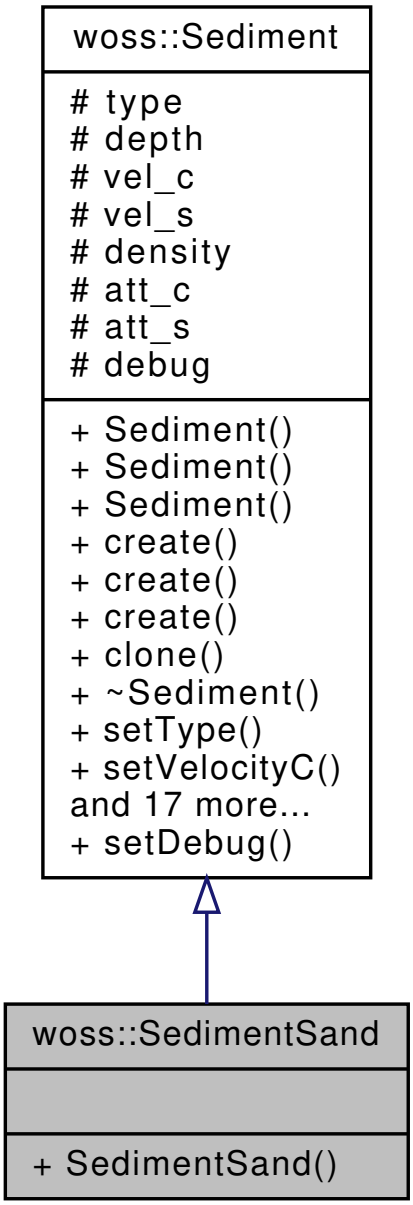
Sand type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentSand:



Collaboration diagram for woss::SedimentSand:



Additional Inherited Members

13.59.1 Detailed Description

Sand type implementation.

Literature sand implementation

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

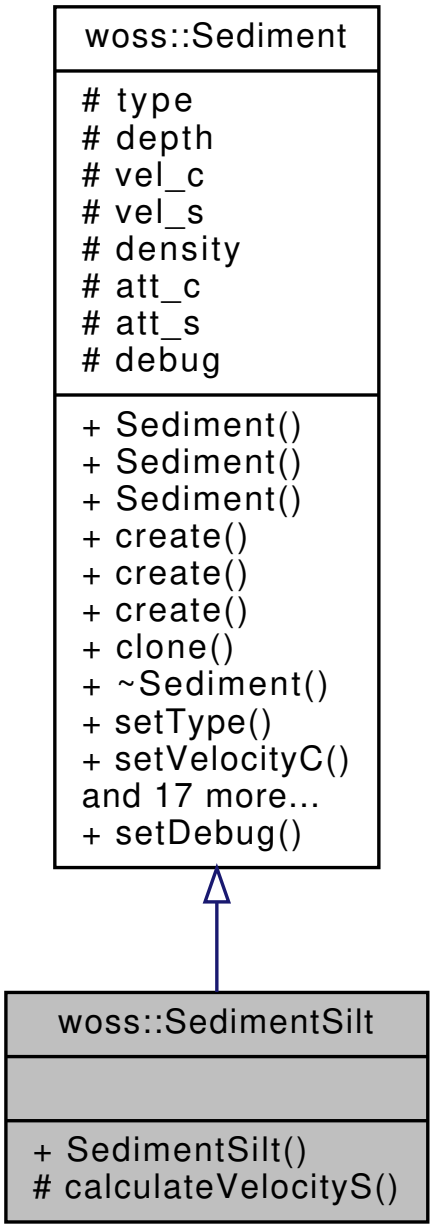
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.60 woss::SedimentSilt Class Reference

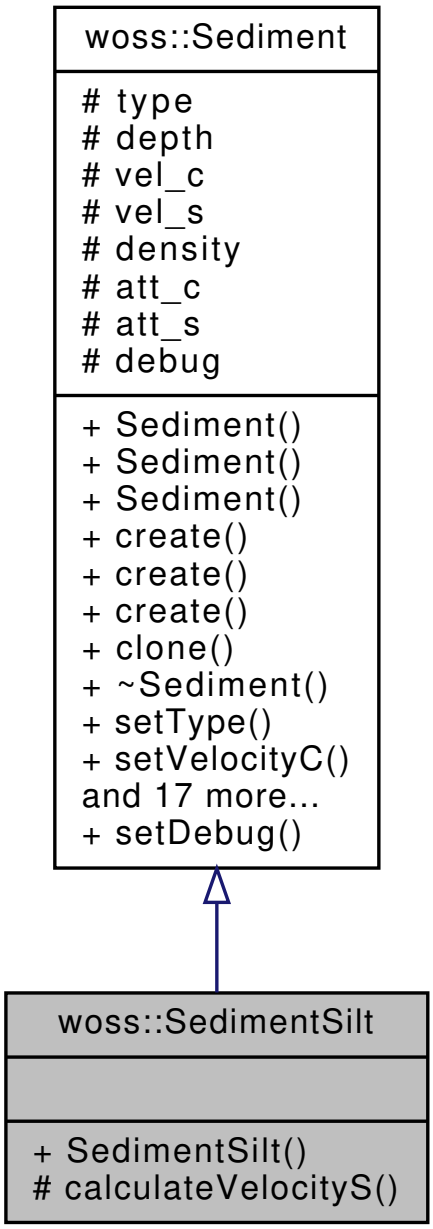
Silt type implementation.

`#include <sediment-definitions.h>`

Inheritance diagram for woss::SedimentSilt:



Collaboration diagram for woss::SedimentSilt:



Public Member Functions

- **SedimentSilt** (double depth=1.0)

Protected Member Functions

- double calculateVelocityS (double vels, double bottom_depth)

Additional Inherited Members

13.60.1 Detailed Description

Silt type implementation.

Literature silt implementation

13.60.2 Member Function Documentation

13.60.2.1 calculateVelocityS() `double SedimentSilt::calculateVelocityS ( double vels, double bottom_depth ) [protected]`

Shear wave velocity formula taken from the literature

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

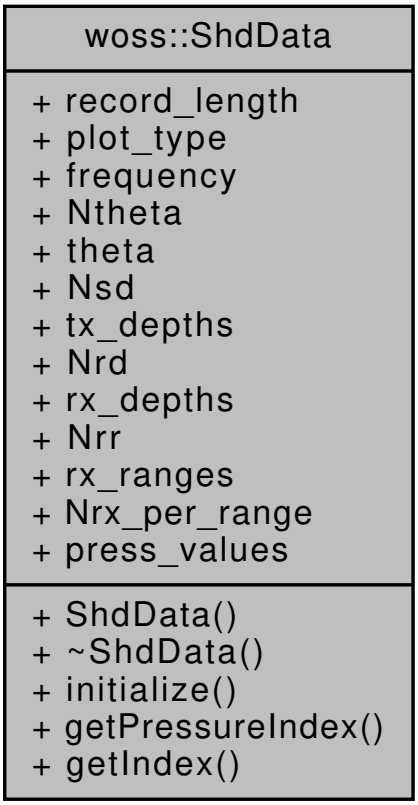
- woss/woss_def/sediment-definitions.h
- woss/woss_def/sediment-definitions.cpp

13.61 woss::ShdData Class Reference

class for storing data of any acoustic toolbox SHD file

`#include <ac-toolbox-shd-reader.h>`

Collaboration diagram for woss::ShdData:



Public Member Functions

- ~ShdData ()
- void initialize ()
- int getPressureIndex (double tx_depth, double rx_depth, double rx_range, double theta=0.0) const
- int getIndex (float value, float *array, int32_t array_size) const

Public Attributes

- int32_t record_length
- char * plot_type
- float frequency
- int32_t Ntheta
- float * theta
- int32_t Nsd
- float * tx_depths
- int32_t Nrd
- float * rx_depths
- int32_t Nrr
- float * rx_ranges
- int32_t Nrx_per_range
- ::std::complex< double > * press_values

13.61.1 Detailed Description

class for storing data of any acoustic toolbox SHD file

class ShdData stores Pressure provided by any acoustic toolbox SHD file

13.61.2 Constructor & Destructor Documentation

13.61.2.1 ~ShdData() `woss::ShdData::~ShdData ( ) [inline]`

Destructor

References press_values, rx_depths, rx_ranges, theta, and tx_depths.

13.61.3 Member Function Documentation

13.61.3.1 getIndex()

```
int ShdData::getIndex (
        float value,
        float * array,
        int32_t array_size ) const
```

Returns the index of given array associated to given value

Parameters

value	test value
array	valid pointer to an array
array_size	size of passed array

Returns

valid array index value

Referenced by getPressureIndex().

13.61.3.2 getPressureIndex()

```
int ShdData::getPressureIndex (
        double tx_depth,
        double rx_depth,
        double rx_range,
        double theta = 0.0 ) const
```

Returns the press_values index associated to given parameters

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]
theta	theta value

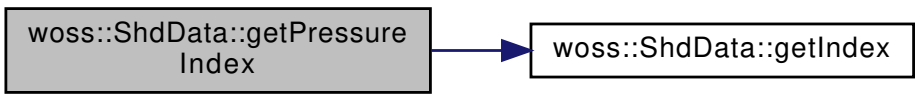
Returns

valid press_values index value

References getIndex(), Nrr, Nrx_per_range, Nsd, Ntheta, rx_depths, rx_ranges, theta, and tx_depths.

Referenced by woss::ShdResReader::accessMap(), and woss::ShdResReader::readMapAvgPressure().

Here is the call graph for this function:



13.61.3.3 initialize()

```
void woss::ShdData::initialize ( ) [inline]
```

Initializes the struct

References frequency, Nrd, Nrr, Nrx_per_range, Nsd, Ntheta, plot_type, press_values, record_length, rx_depths, rx_ranges, theta, and tx_depths.

13.61.4 Member Data Documentation

13.61.4.1 frequency `float woss::ShdData::frequency`

Frequency value [Hz]

Referenced by woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.2 Nrd `int32_t woss::ShdData::Nrd`

Total number of receiver depths

Referenced by woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.3 Nrr `int32_t woss::ShdData::Nrr`

Total number of receiver ranges

Referenced by getPressureIndex(), woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.4 Nrx_per_range `int32_t woss::ShdData::Nrx_per_range`

Total number of receiver per range

Referenced by getPressureIndex(), woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.5 Nsd `int32_t woss::ShdData::Nsd`

Total number of transmitter depths

Referenced by getPressureIndex(), woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.6 Ntheta `int32_t woss::ShdData::Ntheta`

Total number of theta values. See Bellhop code for more info

Referenced by getPressureIndex(), woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.7 plot_type `char* woss::ShdData::plot_type`

Plot typename. See Bellhop code for more info

Referenced by woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.8 press_values `::std::complex<double>* woss::ShdData::press_values`

Pointer to an array of complex<double> values [m]

Referenced by woss::ShdResReader::accessMap(), woss::ShdResReader::getShdHeader(), initialize(), woss::ShdResReader::readM and ~ShdData().

13.61.4.9 record_length `int32_t woss::ShdData::record_length`

Record byte length of a binary SHD file. See Bellhop code for more info

Referenced by woss::ShdResReader::getShdHeader(), and initialize().

13.61.4.10 rx_depths `float* woss::ShdData::rx_depths`

Pointer to an array of receiver depths [m]

Referenced by getPressureIndex(), woss::ShdResReader::getShdHeader(), initialize(), and ~ShdData().

13.61.4.11 rx_ranges `float* woss::ShdData::rx_ranges`

Pointer to an array of receiver ranges [m]

Referenced by getPressureIndex(), woss::ShdResReader::getShdHeader(), initialize(), and ~ShdData().

13.61.4.12 theta `float* woss::ShdData::theta`

Pointer to an array of theta values

Referenced by getPressureIndex(), woss::ShdResReader::getShdHeader(), initialize(), and ~ShdData().

13.61.4.13 tx_depths `float* woss::ShdData::tx_depths`

Pointer to an array of transmitter depths [m]

Referenced by getPressureIndex(), woss::ShdResReader::getShdHeader(), initialize(), and ~ShdData().

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

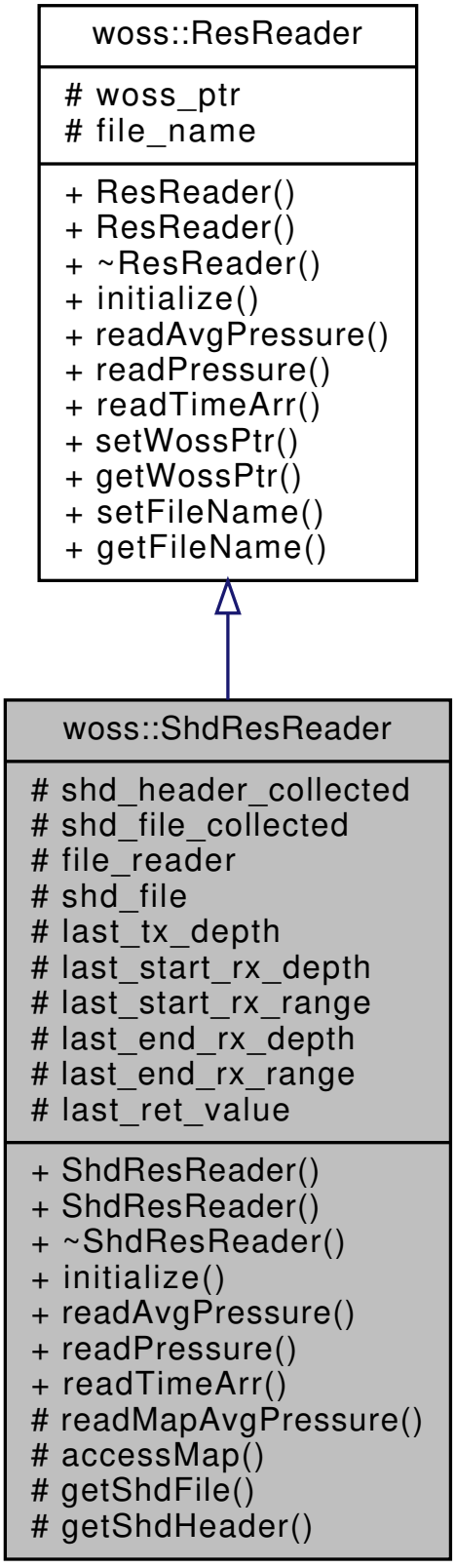
- woss/ac-toolbox-shd-reader.h
- woss/ac-toolbox-shd-reader.cpp

13.62 woss::ShdResReader Class Reference

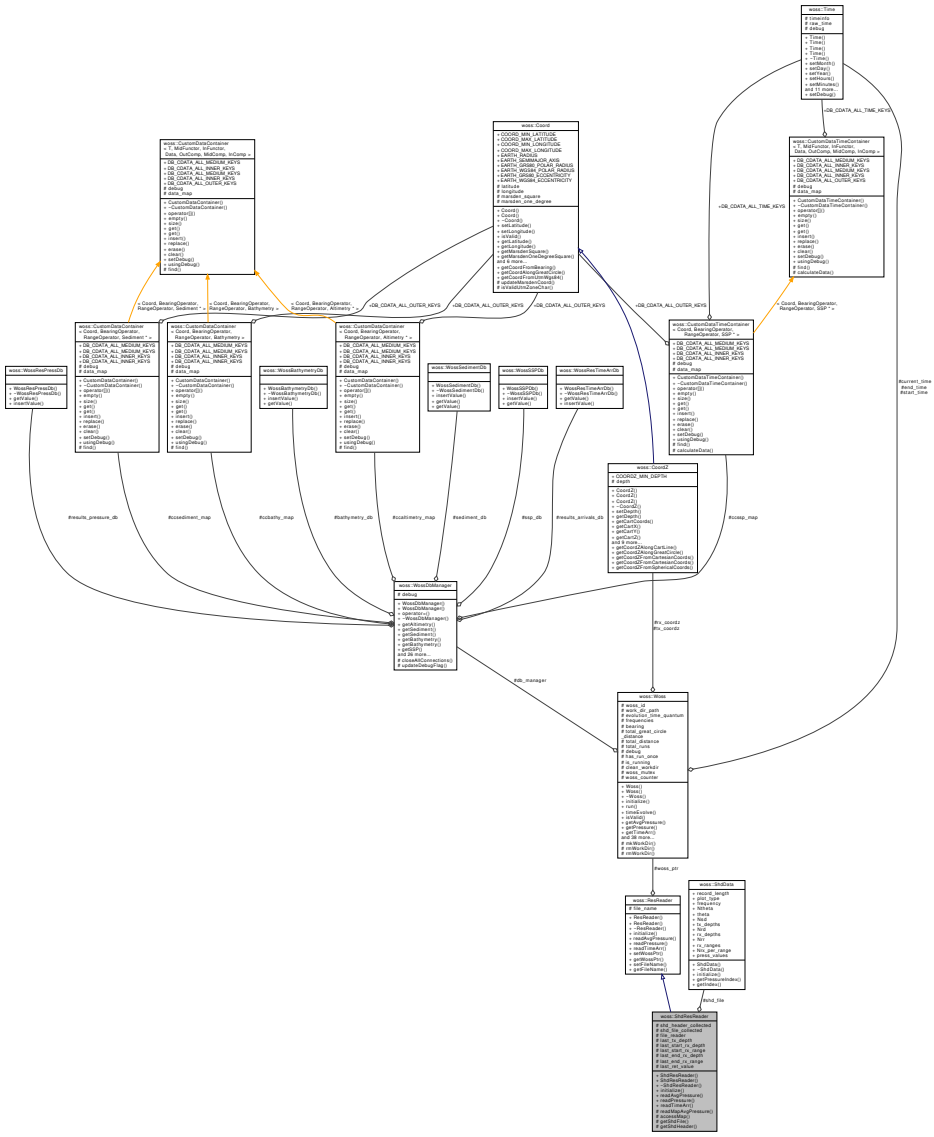
Class for reading and manipulating results provided by any acoustic toolbox SHD file.

`#include <ac-toolbox-shd-reader.h>`

Inheritance diagram for woss::ShdResReader:



Collaboration diagram for woss::ShdResReader:



Public Member Functions

- ShdResReader ()
- ShdResReader (const Woss ∗const woss)
- virtual bool initialize ()
- virtual Pressure ∗ readAvgPressure (double tx_depth, double start_rx_depth, double start_rx_range, double end_rx_depth, double end_rx_range)
- virtual Pressure ∗ readPressure (double tx_depth, double rx_depth, double rx_range) const
- virtual TimeArr ∗ readTimeArr (double tx_depth, double rx_depth, double rx_range) const

Protected Member Functions

- ::std::complex< double > readMapAvgPressure (double tx_depth, double start_rx_depth, double start_rx↩ _range, double end_rx_depth, double end_rx_range, double theta=0.0)
- ::std::complex< double > accessMap (double tx_depth, double rx_depth, double rx_range, double theta=0.0) const
- bool getShdFile ()
- bool getShdHeader ()

Protected Attributes

- bool shd_header_collected
- bool shd_file_collected
- ::std::ifstream file_reader
- ShdData shd_file
- double **last_tx_depth**
- double **last_start_rx_depth**
- double **last_start_rx_range**
- double **last_end_rx_depth**
- double **last_end_rx_range**
- ::std::complex< double > **last_ret_value**

13.62.1 Detailed Description

Class for reading and manipulating results provided by any acoustic toolbox SHD file.

Class ShdResReader stores Pressure provided by any acoustic toolbox SHD file in a ShdData. It also offers Pressure manipulation and TimeArr conversion methods.

13.62.2 Constructor & Destructor Documentation

13.62.2.1 ShdResReader() [1/2] ShdResReader::ShdResReader ( )

ShdResReader default constructor

13.62.2.2 ShdResReader() [2/2] ShdResReader::ShdResReader (const Woss *const *woss*)

ShdResReader constructor

Parameters

woss	const pointer to a const Woss object

13.62.3 Member Function Documentation

13.62.3.1 accessMap() std::complex< double > woss::ShdResReader::accessMap (double *tx_depth*, double *rx_depth*, double *rx_range*, double *theta = 0.0*) const [inline], [protected]

Gets the Pressure value from ShdData Pressure array associated to given parameters

Parameters

tx_depth	transmitter depth [m]
rx_depth	start receiver depth [m]
rx_range	start receiver range [m]
theta	theta value

Returns

a valid Pressure value; a not valid Pressure if shd_file hasn't been read yet

References woss::ShdData::getPressureIndex(), woss::ShdData::press_values, and shd_file.

Referenced by readPressure(), and readTimeArr().

Here is the call graph for this function:

13.62.3.2 getShdFile() `bool ShdResReader::getShdFile ( ) [protected]`

Process the SHD file data

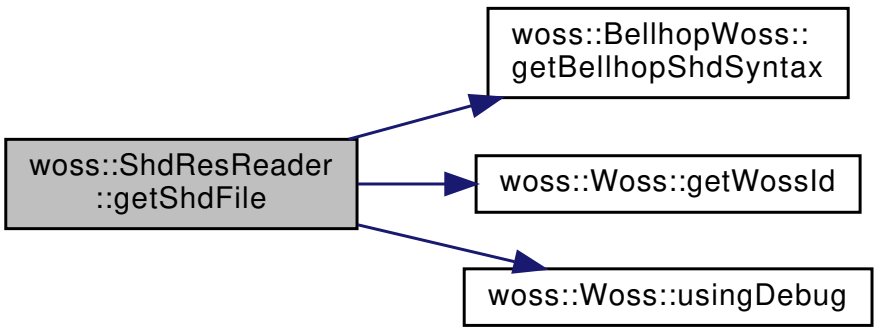
Returns

true if method was successful, *false* otherwise

References woss::ResReader::file_name, file_reader, woss::BellhopWoss::getBellhopShdSyntax(), woss::Woss::getWossId(), woss::ShdData::Nrd, woss::ShdData::Nrr, woss::ShdData::Nrx_per_range, woss::ShdData::Nsd, woss::ShdData::Ntheta, shd_file, woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by initialize().

Here is the call graph for this function:



13.62.3.3 getShdHeader() `bool ShdResReader::getShdHeader ( ) [protected]`

Process the SHD file header

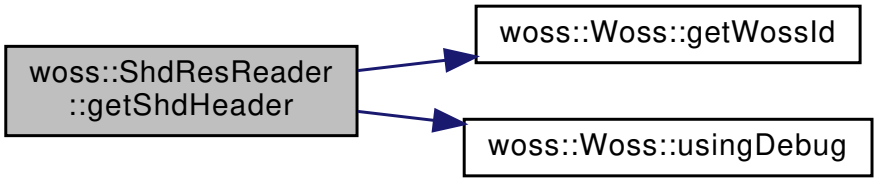
Returns

true if method was successful, *false* otherwise

References woss::ResReader::file_name, file_reader, woss::ShdData::frequency, woss::Woss::getWossId(), woss::ShdData::Nrd, woss::ShdData::Nrr, woss::ShdData::Nrx_per_range, woss::ShdData::Nsd, woss::ShdData::Ntheta, woss::ShdData::plot_type, woss::ShdData::press_values, woss::ShdData::record_length, woss::ShdData::rx_depths, woss::ShdData::rx_ranges, shd_file, shd_header_collected, woss::ShdData::theta, woss::ShdData::tx_depths, woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by initialize().

Here is the call graph for this function:



13.62.3.4 initialize() `bool ShdResReader::initialize ( ) [virtual]`

Initializes the ShdResReader object, reads SHD file, and stores read Pressure values

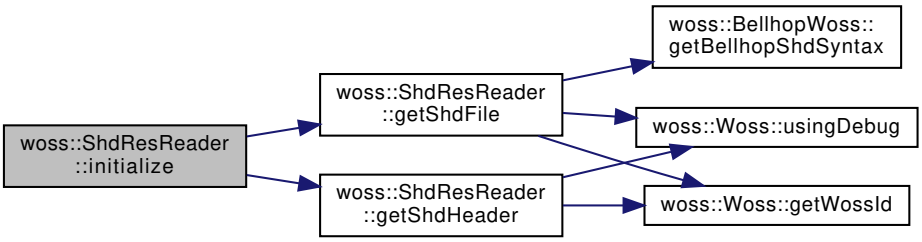
Returns

true if method was successful, *false* otherwise

Implements woss::ResReader.

References woss::ResReader::file_name, getShdFile(), getShdHeader(), and woss::ResReader::woss_ptr.

Here is the call graph for this function:



13.62.3.5 readAvgPressure() `Pressure * ShdResReader::readAvgPressure (`

```
double tx_depth,
double start_rx_depth,
double start_rx_range,
double end_rx_depth,
double end_rx_range ) [virtual]
```

Gets the average Pressure value in given rx range-depth box

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

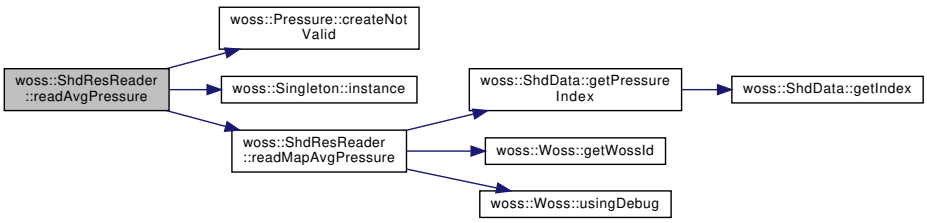
Returns

a valid Pressure value; a not valid Pressure if shd_file hasn't been read yet

Implements woss::ResReader.

References woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), readMapAvgPressure(), and shd_file_collected.

Here is the call graph for this function:



13.62.3.6 readMapAvgPressure()

```
std::complex< double > ShdResReader::readMapAvgPressure (
        double tx_depth,
        double start_rx_depth,
        double start_rx_range,
        double end_rx_depth,
        double end_rx_range,
        double theta = 0.0 )  [protected]
```

Gets the average Pressure value in given rx range-depth box from ShdData Pressure array

Parameters

tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]
theta	theta value

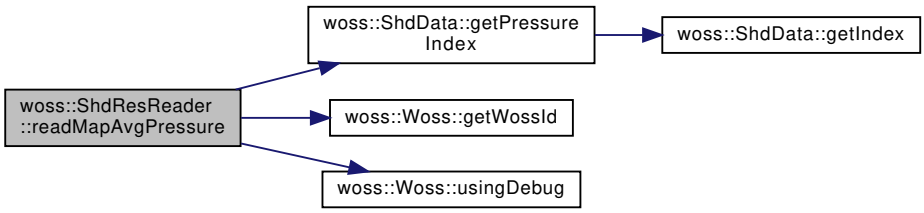
Returns

a valid Pressure value; a not valid Pressure if shd_file hasn't been read yet

References woss::ShdData::getPressureIndex(), woss::Woss::getWossId(), woss::ShdData::press_values, shd_file, woss::Woss::usingDebug(), and woss::ResReader::woss_ptr.

Referenced by readAvgPressure().

Here is the call graph for this function:



13.62.3.7 readPressure()

```
Pressure * ShdResReader::readPressure (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

Gets a Pressure value of given range, depths

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

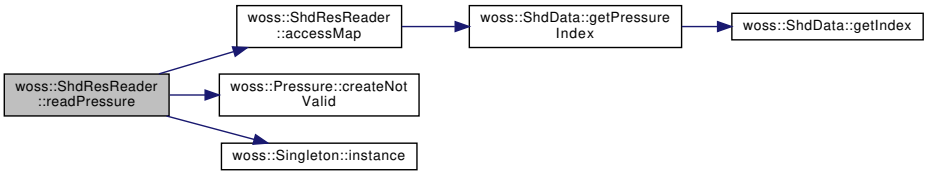
Returns

a valid Pressure value; a not valid Pressure if shd_file hasn't been read yet

Implements woss::ResReader.

References accessMap(), woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), and shd_file_collected.

Here is the call graph for this function:



13.62.3.8 readTimeArr()

```
TimeArr * ShdResReader::readTimeArr (
        double tx_depth,
        double rx_depth,
        double rx_range ) const  [virtual]
```

SHD files don't hold any time arrivals information. A special TimeArr is constructed from Pressure associated to given paramaters.

Parameters

tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

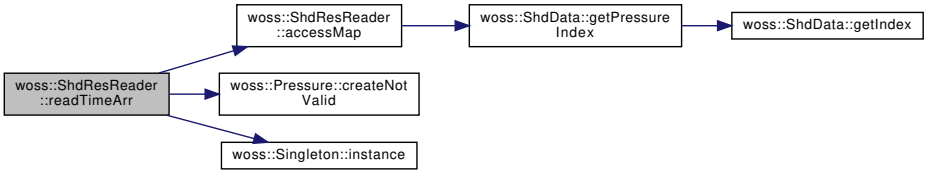
Returns

a special TimeArr that holds a single Pressure value and no delay information

Implements woss::ResReader.

References accessMap(), woss::Pressure::createNotValid(), woss::Singleton< T >::instance(), and shd_file_collected.

Here is the call graph for this function:



13.62.4 Member Data Documentation

13.62.4.1 file_reader

```
::std::ifstream woss::ShdResReader::file_reader  [protected]
```

Input file stream

Referenced by getShdFile(), and getShdHeader().

13.62.4.2 shd_file

```
ShdData woss::ShdResReader::shd_file  [protected]
```

Struct that holds Pressure data read from SHD file

Referenced by accessMap(), getShdFile(), getShdHeader(), and readMapAvgPressure().

13.62.4.3 shd_file_collected `bool woss::ShdResReader::shd_file_collected [protected]`

Boolean associated to the reading of SHD file data

Referenced by readAvgPressure(), readPressure(), and readTimeArr().

13.62.4.4 shd_header_collected `bool woss::ShdResReader::shd_header_collected [protected]`

Boolean associated to the reading of SHD file header

Referenced by getShdHeader().

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

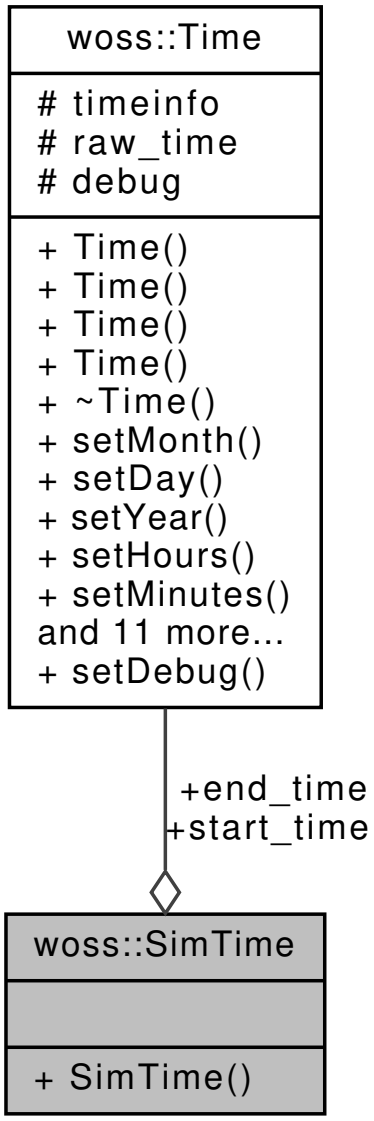
- woss/ac-toolbox-shd-reader.h
- woss/ac-toolbox-shd-reader.cpp

13.63 woss::SimTime Struct Reference

Struct that stores start and end Time.

`#include <time-definitions.h>`

Collaboration diagram for woss::SimTime:



Public Member Functions

- **SimTime** (Time time1=Time(), Time time2=Time())

Public Attributes

- Time **start_time**
- Time **end_time**

Friends

- std::ostream & **operator**<< (std::ostream &os, const SimTime &instance)

13.63.1 Detailed Description

Struct that stores start and end Time.

woss::SimeTime is a pair of woss::Time objects for simulation purposes. A start Time and a end Time

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

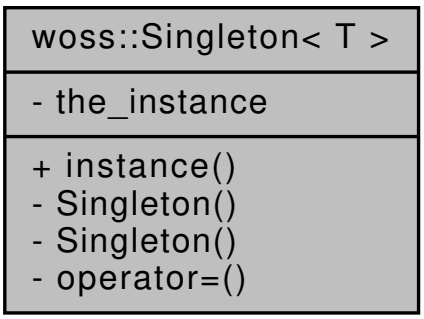
- woss/woss_def/time-definitions.h

13.64 woss::Singleton< T > Class Template Reference

Singleton design pattern template.

`#include <singleton-definitions.h>`

Collaboration diagram for woss::Singleton< T >:



Static Public Member Functions

- static T * instance ()

Private Member Functions

- Singleton ()
- Singleton (const Singleton &copy)
- Singleton & operator= (const Singleton &copy)

Static Private Attributes

- static T * the_instance = 0

13.64.1 Detailed Description

template<typename T>
class woss::Singleton< T >

Singleton design pattern template.

A simple singleton template pattern, useful to decouple classes from singleton implementation

13.64.2 Constructor & Destructor Documentation

13.64.2.1 Singleton() [1/2] template<typename T >

```
woss::Singleton< T >::Singleton ( )  [inline], [private]
```

Disallowed default constructor

13.64.2.2 Singleton() [2/2] template<typename T >

```
woss::Singleton< T >::Singleton (
            const Singleton< T > & copy )  [inline], [private]
```

Disallowed copy constructor

13.64.3 Member Function Documentation

13.64.3.1 instance() template<typename T >

```
T * woss::Singleton< T >::instance  [inline], [static]
```

Returns the singleton instance

Returns

a pointer to *the_instance*

Referenced by woss::WossManagerResDb::dbGetPressure(), woss::WossManagerResDb::dbGetTimeArr(), woss::SSP::fullRandomize(), woss::WossDbManager::getAltimetry(), woss::ArrBinResReader::getArrBinHeader(), woss::WossDbManager::getAverageSSP(), woss::WossDbManager::getPressure(), woss::WossDbManager::getSediment(), woss::WossDbManager::getSSP(), woss::WossDbManager::getTimeArr(), WossMPhyBpsk::getTxPower(), woss::ResPressureTxtDb::getValue(), woss::ResTimeArrTxtDb::getValue(), woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWos woss::WossManagerResDbMT::getWossTimeArr(), woss::WossManager::getWossTimeArr(), woss::WossDbManager::importCustom woss::TransducerHandler::importValueAscii(), woss::TransducerHandler::importValueBinary(), woss::TransducerHandler::operator=(), woss::Altimetry::randomize(), woss::SSP::randomize(), woss::ArrAscResReader::readAvgPressure(), woss::ArrBinResReader::readA woss::ShdResReader::readAvgPressure(), woss::ArrAscResReader::readPressure(), woss::ArrBinResReader::readPressure(), woss::ShdResReader::readPressure(), woss::ArrAscResReader::readTimeArr(), woss::ArrBinResReader::readTimeArr(), woss::ShdResReader::readTimeArr(), and woss::TransducerHandler::TransducerHandler().

13.64.3.2 operator=() template<typename T >

```
Singleton & woss::Singleton< T >::operator= (
            const Singleton< T > & copy )  [inline], [private]
```

Disallowed assignment operator

13.64.4 Member Data Documentation

13.64.4.1 the_instance `template<typename T >`

`T * woss::Singleton< T >::the_instance = 0 [static], [private]`

static heap instance

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

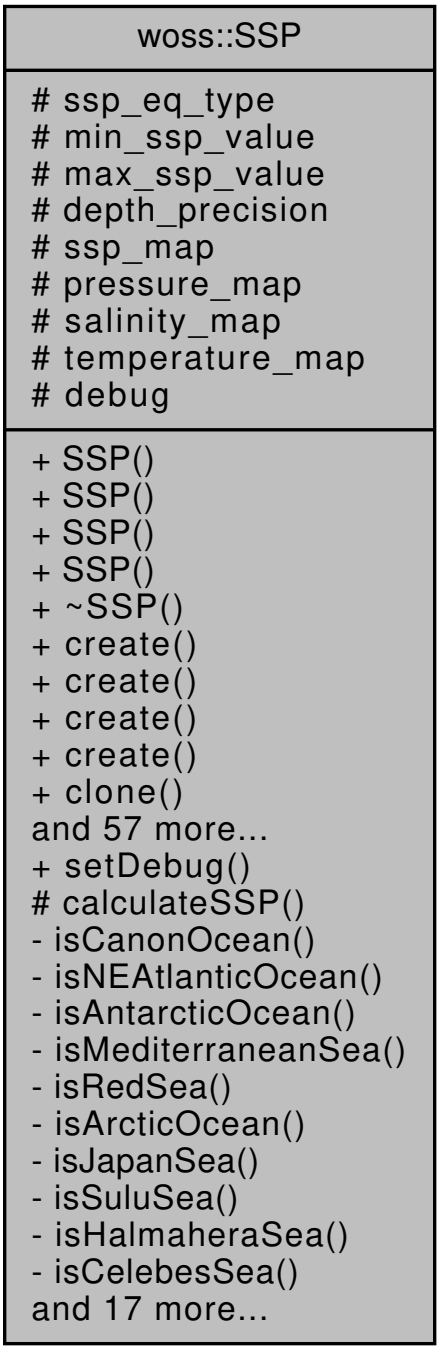
- woss/woss_def/singleton-definitions.h

13.65 woss::SSP Class Reference

SSP class offers multiple creation and manipulation capabilities for sound speed profile.

`#include <ssp-definitions.h>`

Collaboration diagram for woss::SSP:



Public Types

- enum SSPEqType { SSP_EQ_CHEN_MILLERO = 0 , SSP_EQ_TEOS_10 = 1 , SSP_EQ_TEOS_10_EXACT = 2 , SSP_EQ_INVALID }

Public Member Functions

- SSP (long double depth_precision=SSP_CUSTOM_DEPTH_PRECISION)

- SSP (DepthMap &ssp_map, DepthMap &temp_map, DepthMap &sal_map, DepthMap &press_map, long double depth_precision=SSP_CUSTOM_DEPTH_PRECISION)
- SSP (DepthMap &ssp_map, long double depth_precision=SSP_CUSTOM_DEPTH_PRECISION)
- SSP (const SSP &copy)
- virtual SSP * create (long double depth_precision=SSP_CUSTOM_DEPTH_PRECISION) const
- virtual SSP * create (DepthMap &ssp_map, DepthMap &temp_map, DepthMap &sal_map, DepthMap &press_map, long double depth_precision=SSP_CUSTOM_DEPTH_PRECISION) const
- virtual SSP * create (DepthMap &ssp_map, long double depth_precision=SSP_CUSTOM_DEPTH_↩ PRECISION) const
- virtual SSP * create (const SSP &copy) const
- virtual SSP * clone () const
- int size () const
- bool empty () const
- void clear ()
- DConstIter begin () const
- DConstIter end () const
- DConstRIter rbegin () const
- DConstRIter rend () const
- DConstIter lower_bound (const PDouble &depth) const
- DConstIter upper_bound (const PDouble &depth) const
- DConstIter at (const int i) const
- DConstIter pressure_begin () const
- DConstIter pressure_end () const
- DConstRIter pressure_rbegin () const
- DConstRIter pressure_rend () const
- DConstIter pressure_lower_bound (const PDouble &depth) const
- DConstIter pressure_upper_bound (const PDouble &depth) const
- DConstIter pressure_find (const PDouble &depth) const
- DConstIter temperature_begin () const
- DConstIter temperature_end () const
- DConstRIter temperature_rbegin () const
- DConstRIter temperature_rend () const
- DConstIter temperature_lower_bound (const PDouble &depth) const
- DConstIter temperature_upper_bound (const PDouble &depth) const
- DConstIter temperature_find (const PDouble &depth) const
- DConstIter salinity_begin () const
- DConstIter salinity_end () const
- DConstRIter salinity_rbegin () const
- DConstRIter salinity_rend () const
- DConstIter salinity_lower_bound (const PDouble &depth) const
- DConstIter salinity_upper_bound (const PDouble &depth) const
- DConstIter salinity_find (const PDouble &depth) const
- virtual bool isValid () const
- virtual bool isTransformable () const
- virtual bool isRandomizable () const
- virtual SSP * transform (const Coord &coordinates, double new_min_depth=-HUGE_VAL, double new_↩ max_depth=HUGE_VAL, int total_depth_steps=SSP_CUSTOM_DEPTH_STEPS) const
- virtual SSP * truncate (double max_depth) const
- virtual SSP * fullRandomize (double ratio_incr_value) const
- virtual SSP * randomize (double ratio_incr_value) const
- virtual bool import (::std::istream &stream_in)
- virtual bool write (::std::ostream &stream_out) const
- SSP & insertValue (double depth, double ssp_value)
- SSP & insertValue (double depth, double temperature, double salinity, const ::std::complex< double > &pressure, double ssp_value)

- SSP & insertValue (double temperature, double salinity, const ::std::complex< double > &pressure, const Coord &coordinates=Coord(0.0, 0.0))
- SSP & insertValue (double depth, double temperature, double salinity, const Coord &coordinates=Coord(0.0, 0.0))
- DConstIter findValue (const double &depth) const
- SSP & eraseValue (const double &depth)
- double getMaxDepthValue () const
- double getMinDepthValue () const
- double getMaxSSPValue () const
- double getMinSSPValue () const
- virtual void setDepthPrecision (long double prec)
- long double getDepthPrecision () const
- SSP & setSSPEqType (SSPEqType eq_type)
- SSPEqType getSSPEqType () const
- SSP & operator= (const SSP &x)
- friend::std::ostream & operator<< (::std::ostream &os, const SSP &instance)
- friend::std::ostream & operator>> (::std::istream &is, const SSP &instance)

Static Public Member Functions

- static void setDebug (bool flag)

Protected Member Functions

- virtual double calculateSSP (double temperature, double salinity, double pressure) const

Protected Attributes

- SSPEqType ssp_eq_type
- double min_ssp_value
- double max_ssp_value
- long double depth_precision
- DepthMap ssp_map
- DepthMap pressure_map
- DepthMap salinity_map
- DepthMap temperature_map

Static Protected Attributes

- static bool debug = false

Private Member Functions

- bool isCanonOcean (const Coord &coordinates) const
- bool isNEAtlanticOcean (const Coord &coordinates) const
- bool isAntarcticOcean (const Coord &coordinates) const
- bool isMediterraneanSea (const Coord &coordinates) const
- bool isRedSea (const Coord &coordinates) const
- bool isArcticOcean (const Coord &coordinates) const
- bool isJapanSea (const Coord &coordinates) const
- bool isSuluSea (const Coord &coordinates) const
- bool isHalmaheraSea (const Coord &coordinates) const
- bool isCelebesSea (const Coord &coordinates) const
- bool isBlackSea (const Coord &coordinates) const
- bool isBalticSea (const Coord &coordinates) const
- double thyh (double z) const
- double g (double lat) const
- double k (double z, double lat) const
- double hq (double z) const
- double h (double z, double lat) const
- double getPressureCorreptions (const Coord &coordinates, double depth) const
- double getPressureFromDepth (const Coord &coordinates, double depth) const
- double g_z (double lat) const
- double getDepthCorreptions (const Coord &coordinates, double pressure) const
- double getDepthfromPressure (const Coord &coordinates, double pressure) const
- double d (double t, double p) const
- double b (double t, double p) const
- double a (double t, double p) const
- double cw (double t, double p) const
- double gibbs (int ns, int nt, int np, double sa, double t, double p) const

Friends

- bool operator== (const SSP &left, const SSP &right)
- bool operator!= (const SSP &left, const SSP &right)
- const SSP operator+ (const SSP &left, const SSP &right)
- const SSP operator- (const SSP &left, const SSP &right)
- const SSP operator∗ (const SSP &left, const SSP &right)
- const SSP operator/ (const SSP &left, const SSP &right)
- const SSP operator+ (const SSP &left, const double right)
- const SSP operator- (const SSP &left, const double right)
- const SSP operator/ (const SSP &left, const double right)
- const SSP operator∗ (const SSP &left, const double right)
- const SSP operator+ (const double left, const SSP &right)
- const SSP operator- (const double left, const SSP &right)
- const SSP operator/ (const double left, const SSP &right)
- const SSP operator∗ (const double left, const SSP &right)
- SSP & operator+= (SSP &left, const SSP &right)
- SSP & operator-= (SSP &left, const SSP &right)
- SSP & operator∗= (SSP &left, const SSP &right)
- SSP & operator/= (SSP &left, const SSP &right)
- SSP & operator+= (SSP &left, const double right)
- SSP & operator-= (SSP &left, const double right)
- SSP & operator/= (SSP &left, const double right)
- SSP & operator∗= (SSP &left, const double right)

13.65.1 Detailed Description

SSP class offers multiple creation and manipulation capabilities for sound speed profile.

SSP can store all information related to a sound speed profile: temperature [C°], pressure [bar], salinity [psu] and sound speed [m/s]. It offers capabilities for arithmetic computations, sound speed calculations (Chen and Millero equations), depth to pressure conversions (and viceversa) with coordinates corrections, sound speed profile transformations and random perturbation.

13.65.2 Member Enumeration Documentation

13.65.2.1 SSPEqType `enum woss::SSP::SSPEqType`

SSP EQ

Enumerator

SSP_EQ_CHEN_MILLERO	UNESCO Chen & Miller, Wong Zu Eq.
SSP_EQ_TEOS_10	TEOS-10 Eq (. **See also** reference [10])
SSP_EQ_TEOS_10_EXACT	TEOS-10 Exact Eq (. **See also** reference [10])
SSP_EQ_INVALID	Must always be the last.

13.65.3 Constructor & Destructor Documentation

13.65.3.1 SSP() [1/4] `SSP::SSP (long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION )`

SSP default constructor. The object created is not valid

Parameters

depth_precision	precision of PDouble objects representing depth.

Referenced by clone(), create(), fullRandomize(), randomize(), transform(), and truncate().

13.65.3.2 SSP() [2/4]

```
SSP::SSP (
    DepthMap & ssp_map,
    DepthMap & temp_map,
    DepthMap & sal_map,
    DepthMap & press_map,
    long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION )
```

SSP constructor.

Parameters

ssp_map	map linking a PDouble depth [m] (with precision depth_precision) to a sound speed [m/s]
temp_map	map linking a PDouble depth [m] (with precision depth_precision) to a temperature [C°]
sal_map	map linking a PDouble depth [m] (with precision depth_precision) to a salinity [ppu]
press_map	map linking a PDouble depth [m] (with precision depth_precision) to a pressure [bar]
depth_precision	depth precision of given maps [m]

References pressure_map, salinity_map, ssp_map, and temperature_map.

13.65.3.3 SSP() [3/4]

```
SSP::SSP (
    DepthMap & ssp_map,
    long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION )
```

SSP constructor.

Parameters

ssp_map	map linking a PDouble depth [m] (with precision depth_precision) to a sound speed [m/s]
depth_precision	depth precision of given maps [m]

References ssp_map.

13.65.3.4 SSP() [4/4]

```
SSP::SSP (
    const SSP & copy )
```

SSP copy constructor

Parameters

copy	SSP to be copied

References depth_precision, max_ssp_value, min_ssp_value, pressure_map, salinity_map, ssp_eq_type, ssp_map, and temperature_map.

13.65.4 Member Function Documentation

13.65.4.1 a()

```
double woss::SSP::a (
    double t,
    double p ) const  [inline], [private]
```

UNESCO Chen and Millero Equation with Wong and Zu corrections for sound speed calculations

Parameters

t	temperature provided [C°]
p	pressure provided [bar]

Referenced by calculateSSP().

13.65.4.2 at()

```
DConstIter SSP::at (
    const int i ) const
```

Returns a const iterator to the sound speed value at i-th position

Parameters

i	integer should be between 0 and size()

Returns

const iterator to end() if position *i* is not found

References ssp_map.

13.65.4.3 b()

```
double woss::SSP::b (
    double t,
    double p ) const  [inline], [private]
```

UNESCO Chen and Millero Equation with Wong and Zu corrections for sound speed calculations

Parameters

t	temperature provided [C°]
p	pressure provided [bar]

Referenced by calculateSSP().

13.65.4.4 begin()

```
DConstIter woss::SSP::begin ( ) const  [inline]
```

Returns a const iterator to the beginning of the sound speed map

Returns

const iterator

References ssp_map.

Referenced by truncate(), and woss::BellhopWoss::writeNormalizedSSP().

13.65.4.5 calculateSSP()

```
double woss::SSP::calculateSSP (
        double temperature,
        double salinity,
        double pressure ) const  [inline], [protected], [virtual]
```

Calculates sound speed from temperature, salinity, pressure with UNESCO Chen and Millero equations and with Wong and Zu corrections

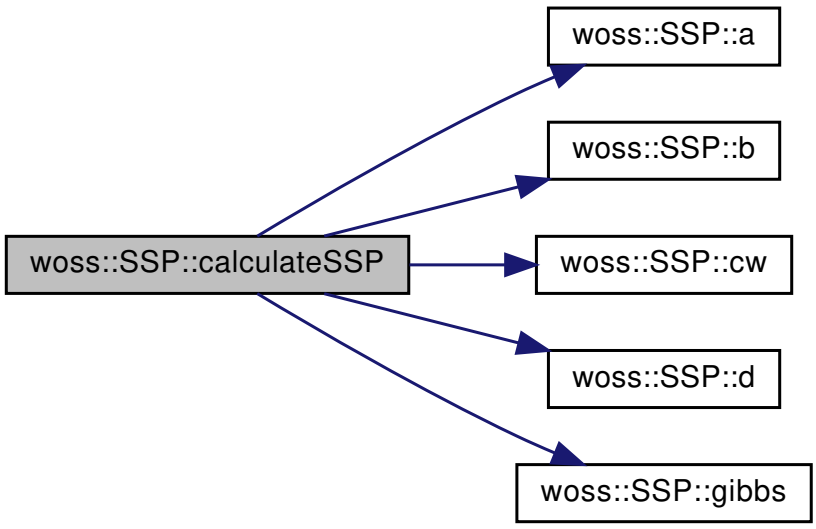
Parameters

temperature	temperature value [C°]
salinity	salinity value [ppu]
pressure	pressure value [bar]

References a(), b(), cw(), d(), gibbs(), SSP_EQ_CHEN_MILLERO, SSP_EQ_TEOS_10, SSP_EQ_TEOS_10_EXACT, and ssp_eq_type.

Referenced by fullRandomize(), insertValue(), and transform().

Here is the call graph for this function:



13.65.4.6 clear() `void woss::SSP::clear ( ) [inline]`

Erase all values of sound speed, temperature, pressure and salinity. The object therefore is not valid

References pressure_map, salinity_map, ssp_map, and temperature_map.

13.65.4.7 clone() `virtual SSP * woss::SSP::clone ( ) const [inline], [virtual]`

SSP virtual factory method

Returns

a heap-created copy of **this** instance

References SSP().

Referenced by woss::DefHandler::operator=(), and truncate().

Here is the call graph for this function:

13.65.4.8 create() [1/4] `virtual SSP * woss::SSP::create ( const SSP & copy ) const [inline], [virtual]`

SSP virtual factory method

Parameters

copy	SSP to be copied

Returns

a heap-created SSP object

References SSP().

Here is the call graph for this function:

13.65.4.9 create() [2/4]

```
virtual SSP * woss::SSP::create (
        DepthMap & ssp_map,
        DepthMap & temp_map,
        DepthMap & sal_map,
        DepthMap & press_map,
        long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const  [inline], [virtual]
```

SSP virtual factory method

Parameters

ssp_map	map linking a PDouble depth [m] (with precision depth_precision) to a sound speed [m/s]
temp_map	map linking a PDouble depth [m] (with precision depth_precision) to a temperature [C°]
sal_map	map linking a PDouble depth [m] (with precision depth_precision) to a salinity [ppu]
press_map	map linking a PDouble depth [m] (with precision depth_precision) to a pressure [bar]
depth_precision	depth precision of given maps [m]

Returns

a heap-created SSP object

References depth_precision, SSP(), and ssp_map.

Here is the call graph for this function:

13.65.4.10 create() [3/4]

```
virtual SSP * woss::SSP::create (
        DepthMap & ssp_map,
        long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const  [inline], [virtual]
```

SSP virtual factory method

Parameters

ssp_map	map linking a PDouble depth [m] (with precision depth_precision) to a sound speed [m/s]
depth_precision	depth precision of given maps [m]

Returns

a heap-created SSP object

References depth_precision, SSP(), and ssp_map.

Here is the call graph for this function:

13.65.4.11 create() [4/4]

```
virtual SSP * woss::SSP::create (
    long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const  [inline], [virtual]
```

SSP virtual factory method

Parameters

depth_precision	precision of PDouble objects representing depth.

Returns

a heap-created SSP object

References depth_precision, and SSP().

Here is the call graph for this function:

13.65.4.12 cw()

```
double woss::SSP::cw (
    double t,
    double p ) const  [inline], [private]
```

UNESCO Chen and Millero Equation with Wong and Zu corrections for sound speed calculations

Parameters

t	temperature provided [C°]
p	pressure provided [bar]

Referenced by calculateSSP().

13.65.4.13 d()

```
double woss::SSP::d (
    double t,
    double p ) const  [inline], [private]
```

UNESCO Chen and Millero Equation with Wong and Zu corrections for sound speed calculations

Parameters

t	temperature provided [C°]
p	pressure provided [bar]

Referenced by calculateSSP().

13.65.4.14 empty() `bool woss::SSP::empty ( ) const [inline]`

Checks if the instance has stored values

Returns

true if condition applies, *false* otherwise

References ssp_map.

13.65.4.15 end() `DConstIter woss::SSP::end ( ) const [inline]`

Returns a const iterator to the end of the sound speed map

Returns

const iterator

References ssp_map.

Referenced by truncate(), and woss::BellhopWoss::writeNormalizedSSP().

13.65.4.16 eraseValue() `SSP & woss::SSP::eraseValue ( const double & depth ) [inline]`

Erase the sound speed value with key == of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

reference to ∗**this**

References ssp_map.

13.65.4.17 findValue() DConstIter woss::SSP::findValue (const double & *depth*) const [inline]

Returns a const iterator to the sound speed value with key == of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to end() if *depth* is not found

References ssp_map.

13.65.4.18 fullRandomize() SSP * SSP::fullRandomize (double *ratio_incr_value*) const [virtual]

Performs a random perturbation of temperature, salinity, pressure with given ratio Sound speed values are then calculated from this new data

Parameters

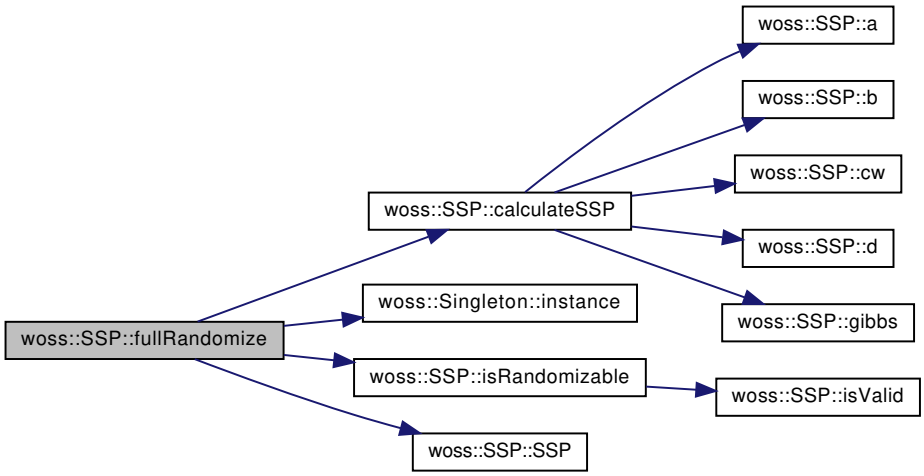
ratio_incr_value	perturbation ratio

Returns

a *new* SSP object

References calculateSSP(), depth_precision, woss::Singleton< T >::instance(), isRandomizable(), pressure_map, salinity_map, SSP(), ssp_map, and temperature_map.

Here is the call graph for this function:



13.65.4.19 g() `double woss::SSP::g ( double lat ) const [inline], [private]`

Equation for pressure from depth conversion (Leroy and Parthiot)

Parameters

lat	latitude [decimal degree]

Referenced by k().

13.65.4.20 g_z() `double woss::SSP::g_z ( double lat ) const [inline], [private]`

Equation for depth from pressure conversion (Leroy and Parthiot)

Parameters

lat	latitude [decimal degree]

Referenced by getDepthfromPressure().

13.65.4.21 getDepthCorreptions() `double SSP::getDepthCorreptions ( const Coord & coordinates, double pressure ) const [private]`

Returns depth correction for given coordinates and pressure (Leroy and Parthiot)

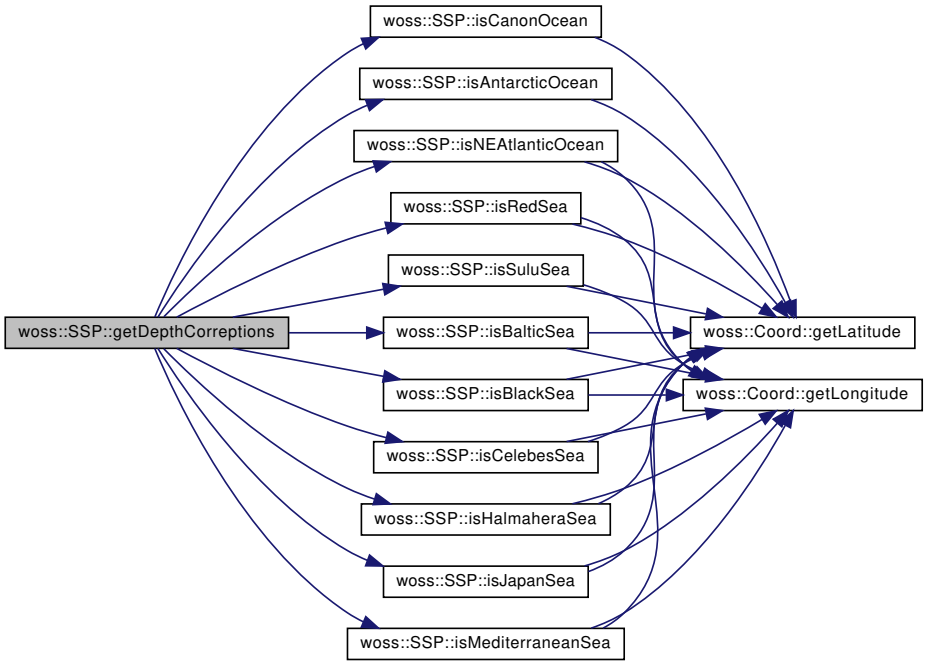
Parameters

coordinates	coordinates provided
pressure	pressure provided [bar]

References isAntarcticOcean(), isBalticSea(), isBlackSea(), isCanonOcean(), isCelebesSea(), isHalmaheraSea(), isJapanSea(), isMediterraneanSea(), isNEAtlanticOcean(), isRedSea(), and isSuluSea().

Referenced by getDepthfromPressure().

Here is the call graph for this function:



13.65.4.22 getDepthfromPressure()

```
double woss::SSP::getDepthfromPressure (
    const Coord & coordinates,
    double pressure ) const  [inline], [private]
```

Returns depth for given coordinates and pressure (Leroy and Parthiot)

Parameters

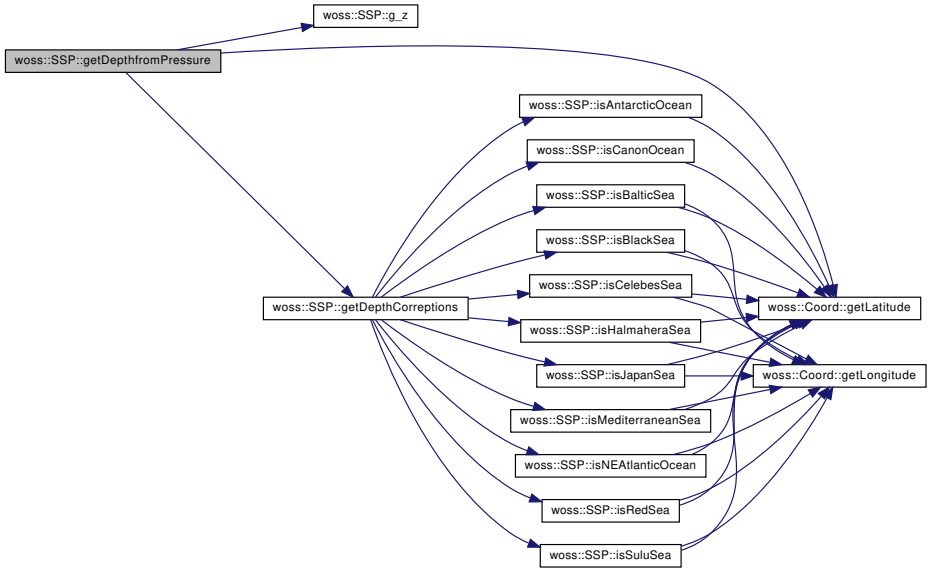
coordinates	coordinates provided
pressure	pressure provided [bar]

Returns

depth [m]

References g_z(), getDepthCorreptions(), and woss::Coord::getLatitude().

Here is the call graph for this function:



13.65.4.23 getDepthPrecision() `long double woss::SSP::getDepthPrecision ( ) const [inline]`

Returns the sound speed profile depth precision

Returns

depth precision [m]

References depth_precision.

13.65.4.24 getMaxDepthValue() `double woss::SSP::getMaxDepthValue ( ) const [inline]`

Returns the maximum depth value

Returns

maximum depth [m]

References ssp_map.

Referenced by woss::ACToolboxWoss::initSSPVector().

13.65.4.25 getMaxSSPValue() `double woss::SSP::getMaxSSPValue ( ) const [inline]`

Returns the maximum sound speed value

Returns

maximum sound speed [m/s]

References max_ssp_value.

13.65.4.26 getMinDepthValue() `double woss::SSP::getMinDepthValue ( ) const [inline]`

Returns the minimum depth value

Returns

minimum depth [m]

References ssp_map.

Referenced by woss::ACToolboxWoss::initSSPVector().

13.65.4.27 getMinSSPValue() `double woss::SSP::getMinSSPValue ( ) const [inline]`

Returns the maximum depth value

Returns

maximum sound speed [m/s]

References min_ssp_value.

13.65.4.28 getPressureCorreptions() `double SSP::getPressureCorreptions ( const Coord & coordinates, double depth ) const [private]`

Returns pressure correction for given coordinates and depth (Leroy and Parthiot)

Parameters

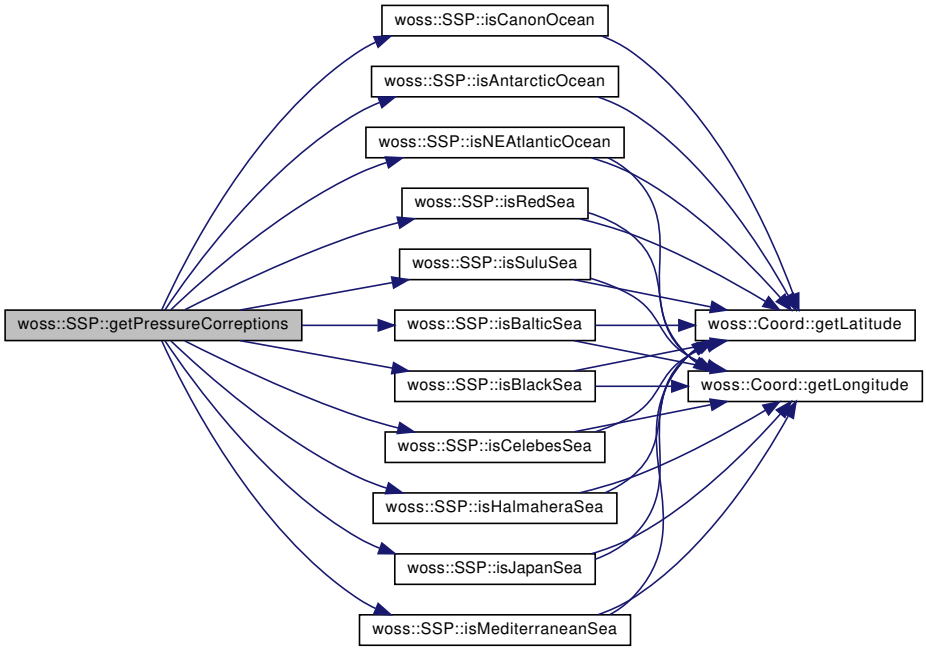
coordinates	coordinates provided
depth	depth provided [m]

References isAntarcticOcean(), isBalticSea(), isBlackSea(), isCanonOcean(), isCelebesSea(), isHalmaheraSea(),

isJapanSea(), isMediterraneanSea(), isNEAtlanticOcean(), isRedSea(), and isSuluSea().

Referenced by getPressureFromDepth().

Here is the call graph for this function:



13.65.4.29 getPressureFromDepth()

```
double woss::SSP::getPressureFromDepth (
    const Coord & coordinates,
    double depth ) const  [inline], [private]
```

Returns pressure for given coordinates and depth (Leroy and Parthiot)

Parameters

coordinates	coordinates provided
depth	depth provided [m]

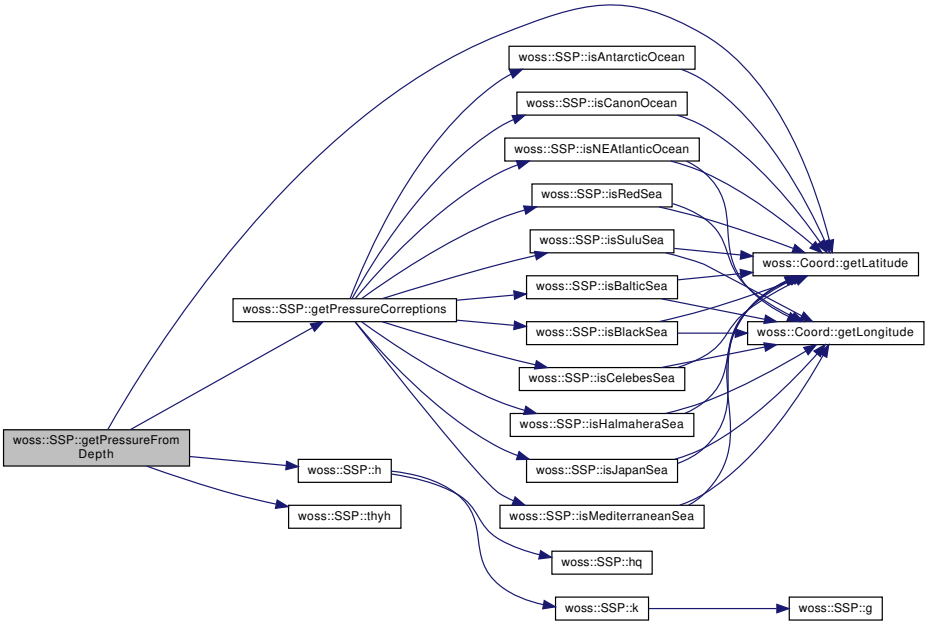
Returns

pressure [bar]

References woss::Coord::getLatitude(), getPressureCorreptions(), h(), and thyh().

Referenced by insertValue(), and transform().

Here is the call graph for this function:



13.65.4.30 getSSPEqType() `SSPEqType woss::SSP::getSSPEqType ( ) const [inline]`

Gets the current SSPEqType

Returns

current SSPEqType

References ssp_eq_type.

13.65.4.31 gibbs() `double woss::SSP::gibbs (`

```
int ns,
int nt,
int np,
double sa,
double t,
double p ) const [inline], [private]
```

Seawater specific Gibbs free energy and derivatives up to order 2

Parameters

ns	order of salinity derivative
nt	order of temperature derivative
np	order of pressure derivative
sa	salinity [ppu]
t	temperature [C°]
p	pressure [dbar]

Referenced by calculateSSP().

13.65.4.32 h() double woss::SSP::h (double z, double lat) const [inline], [private]

Equation for pressure from depth conversion (Leroy and Parthiot)

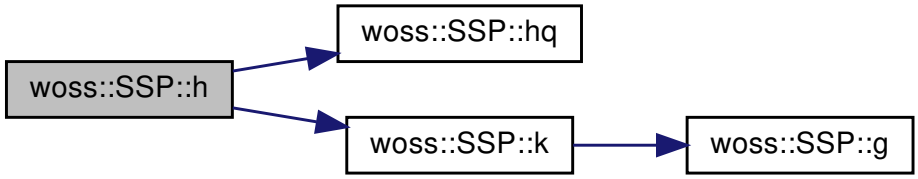
Parameters

z	depth [m]
lat	latitude [decimal degree]

References hq(), and k().

Referenced by getPressureFromDepth().

Here is the call graph for this function:



13.65.4.33 hq() double woss::SSP::hq (double z) const [inline], [private]

Equation for pressure from depth conversion (Leroy and Parthiot)

Parameters

z	depth [m]

Referenced by h().

13.65.4.34 import() bool SSP::import (::std::istream & stream_in) [virtual]

Imports values in from the given stream

Parameters

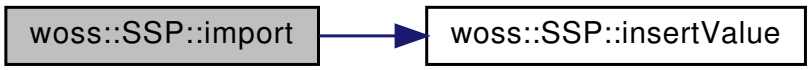
stream_in	const reference to an istream instance

Returns

true if method was successful, false otherwise

References insertValue().

Here is the call graph for this function:



13.65.4.35 insertValue() [1/4]

```
SSP & SSP::insertValue (
    double depth,
    double ssp_value )
```

Inserts and doesn't replace a sound speed value at given depth

Parameters

depth	depth value [m]. The corresponding PDouble will take SSP::depth_precision as precision
ssp_value	sound speed value [m/s]

Returns

reference to ∗**this**

References depth_precision, max_ssp_value, min_ssp_value, and ssp_map.

Referenced by import(), woss::WossDbManager::importCustomSSP(), and truncate().

13.65.4.36 insertValue() [2/4]

```
SSP & woss::SSP::insertValue (
    double depth,
    double temperature,
    double salinity,
    const ::std::complex< double > & pressure,
    double ssp_value )
```

Inserts and doesn't replace a sound speed value at given depth

Parameters

depth	depth value [m]. The corresponding PDouble will take SSP::depth_precision as precision
temperature	temperature value [C°]
salinity	salinity value [ppu]
pressure	pressure value [bar]
ssp_value	sound speed value [m/s]

Returns

reference to ∗**this**

13.65.4.37 insertValue() [3/4]

```
SSP & SSP::insertValue (
        double depth,
        double temperature,
        double salinity,
        const Coord & coordinates = Coord(0.0, 0.0) )
```

Calculates and doesn't replace sound speed from given temperature, pressure and salinity value at given depth, with pressure converted from depth.

Parameters

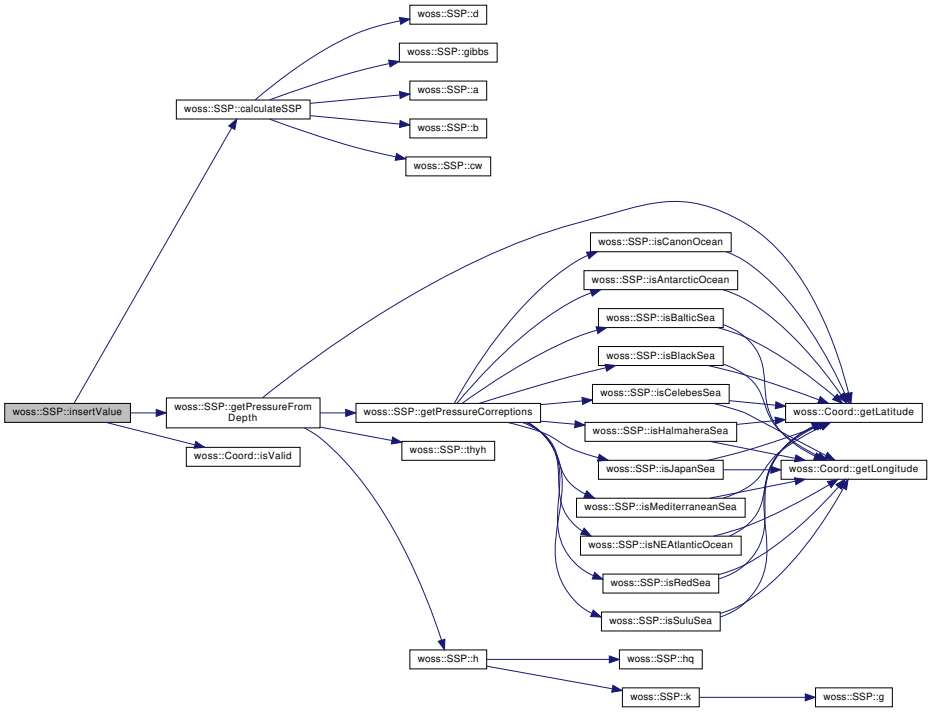
coordinates	coordinates for depth-to-pressure conversion corrections (defaults to CanonOcean)
depth	depth value [m]
temperature	temperature value [C°]
salinity	salinity value [ppu]

Returns

reference to ∗**this**

References calculateSSP(), depth_precision, getPressureFromDepth(), woss::Coord::isValid(), max_ssp_value, min_ssp_value, pressure_map, salinity_map, ssp_map, and temperature_map.

Here is the call graph for this function:



13.65.4.38 insertValue() [4/4]

```
SSP & woss::SSP::insertValue (
    double temperature,
    double salinity,
    const ::std::complex< double > & pressure,
    const Coord & coordinates = Coord(0.0, 0.0) )
```

Calculates and doesn't replace sound speed from given temperature, pressure and salinity value at depth converted from pressure.

Parameters

coordinates	coordinates for pressure-to-depth conversion corrections (defaults to CanonOcean)
temperature	temperature value [C°]
salinity	salinity value [ppu]
pressure	pressure value [bar]

Returns

reference to ∗**this**

13.65.4.39 isAntarcticOcean()

```
bool woss::SSP::isAntarcticOcean (
    const Coord & coordinates ) const  [inline], [private]
```

Checks if coordinates provided are in Antarctic Ocean

Parameters

coordinates	coordinates value

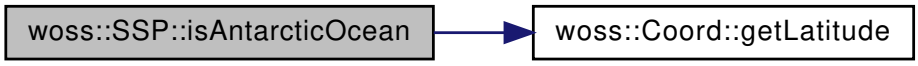
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.40 isArcticOcean()

```
bool woss::SSP::isArcticOcean (
    const Coord & coordinates ) const  [inline], [private]
```

Checks if coordinates provided are in Arctic Ocean

Parameters

coordinates	coordinates value

Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude().

Here is the call graph for this function:

13.65.4.41 isBalticSea()

```
bool woss::SSP::isBalticSea (
    const Coord & coordinates ) const  [inline], [private]
```

Checks if coordinates provided are in Baltic Sea

Parameters

coordinates	coordinates value

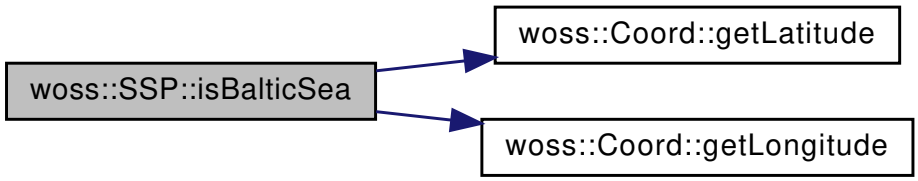
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.42 isBlackSea() `bool woss::SSP::isBlackSea ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in Black Sea

Parameters

coordinates	coordinates value

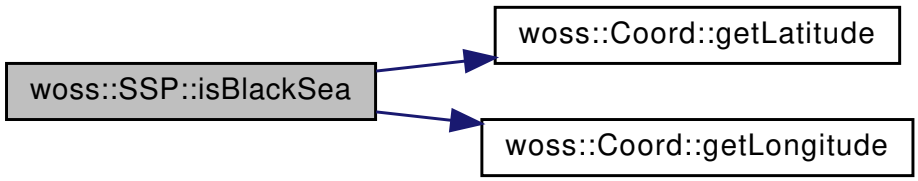
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.43 isCanonOcean() `bool woss::SSP::isCanonOcean ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in canon ocean

Parameters

coordinates	coordinates value

Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:

13.65.4.44 isCelebesSea() `bool woss::SSP::isCelebesSea ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in Celebes Sea

Parameters

coordinates	coordinates value

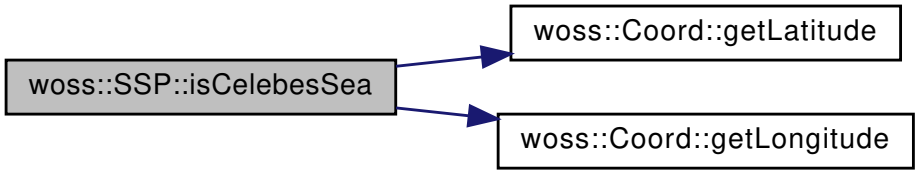
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.45 isHalmaheraSea() bool woss::SSP::isHalmaheraSea (const Coord & *coordinates*) const [inline], [private]

Checks if coordinates provided are in Halmahera Sea

Parameters

coordinates	coordinates value

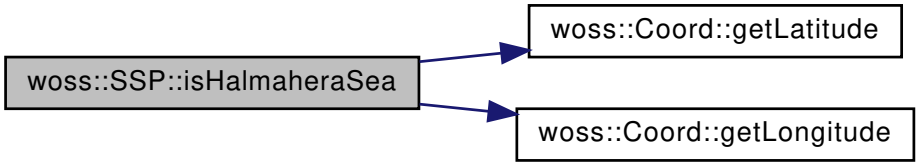
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.46 isJapanSea() bool woss::SSP::isJapanSea (const Coord & *coordinates*) const [inline], [private]

Checks if coordinates provided are in Japan Sea

Parameters

coordinates	coordinates value

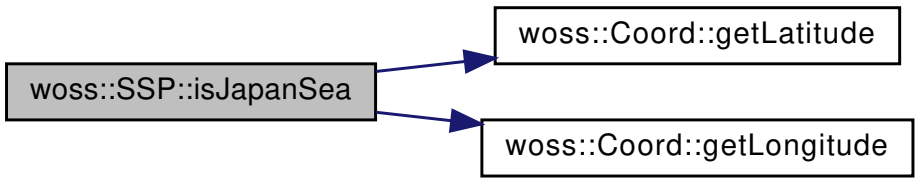
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.47 isMediterraneanSea() `bool woss::SSP::isMediterraneanSea ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in Mediterranean Sea

Parameters

coordinates	coordinates value

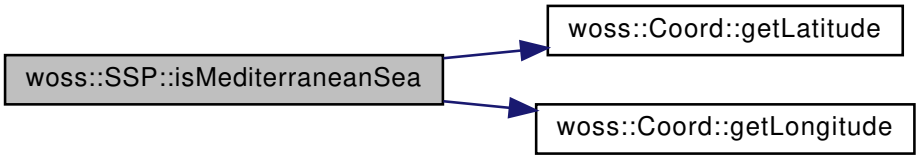
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.48 isNEAtlanticOcean() `bool woss::SSP::isNEAtlanticOcean ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in north eastern Atlantic Ocean

Parameters

coordinates	coordinates value

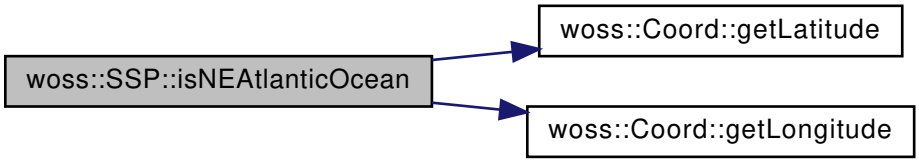
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.49 isRandomizable() `virtual bool woss::SSP::isRandomizable ( ) const [inline], [virtual]`

Checks if the sound speed profile provided can be randomly perturbated

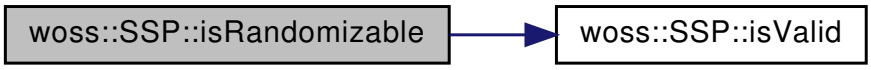
Returns

true if it can, *false* otherwise

References isValid(), pressure_map, salinity_map, and temperature_map.

Referenced by fullRandomize(), and transform().

Here is the call graph for this function:



13.65.4.50 isRedSea() `bool woss::SSP::isRedSea ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in Red Sea

Parameters

coordinates	coordinates value

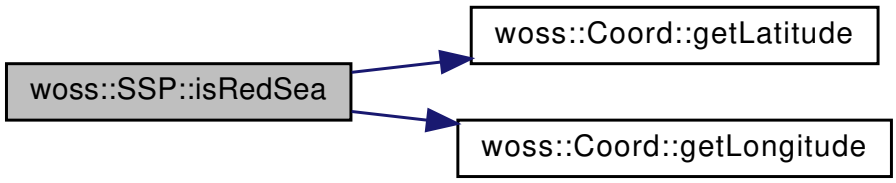
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.51 isSuluSea() `bool woss::SSP::isSuluSea ( const Coord & coordinates ) const [inline], [private]`

Checks if coordinates provided are in Sulu Sea

Parameters

coordinates	coordinates value

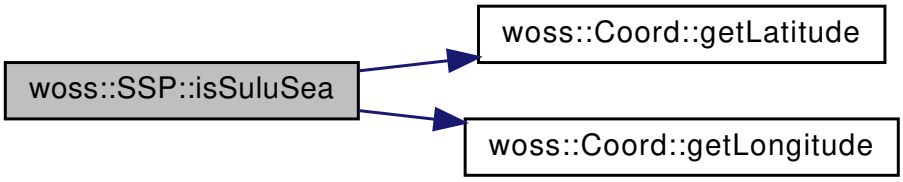
Returns

true if condition holds, *false* otherwise

References woss::Coord::getLatitude(), and woss::Coord::getLongitude().

Referenced by getDepthCorreptions(), and getPressureCorreptions().

Here is the call graph for this function:



13.65.4.52 isTransformable() `virtual bool woss::SSP::isTransformable ( ) const [inline], [virtual]`

Checks if the sound speed profile provided can be transformed

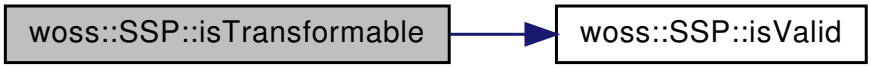
Returns

true if it can, *false* otherwise

References isValid().

Referenced by woss::ACToolboxWoss::initSSPVector(), and transform().

Here is the call graph for this function:



13.65.4.53 isValid()

```
virtual bool woss::SSP::isValid ( ) const  [inline], [virtual]
```

Checks the validity of sound speed profile provided

Returns

true if it has at least one value, *false* otherwise

References ssp_map.

Referenced by woss::ACToolboxWoss::initSSPVector(), isRandomizable(), isTransformable(), and randomize().

13.65.4.54 k()

```
double woss::SSP::k (
    double z,
    double lat ) const  [inline], [private]
```

Equation for pressure from depth conversion (Leroy and Parthiot)

Parameters

z	depth [m]
lat	latitude [decimal degree]

References g().

Referenced by h().

Here is the call graph for this function:

13.65.4.55 lower_bound() DConstIter woss::SSP::lower_bound (const PDouble & *depth*) const [inline]

Returns a const iterator to the sound speed value with key >= of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to end() if *depth* is not found

References ssp_map.

13.65.4.56 operator<<() friend::std::ostream & woss::SSP::operator<< (::std::ostream & *os*, const SSP & *instance*)

<< operator

Parameters

os	left operand ostream reference
instance	right operand const SSP reference

Returns

os reference after the operation

13.65.4.57 operator=() SSP & SSP::operator= (const SSP & *x*)

Assignment operator

Parameters

copy	const reference to a SSP object to be copied

Returns

SSP reference to *this*

References depth_precision, max_ssp_value, min_ssp_value, pressure_map, salinity_map, ssp_map, and temperature_map.

13.65.4.58 operator>>() friend::std::ostream & woss::SSP::operator>> (::std::istream & *is*, const SSP & *instance*)

operator

Parameters

os	left operand ostream reference
instance	right operand const SSP reference

Returns

os reference after the operation

13.65.4.59 pressure_begin() DConstIter woss::SSP::pressure_begin ( ) const [inline]

Returns a const iterator to the begin of the pressure map

Returns

const iterator

References pressure_map.

13.65.4.60 pressure_end() DConstIter woss::SSP::pressure_end ( ) const [inline]

Returns a const iterator to the end of the pressure map

Returns

const iterator

References pressure_map.

13.65.4.61 pressure_find() DConstIter woss::SSP::pressure_find (const PDouble & *depth*) const [inline]

Returns a const iterator to the pressure value with key == *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to pressure_end() if *depth* is not found

References pressure_map.

13.65.4.62 pressure_lower_bound() `DConstIter woss::SSP::pressure_lower_bound ( const PDouble & depth ) const [inline]`

Returns a const iterator to the pressure value with key >= of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to pressure_end() if *depth* is not found

References pressure_map.

13.65.4.63 pressure_rbegin() `DConstRIter woss::SSP::pressure_rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse begin of the pressure map

Returns

const reverse iterator

References pressure_map.

13.65.4.64 pressure_rend() `DConstRIter woss::SSP::pressure_rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the pressure map

Returns

const reverse iterator

References pressure_map.

13.65.4.65 pressure_upper_bound() `DConstIter woss::SSP::pressure_upper_bound ( const PDouble & depth ) const [inline]`

Returns a const iterator to the pressure value with key > of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to pressure_end() if *depth* is not found

References pressure_map.

13.65.4.66 randomize()

```
SSP * SSP::randomize (
        double ratio_incr_value ) const  [virtual]
```

Performs a random perturbation of sound speed values with given ratio

Parameters

ratio_incr_value	perturbation ratio

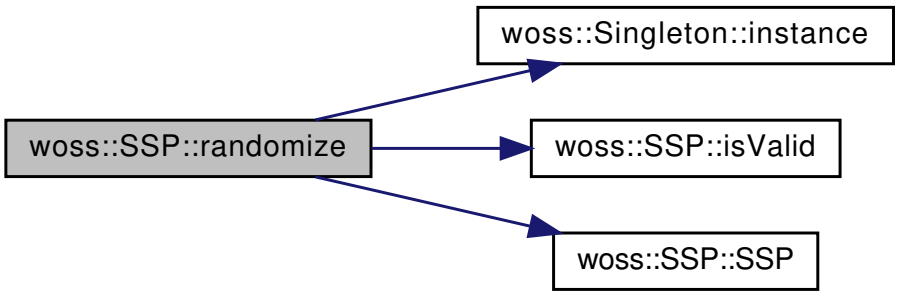
Returns

a *new* SSP object

References depth_precision, woss::Singleton< T >::instance(), isValid(), SSP(), and ssp_map.

Referenced by woss::BellhopWoss::writeNormalizedSSP().

Here is the call graph for this function:



13.65.4.67 rbegin()

```
DConstRIter woss::SSP::rbegin ( ) const  [inline]
```

Returns a const reverse iterator to the reverse beginning of the sound speed map

Returns

const reverse iterator

References ssp_map.

13.65.4.68 rend() DConstRIter woss::SSP::rend ( ) const [inline]

Returns a const reverse iterator to the reverse end of the sound speed map

Returns

const reverse iterator

References ssp_map.

13.65.4.69 salinity_begin() DConstIter woss::SSP::salinity_begin ( ) const [inline]

Returns a const iterator to the beginning of the salinity map

Returns

const reverse iterator

References salinity_map.

13.65.4.70 salinity_end() DConstIter woss::SSP::salinity_end ( ) const [inline]

Returns a const iterator to the end of the salinity map

Returns

const reverse iterator

References salinity_map.

13.65.4.71 salinity_find() DConstIter woss::SSP::salinity_find (const PDouble & *depth*) const [inline]

Returns a const iterator to the salinity value with key $>$ of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to salinity_end() if *depth* is not found

References salinity_map.

13.65.4.72 salinity_lower_bound() DConstIter woss::SSP::salinity_lower_bound (const PDouble & *depth*) const [inline]

Returns a const iterator to the salinity value with key >= of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to salinity_end() if *depth* is not found

References salinity_map.

13.65.4.73 salinity_rbegin() DConstRIter woss::SSP::salinity_rbegin ( ) const [inline]

Returns a const reverse iterator to the reverse beginning of the salinity map

Returns

const reverse iterator

References salinity_map.

13.65.4.74 salinity_rend() DConstRIter woss::SSP::salinity_rend ( ) const [inline]

Returns a const reverse iterator to the reverse end of the salinity map

Returns

const reverse iterator

References salinity_map.

13.65.4.75 salinity_upper_bound() DConstIter woss::SSP::salinity_upper_bound (const PDouble & *depth*) const [inline]

Returns a const iterator to the salinity value with key > of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to salinity_end() if *depth* is not found

References salinity_map.

13.65.4.76 setDebug() `static void woss::SSP::setDebug ( bool flag ) [inline], [static]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.65.4.77 setDepthPrecision() `void SSP::setDepthPrecision ( long double prec ) [virtual]`

Sets the sound speed profile depth precision for all PDouble depth values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

prec	depth precision [m]

References depth_precision, pressure_map, salinity_map, ssp_map, and temperature_map.

13.65.4.78 setSSPEqType() `SSP & woss::SSP::setSSPEqType ( SSPEqType eq_type ) [inline]`

Sets the SSP EQ that should be used by when inserting data

Parameters

eq_type	valid SSPEqType

Returns

reference to ∗**this**

References ssp_eq_type.

13.65.4.79 size() `int woss::SSP::size ( ) const [inline]`

Returns the number of sound speed values stored

Returns

number of sound speed values stored

References ssp_map.

Referenced by woss::ACToolboxWoss::initSSPVector().

13.65.4.80 temperature_begin() `DConstIter woss::SSP::temperature_begin ( ) const [inline]`

Returns a const iterator to the beginning of the temperature map

Returns

const reverse iterator

References temperature_map.

13.65.4.81 temperature_end() `DConstIter woss::SSP::temperature_end ( ) const [inline]`

Returns a const iterator to the end of the temperature map

Returns

const reverse iterator

References temperature_map.

13.65.4.82 temperature_find() `DConstIter woss::SSP::temperature_find ( const PDouble & depth ) const [inline]`

Returns a const iterator to the temperature value with key == of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to temperature_end() if *depth* is not found

References temperature_map.

13.65.4.83 temperature_lower_bound()

```
DConstIter woss::SSP::temperature_lower_bound (
        const PDouble & depth ) const  [inline]
```

Returns a const iterator to the temperature value with key >= of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to temperature_end() if *depth* is not found

References temperature_map.

13.65.4.84 temperature_rbegin()

```
DConstRIter woss::SSP::temperature_rbegin ( ) const  [inline]
```

Returns a const reverse iterator to the reverse beginning of the temperature map

Returns

const reverse iterator

References temperature_map.

13.65.4.85 temperature_rend()

```
DConstRIter woss::SSP::temperature_rend ( ) const  [inline]
```

Returns a const reverse iterator to the reverse end of the temperature map

Returns

const reverse iterator

References temperature_map.

13.65.4.86 temperature_upper_bound()

```
DConstIter woss::SSP::temperature_upper_bound (
        const PDouble & depth ) const  [inline]
```

Returns a const iterator to the temperature value with key > of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to temperature_end() if *depth* is not found

References temperature_map.

13.65.4.87 thyh()

```
double woss::SSP::thyh (
    double z ) const  [inline], [private]
```

Equation for pressure from depth conversion (Leroy and Parthiot)

Parameters

z	depth [m]

Referenced by getPressureFromDepth().

13.65.4.88 transform()

```
SSP * SSP::transform (
    const Coord & coordinates,
    double new_min_depth = -HUGE_VAL,
    double new_max_depth = HUGE_VAL,
    int total_depth_steps = SSP_CUSTOM_DEPTH_STEPS ) const  [virtual]
```

Transform the sound speed profile. If the current ssp can't be transformed it returns a not valid SSP. If both depth are not changed while depth steps is increased, all data will be linearly interpolated. If the SSP is extended in depth, sound speed will be calculated from last known temperature and salinity, while pressure will be calculated from depth conversion.

Parameters

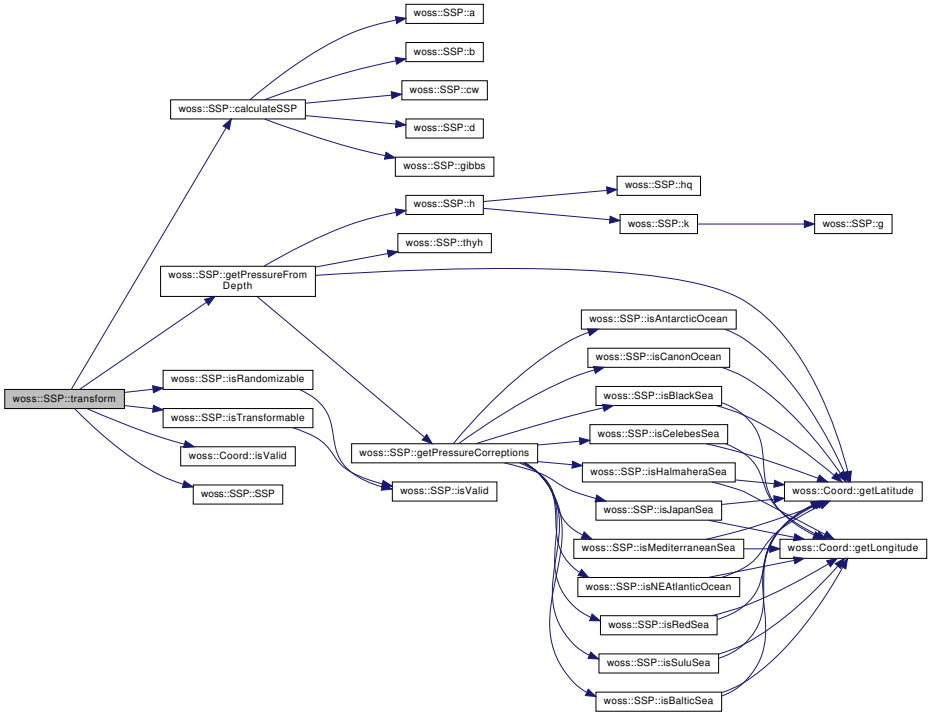
coordinates	coordinates for pressure, depth conversion corrections
new_min_depth	miminum depth [m] of the new SSP
new_max_depth	maximum depth [m] of the new SSP
total_depth_steps	number of depths of the new SSP

Returns

a *new* SSP object

References calculateSSP(), depth_precision, getPressureFromDepth(), isRandomizable(), isTransformable(), woss::Coord::isValid(), pressure_map, salinity_map, SSP(), ssp_map, and temperature_map.

Here is the call graph for this function:



13.65.4.89 truncate()

```
SSP * SSP::truncate (
        double max_depth ) const  [virtual]
```

It returns a new SSP object, truncated at the input depth. If the current ssp can't be truncated it returns a not valid SSP.

Parameters

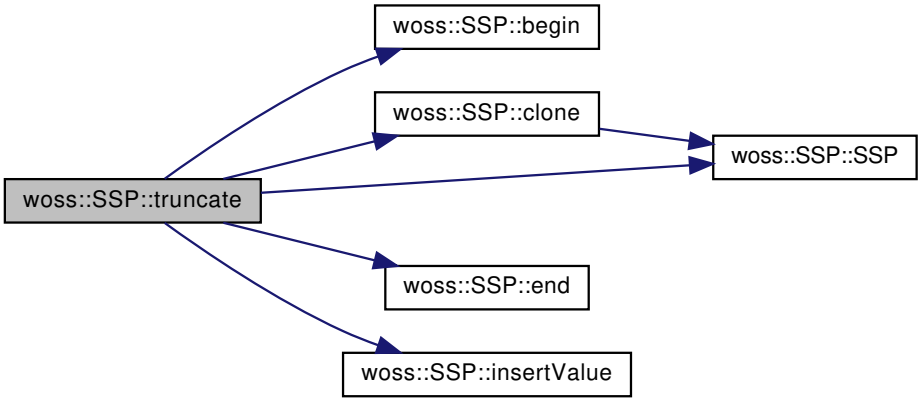
max_depth	truncation depth [m]

Returns

a *new* SSP object

References begin(), clone(), debug, end(), insertValue(), SSP(), and ssp_map.

Here is the call graph for this function:



13.65.4.90 upper_bound() `DConstIter woss::SSP::upper_bound ( const PDouble & depth ) const [inline]`

Returns a const iterator to the sound speed value with key $>$ of *depth* parameter

Parameters

depth	const reference to a PDouble depth value

Returns

const iterator to end() if *depth* is not found

References ssp_map.

13.65.4.91 write() `bool SSP::write ( ::std::ostream & stream_out ) const [virtual]`

Write values out to the given stream

Parameters

stream_out	const reference to an ostream instance

Returns

true if method was successful, false otherwise

References pressure_map, salinity_map, ssp_map, and temperature_map.

13.65.5 Friends And Related Function Documentation

13.65.5.1 operator"!= `bool operator!= (`

```
const SSP & left,
const SSP & right )  [friend]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.65.5.2 operator* [1/3] `const SSP operator* (`

```
const double left,
const SSP & right )  [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.3 operator* [2/3] `const SSP operator* (`

```
const SSP & left,
const double right )  [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.4 operator* [3/3]

```
const SSP operator* (
    const SSP & left,
    const SSP & right )  [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.5 operator*= [1/2]

```
SSP & operator*= (
    SSP & left,
    const double right )  [friend]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.6 operator*= [2/2]

```
SSP & operator*= (
    SSP & left,
    const SSP & right )  [friend]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.7 operator+ [1/3]

```
const SSP operator+ (
    const double left,
    const SSP & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.8 operator+ [2/3]

```
const SSP operator+ (
    const SSP & left,
    const double right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.9 operator+ [3/3]

```
const SSP operator+ (
    const SSP & left,
    const SSP & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a const instance holding the operation result

13.65.5.10 operator+= [1/2]

```
SSP & operator+= (
    SSP & left,
    const double right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.11 operator+= [2/2]

```
SSP & operator+= (
    SSP & left,
    const SSP & right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.12 operator- [1/3]

```
const SSP operator- (
    const double left,
    const SSP & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.13 operator- [2/3]

```
const SSP operator- (
    const SSP & left,
    const double right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.14 operator- [3/3]

```
const SSP operator- (
    const SSP & left,
    const SSP & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.15 operator-= [1/2]

```
SSP & operator-= (
    SSP & left,
    const double right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.16 operator-= [2/2]

```
SSP & operator-= (
    SSP & left,
    const SSP & right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.17 operator/ [1/3] const SSP operator/ (const double *left*, const SSP & *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.18 operator/ [2/3] const SSP operator/ (const SSP & *left*, const double *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.19 operator/ [3/3] const SSP operator/ (const SSP & *left*, const SSP & *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.65.5.20 operator/= [1/2]

```
SSP & operator/= (
        SSP & left,
        const double right )  [friend]
```

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.21 operator/= [2/2]

```
SSP & operator/= (
        SSP & left,
        const SSP & right )  [friend]
```

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.65.5.22 operator==

```
bool operator== (
        const SSP & left,
        const SSP & right )  [friend]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.65.6 Member Data Documentation

13.65.6.1 debug `bool SSP::debug = false [static], [protected]`

Stores the common debug flag

Referenced by setDebug(), and truncate().

13.65.6.2 depth_precision `long double woss::SSP::depth_precision [protected]`

Stores the precision of all PDouble depth instances [m]

Referenced by create(), fullRandomize(), getDepthPrecision(), insertValue(), operator=(), randomize(), setDepthPrecision(), SSP(), and transform().

13.65.6.3 max_ssp_value `double woss::SSP::max_ssp_value [protected]`

Stores the maximum sound speed value [m/s]

Referenced by getMaxSSPValue(), insertValue(), operator=(), and SSP().

13.65.6.4 min_ssp_value `double woss::SSP::min_ssp_value [protected]`

Stores the miminum sound speed value [m/s]

Referenced by getMinSSPValue(), insertValue(), operator=(), and SSP().

13.65.6.5 pressure_map `DepthMap woss::SSP::pressure_map [protected]`

Pressure values map

Referenced by clear(), fullRandomize(), insertValue(), isRandomizable(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), pressure_begin(), pressure_end(), pressure_find(), pressure_lower_bound(), pressure_rbegin(), pressure_rend(), pressure_upper_bound(), setDepthPrecision(), SSP(), transform(), and write().

13.65.6.6 salinity_map

`DepthMap woss::SSP::salinity_map [protected]`

Salinity values map

Referenced by clear(), fullRandomize(), insertValue(), isRandomizable(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), salinity_begin(), salinity_end(), salinity_find(), salinity_lower_bound(), salinity_rbegin(), salinity_rend(), salinity_upper_bound(), setDepthPrecision(), SSP(), transform(), and write().

13.65.6.7 ssp_eq_type

`SSPEqType woss::SSP::ssp_eq_type [protected]`

Stores the SSP eq type that should be used

Referenced by calculateSSP(), getSSPEqType(), setSSPEqType(), and SSP().

13.65.6.8 ssp_map

`DepthMap woss::SSP::ssp_map [protected]`

Sound speed values map

Referenced by at(), begin(), clear(), create(), empty(), end(), eraseValue(), findValue(), fullRandomize(), getMaxDepthValue(), getMinDepthValue(), insertValue(), isValid(), lower_bound(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), randomize(), rbegin(), rend(), setDepthPrecision(), size(), SSP(), transform(), truncate(), upper_bound(), and write().

13.65.6.9 temperature_map

`DepthMap woss::SSP::temperature_map [protected]`

Temperature values map

Referenced by clear(), fullRandomize(), insertValue(), isRandomizable(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), setDepthPrecision(), SSP(), temperature_begin(), temperature_end(), temperature_find(), temperature_lower_bound(), temperature_rbegin(), temperature_rend(), temperature_upper_bound(), transform(), and write().

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

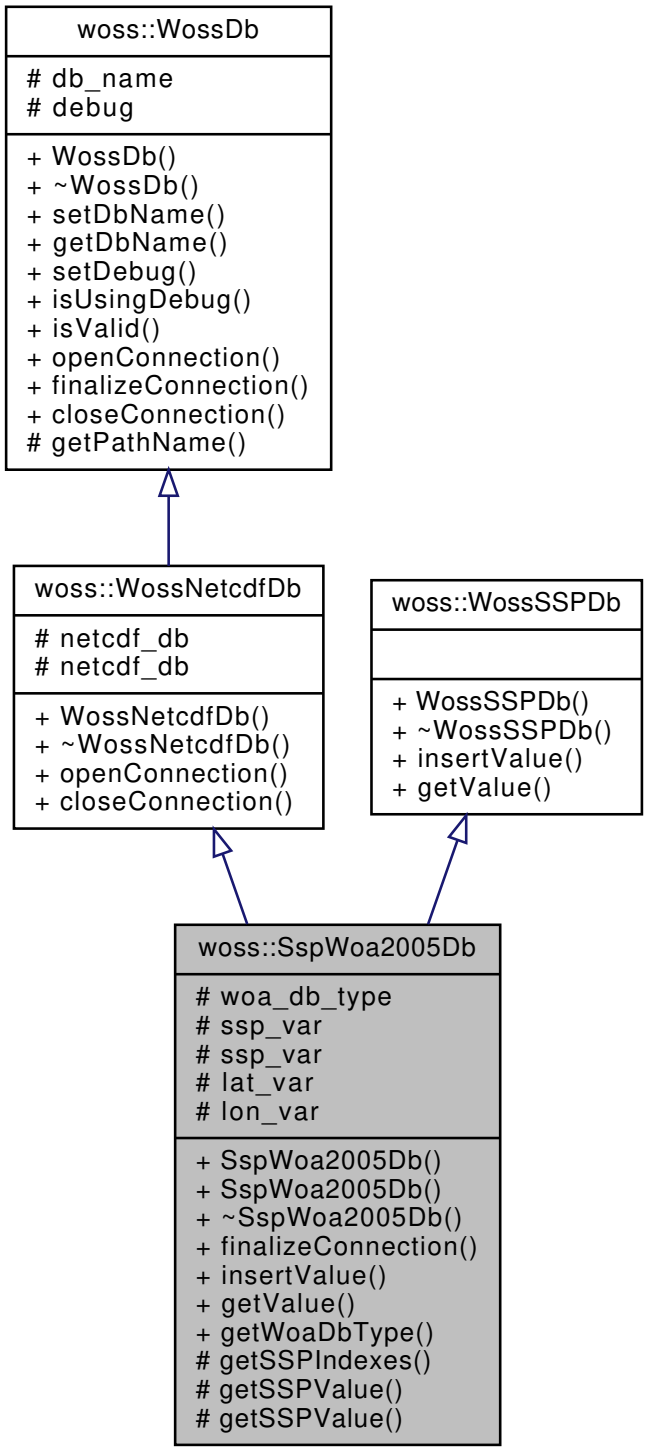
- woss/woss_def/ssp-definitions.h
- woss/woss_def/ssp-definitions.cpp

13.66 woss::SspWoa2005Db Class Reference

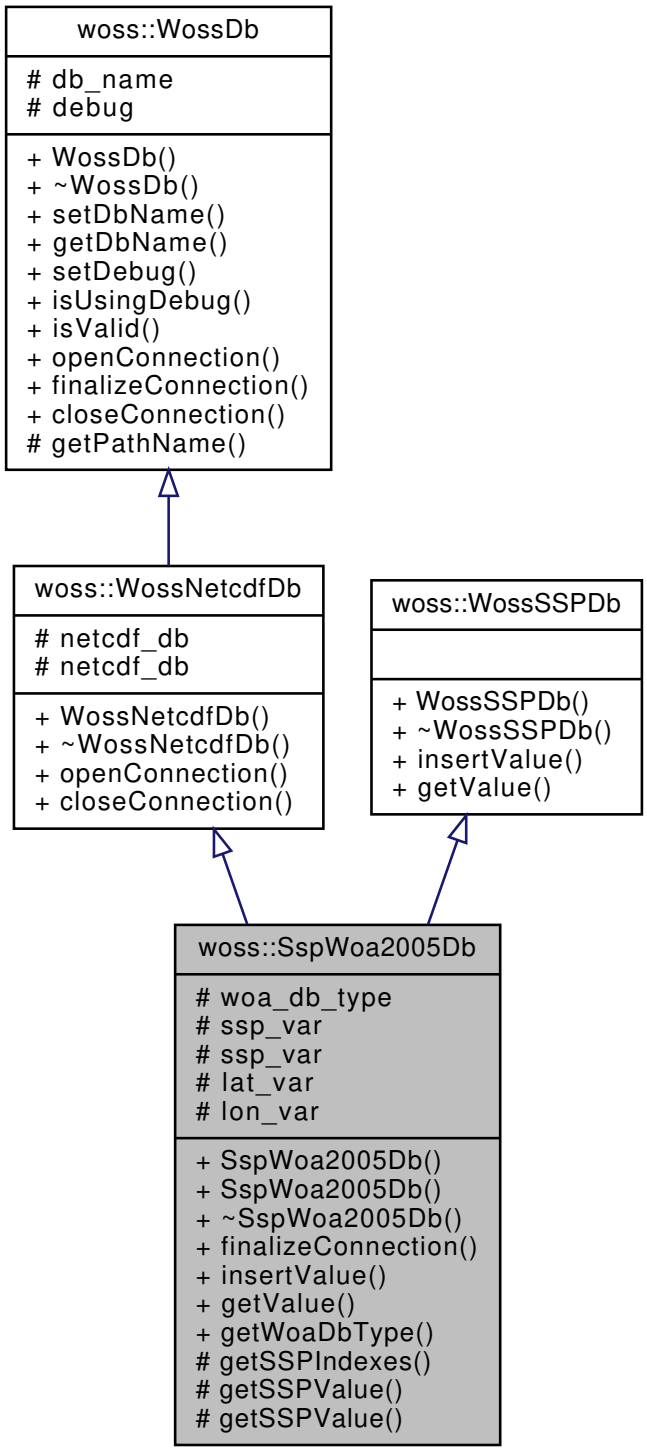
WossDb for the custom made NetCDF WOA2005 SSP database.

`#include <ssp-woa2005-db.h>`

Inheritance diagram for woss::SspWoa2005Db:



Collaboration diagram for woss::SspWoa2005Db:



Public Member Functions

- SspWoa2005Db (const ::std::string &name)
- SspWoa2005Db (const ::std::string &name, WOADbType db_type)
- virtual bool finalizeConnection ()
- virtual bool insertValue (const Coord &coordinates, const Time &time_value, const SSP &ssp_value)
- virtual SSP * getValue (const Coord &coordinates, const Time &time, long double ssp_depth_precision) const
- WOADbType getWoaDbType () const

Protected Member Functions

- SSPIndexes getSSPIndexes (const Coord &coordinates) const
- void getSSPValue (const Coord &coordinates, const SSPIndexes &indexes, double ssp_values[]) const
- void getSSPValue (const SSPIndexes &indexes, double ssp_values[]) const

Protected Attributes

- WOADbType woa_db_type
- netCDF::NcVar ssp_var
- NcVar * **ssp_var**
- netCDF::NcVar lat_var
- netCDF::NcVar lon_var

13.66.1 Detailed Description

WossDb for the custom made NetCDF WOA2005 SSP database.

WossDb for the custom made NetCDF WOA2005 SSP database

13.66.2 Constructor & Destructor Documentation

13.66.2.1 SspWoa2005Db() [1/2]

```
SspWoa2005Db::SspWoa2005Db (
    const ::std::string & name )
```

SspWoa2005Db constructor

Parameters

name	pathname of database

References woss::WOA_DB_TYPE_2005.

13.66.2.2 SspWoa2005Db() [2/2]

```
SspWoa2005Db::SspWoa2005Db (
    const ::std::string & name,
    WOADbType db_type )
```

SspWoa2005Db constructor

Parameters

name	pathname of database
db_type	WOADbType of the database

13.66.3 Member Function Documentation

13.66.3.1 finalizeConnection() `bool SspWoa2005Db::finalizeConnection ( ) [virtual]`

Post openConnection() actions, used to create and initialize NetCDF variables

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References lat_var, lon_var, woss::WossNetcdfDb::netcdf_db, ssp_var, woa_db_type, woss::WOA_DB_TYPE_2005, and woss::WOA_DB_TYPE_2013.

13.66.3.2 getSSPIndexes() `SSPIndexes SspWoa2005Db::getSSPIndexes ( const Coord & coordinates ) const [protected]`

Returns the indexes used by the NetCDF variable to get the SSP values

Parameters

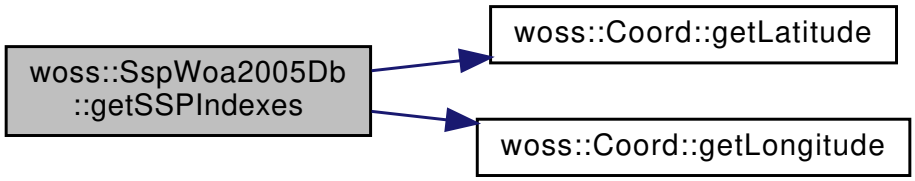
coordinates	const reference to a valid Coord object

Returns

SSPIndexes value

References woss::Coord::getLatitude(), woss::Coord::getLongitude(), woa_db_type, and woss::WOA_DB_TYPE_2005.

Here is the call graph for this function:



13.66.3.3 getSSPValue() [1/2] `void SspWoa2005Db::getSSPValue ( const Coord & coordinates, const SSPIndexes & indexes, double ssp_values[] ) const [protected]`

Insert the SSP values taken from the given indexes into the given array

Parameters

coordinates	const reference to a valid Coord object
indexes	const reference to a valid SSPIndexes object
ssp_values[]	array that will hold SSP values

References woss::WOA_DB_TYPE_2005.

13.66.3.4 getSSPValue() [2/2]

```
void woss::SspWoa2005Db::getSSPValue (
    const SSPIndexes & indexes,
    double ssp_values[] ) const  [protected]
```

Insert the SSP values taken from the given indexes into the given array

Parameters

indexes	const reference to a valid SSPIndexes object
ssp_values[]	array that will hold SSP values

13.66.3.5 getValue()

```
SSP * SspWoa2005Db::getValue (
    const Coord & coordinates,
    const Time & time,
    long double ssp_depth_precision ) const  [virtual]
```

Returns a pointer to a heap-based SSP for given coordinates and date time if both present in the database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid Coord object
time	const reference to a valid Time object
ssp_depth_precision	ssp depth precision [m]

Returns

valid SSP if coordinates and time date are found, *not valid* otherwise

Implements woss::WossSSPDb.

13.66.3.6 getWoaDbType()

```
WOADbType woss::SspWoa2005Db::getWoaDbType ( ) const  [inline]
```

Returns current WOADbType

Returns

current WOADbType

References woa_db_type.

13.66.3.7 insertValue()

```
bool SspWoa2005Db::insertValue (
    const Coord & coordinates,
    const Time & time_value,
    const SSP & ssp_value )  [virtual]
```

Inserts the given woss::SSP value in the database for given coordinates

Parameters

coordinates	const reference to a valid Coord object
time_value	const reference to a valid Time object
ssp_value	const Reference to SSP value to be inserted

Returns

true if method was successful, *false* otherwise

Implements woss::WossSSPDb.

13.66.4 Member Data Documentation

13.66.4.1 lat_var

```
netCDF::NcVar woss::SspWoa2005Db::lat_var  [protected]
```

NetCDF variable representing latitude

Referenced by finalizeConnection().

13.66.4.2 lon_var

```
netCDF::NcVar woss::SspWoa2005Db::lon_var  [protected]
```

NetCDF variable representing longitude

Referenced by finalizeConnection().

13.66.4.3 ssp_var `netCDF::NcVar woss::SspWoa2005Db::ssp_var [protected]`

NetCDF variable representing SSP

Referenced by finalizeConnection().

13.66.4.4 woa_db_type `WOADbType woss::SspWoa2005Db::woa_db_type [protected]`

WOA Db type

Referenced by finalizeConnection(), getSSPIndexes(), and getWoaDbType().

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

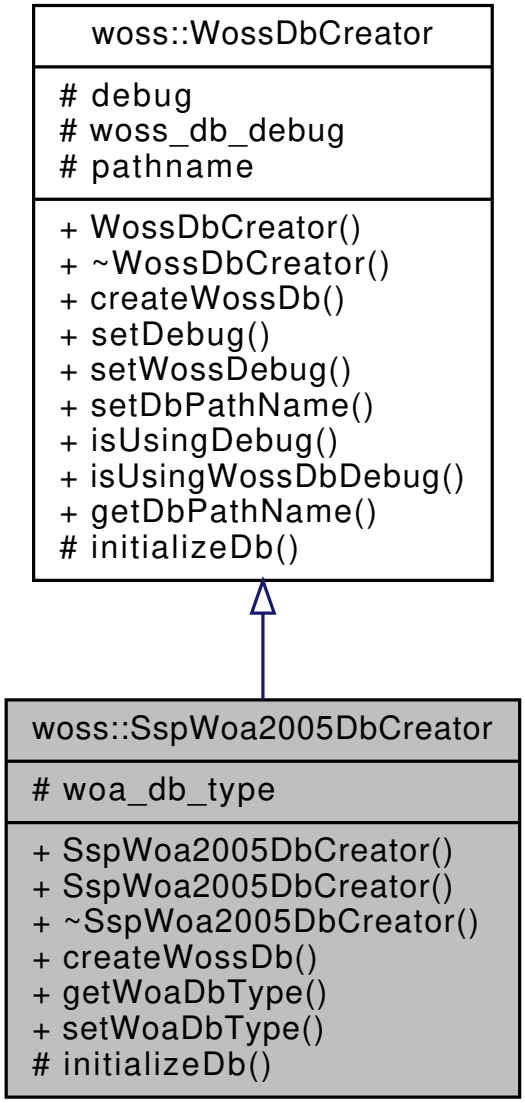
- woss/woss_db/ssp-woa2005-db.h
- woss/woss_db/ssp-woa2005-db.cpp

13.67 woss::SspWoa2005DbCreator Class Reference

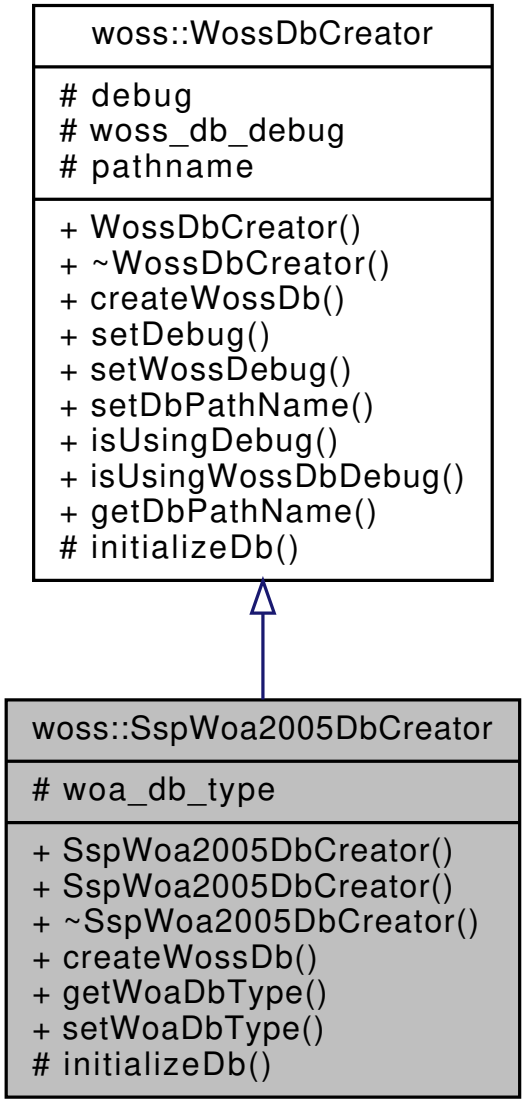
WossDbCreator for the custom made NetCDF WOA2005 SSP database.

```
#include <ssp-woa2005-db-creator.h>
```

Inheritance diagram for woss::SspWoa2005DbCreator:



Collaboration diagram for woss::SspWoa2005DbCreator:



Public Member Functions

- SspWoa2005DbCreator ()
- SspWoa2005DbCreator (WOADbType db_type)
- virtual WossDb ∗const createWossDb ()
- WOADbType getWoaDbType () const
- SspWoa2005DbCreator & setWoaDbType (WOADbType type)

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)

Protected Attributes

- WOADbType **woa_db_type**

13.67.1 Detailed Description

WossDbCreator for the custom made NetCDF WOA2005 SSP database.

Specialization of WossDbCreator for the custom made NetCDF WOA2005 SSP database

13.67.2 Constructor & Destructor Documentation

13.67.2.1 SspWoa2005DbCreator() [1/2] SspWoa2005DbCreator::SspWoa2005DbCreator ( )

Default SspWoa2005DbCreator constructor

References woss::WOA_DB_TYPE_2005.

13.67.2.2 SspWoa2005DbCreator() [2/2] SspWoa2005DbCreator::SspWoa2005DbCreator (WOADbType *db_type*)

SspWoa2005DbCreator constructor

Parameters

db_type	WOADbType of the database

13.67.3 Member Function Documentation

13.67.3.1 createWossDb() WossDb *const SspWoa2005DbCreator::createWossDb ( ) [virtual]

This method is called to create and initialize a SspWoa2005Db

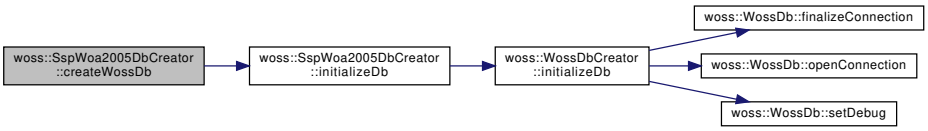
Returns

a pointer to a properly initialized SspWoa2005Db object

Implements woss::WossDbCreator.

References initializeDb(), and woss::WossDbCreator::pathname.

Here is the call graph for this function:



13.67.3.2 getWoaDbType() `WOADbType woss::SspWoa2005DbCreator::getWoaDbType ( ) const [inline]`

Returns current WOADbType

Returns

current WOADbType

13.67.3.3 initializeDb() `bool SspWoa2005DbCreator::initializeDb ( WossDb *const woss_db ) [protected], [virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created SspWoa2005Db

Returns

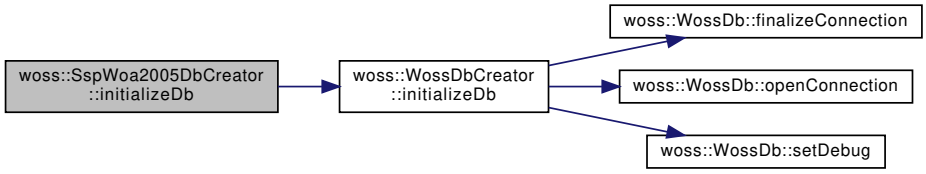
true if the method succeed, *false* otherwise

Implements woss::WossDbCreator.

References woss::WossDbCreator::initializeDb().

Referenced by createWossDb().

Here is the call graph for this function:



13.67.3.4 setWoaDbType() `SspWoa2005DbCreator & woss::SspWoa2005DbCreator::setWoaDbType ( WOADbType type ) [inline]`

Set current WOADbType

Parameters

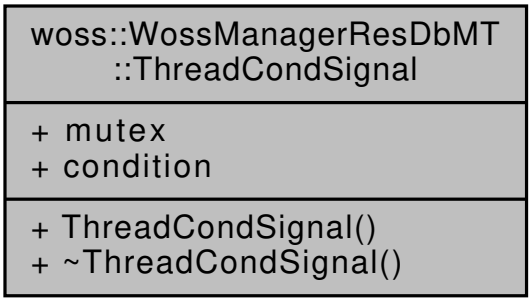
type	valid WOADbType

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

- woss/woss_db/ssp-woa2005-db-creator.h
- woss/woss_db/ssp-woa2005-db-creator.cpp

13.68 woss::WossManagerResDbMT::ThreadCondSignal Struct Reference

Collaboration diagram for woss::WossManagerResDbMT::ThreadCondSignal:



Public Attributes

- pthread_mutex_t **mutex**
- pthread_cond_t **condition**

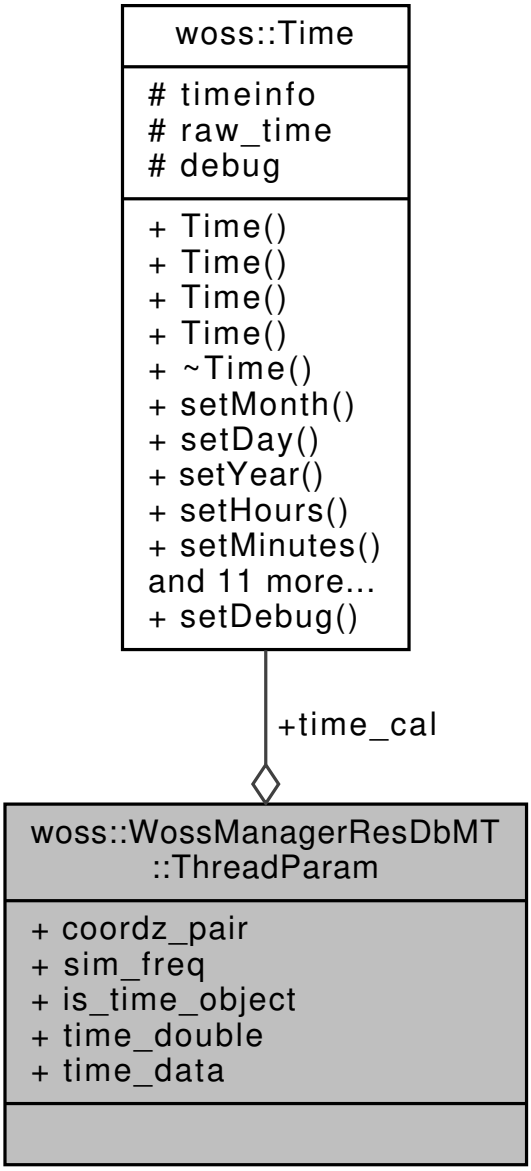
The documentation for this struct was generated from the following file:

- woss/woss-manager.h

13.69 woss::WossManagerResDbMT::ThreadParam Struct Reference

```
#include <woss-manager.h>
```

Collaboration diagram for woss::WossManagerResDbMT::ThreadParam:



Public Attributes

- CoordZPair **coordz_pair**
- SimFreq **sim_freq**
-

  struct {
  bool **is_time_object**
  Time **time_cal**
  double **time_double**
  } **time_data**

13.69.1 Detailed Description

Type used by an active query thread

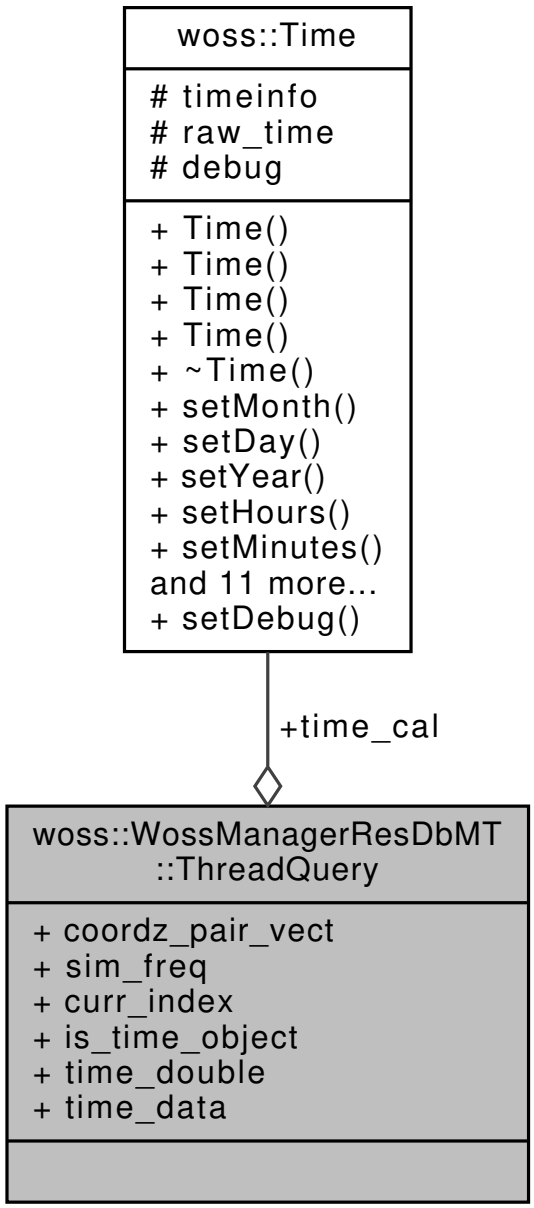
The documentation for this struct was generated from the following file:

- woss/woss-manager.h

13.70 woss::WossManagerResDbMT::ThreadQuery Struct Reference

`#include <woss-manager.h>`

Collaboration diagram for woss::WossManagerResDbMT::ThreadQuery:



Public Attributes

- CoordZPairVect **coordz_pair_vect**
- SimFreq **sim_freq**
- int **curr_index**
- struct {
  bool **is_time_object**
  Time **time_cal**
  double **time_double**
  } **time_data**

13.70.1 Detailed Description

Type for storing current queries

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

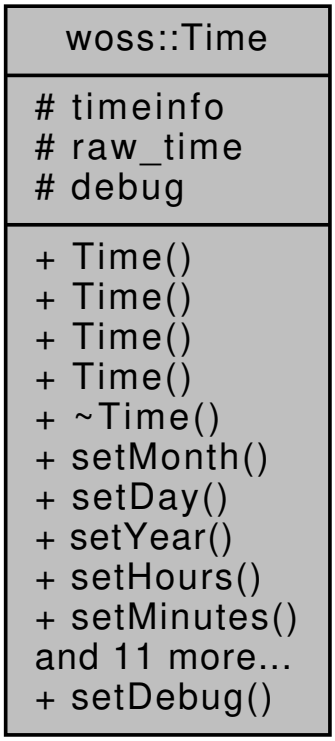
- woss/woss-manager.h

13.71 woss::Time Class Reference

a class for time date manipulation

`#include <time-definitions.h>`

Collaboration diagram for woss::Time:



Public Member Functions

- Time ()
- Time (struct tm *time)
- Time (int day, int month, int year, int hours=0, int mins=0, int seconds=1)
- Time (const Time &copy)
- ~Time ()
- Time & setMonth (int m)
- Time & setDay (int d)
- Time & setYear (int y)
- Time & setHours (int h)
- Time & setMinutes (int m)
- Time & setSeconds (int s)
- bool isValid () const
- int getMonth () const
- int getDay () const
- int getHours () const
- int getYear () const
- int getMinutes () const
- int getSeconds () const
- operator time_t () const
- Time & operator= (const Time &copy)
- friend::std::ostream & operator<< (::std::ostream &os, const Time &instance)

Static Public Member Functions

- static void setDebug (bool flag)

Protected Attributes

- struct tm timeinfo
- time_t raw_time

Static Protected Attributes

- static bool debug = false

Friends

- const Time operator+ (const Time &left, const time_t right)
- const Time operator- (const Time &left, const time_t right)
- double operator- (const Time &left, const Time &right)
- Time & operator+= (Time &left, time_t right)
- Time & operator-= (Time &left, time_t right)
- bool operator== (const Time &left, const Time &right)
- bool operator!= (const Time &left, const Time &right)
- bool operator> (const Time &left, const Time &right)
- bool operator< (const Time &left, const Time &right)
- bool operator<= (const Time &left, const Time &right)
- bool operator>= (const Time &left, const Time &right)

13.71.1 Detailed Description

a class for time date manipulation

Time class offers the possibility to store and manipulate date time. A time date consists of a day, month, year, hours, minutes and seconds.

13.71.2 Constructor & Destructor Documentation

13.71.2.1 Time() [1/4] Time::Time ( )

Default Time constructor. Object is not valid

References raw_time, and timeinfo.

13.71.2.2 Time() [2/4] Time::Time (struct tm * time)

Time constructor

Parameters

time	**struct tm** from ctime library

References raw_time, and timeinfo.

13.71.2.3 Time() [3/4] Time::Time (int day, int month, int year, int hours = 0, int mins = 0, int seconds = 1)

Time constructor

Parameters

day	day value. Should be between 1 and 31
month	month value. Should be between 1 and 12
year	year value.
hours	hours value. Should be between 0 and 23
mins	minutes value. Should be between 0 and 59
seconds	seconds value. Should be between 0 and 59

References raw_time, and timeinfo.

13.71.2.4 Time() [4/4] Time::Time (const Time & copy)

Time copy constructor

Parameters

copy	TimeArr to be copied

References raw_time, and timeinfo.

13.71.2.5 ~Time() woss::Time::~Time ( ) [inline]

Time destructor. It is not **virtual**, since this class is not meant to be inherited from

13.71.3 Member Function Documentation

13.71.3.1 getDay() int woss::Time::getDay ( ) const [inline]

Returns day value

Returns

day value between 1 and 31

References timeinfo.

13.71.3.2 getHours() int woss::Time::getHours ( ) const [inline]

Returns hours value

Returns

hours value between 0 and 23

References timeinfo.

13.71.3.3 getMinutes() `int woss::Time::getMinutes ( ) const [inline]`

Returns minutes value

Returns

minutes value between 0 and 59

References timeinfo.

13.71.3.4 getMonth() `int woss::Time::getMonth ( ) const [inline]`

Returns month value

Returns

month value between 1 and 12

References timeinfo.

13.71.3.5 getSeconds() `int woss::Time::getSeconds ( ) const [inline]`

Returns seconds value

Returns

seconds value between 0 and 59

References timeinfo.

13.71.3.6 getYear() `int woss::Time::getYear ( ) const [inline]`

Returns year value

Returns

year value

References timeinfo.

13.71.3.7 isValid() `bool woss::Time::isValid ( ) const [inline]`

Checks the validity of Time

Returns

true if it has a initialized date time, *false* otherwise

References raw_time.

Referenced by woss::BellhopCreator::createWoss(), woss::ResPressureTxtDb::getValue(), woss::ResTimeArrTxtDb::getValue(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDbMT::getW woss::WossManager::getWossTimeArr(), woss::ACToolboxWoss::isValid(), woss::BellhopWoss::timeEvolve(), woss::Altimetry::timeEvolve(), woss::AltimBretschneider::timeEvolve(), woss::WMSMTcreateThreadPressure(), woss::WMSMTcreateThreadTimeArr(), and woss::Woss::Woss().

13.71.3.8 operator time_t() `woss::Time::operator time_t ( ) const [inline]`

time_t operator

Returns

the time_t

13.71.3.9 operator<<() `friend::std::ostream & woss::Time::operator<< ( ::std::ostream & os, const Time & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Time reference

Returns

os reference after the operation

13.71.3.10 operator=() `Time & Time::operator= ( const Time & copy )`

Assignment operator

Parameters

copy	const reference to a Time object to be copied

Returns

Time reference to *this*

References raw_time, and timeinfo.

13.71.3.11 setDay() `Time & woss::Time::setDay ( int d ) [inline]`

Sets day

Parameters

d	day value. Should be between 1 and 31

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.3.12 setDebug() `static void woss::Time::setDebug ( bool flag ) [inline], [static]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.71.3.13 setHours() `Time & woss::Time::setHours ( int h ) [inline]`

Sets hours

Parameters

m	hours value. Should be between 0 and 23

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.3.14 setMinutes() `Time & woss::Time::setMinutes ( int m ) [inline]`

Sets minutes

Parameters

m	minutes value. Should be between 0 and 59

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.3.15 setMonth() `Time & woss::Time::setMonth ( int m ) [inline]`

Sets month

Parameters

m	month value. Should be between 1 and 12

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.3.16 setSeconds() `Time & woss::Time::setSeconds ( int s ) [inline]`

Sets seconds

Parameters

s	seconds value. Should be between 0 and 59

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.3.17 setYear() `Time & woss::Time::setYear ( int y ) [inline]`

Sets year

Parameters

y	year value

Returns

reference to ∗**this**

References raw_time, and timeinfo.

13.71.4 Friends And Related Function Documentation

13.71.4.1 operator"!= `bool operator!= ( const Time & left, const Time & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.71.4.2 operator+

```
const Time operator+ (
    const Time & left,
    const time_t right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.71.4.3 operator+=

```
Time & operator+= (
    Time & left,
    time_t right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const time_t representing seconds

Returns

left reference after the operation

13.71.4.4 operator- [1/2]

```
double operator- (
    const Time & left,
    const Time & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

time difference in seconds

13.71.4.5 operator- [2/2]

```
const Time operator- (
    const Time & left,
    const time_t right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.71.4.6 operator-=

```
Time & operator-= (
    Time & left,
    time_t right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const time_t representing seconds

Returns

left reference after the operation

13.71.4.7 operator<

```
bool operator< (
    const Time & left,
    const Time & right )  [friend]
```

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

13.71.4.8 operator<= `bool operator<= ( const Time & left, const Time & right ) [friend]`

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

13.71.4.9 operator== `bool operator== ( const Time & left, const Time & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

13.71.4.10 operator> `bool operator> ( const Time & left, const Time & right ) [friend]`

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

13.71.4.11 operator>= `bool operator>= ( const Time & left, const Time & right ) [friend]`

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

13.71.5 Member Data Documentation

13.71.5.1 debug `bool Time::debug = false [static], [protected]`

Debug flag

Referenced by setDebug().

13.71.5.2 raw_time `time_t woss::Time::raw_time [protected]`

Number of seconds corresponding to date time *timeinfo*

Referenced by isValid(), woss::operator+=(), woss::operator-=(), operator=(), setDay(), setHours(), setMinutes(), setMonth(), setSeconds(), setYear(), and Time().

13.71.5.3 timeinfo `struct tm woss::Time::timeinfo [protected]`

Struct tm from ctime library

Referenced by getDay(), getHours(), getMinutes(), getMonth(), getSeconds(), getYear(), woss::operator+=(), woss::operator-(), woss::operator-=(), operator=(), setDay(), setHours(), setMinutes(), setMonth(), setSeconds(), setYear(), and Time().

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

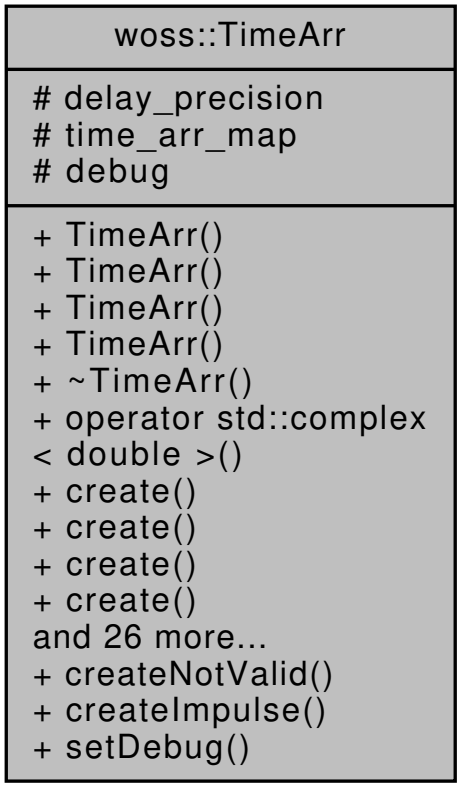
- woss/woss_def/time-definitions.h
- woss/woss_def/time-definitions.cpp

13.72 woss::TimeArr Class Reference

Channel power delay profile class.

`#include <time-arrival-definitions.h>`

Collaboration diagram for woss::TimeArr:



Public Member Functions

- TimeArr (long double custom_delay_prec=TIMEARR_CUSTOM_DELAY_PRECISION)
- TimeArr (TimeArrMap &map, long double custom_delay_prec=TIMEARR_CUSTOM_DELAY_PRECISION)
- TimeArr (const Pressure &pressure, double delay=TIMEARR_PRESSURE_CONVERSION_DELAY, long double custom_delay_prec=TIMEARR_CUSTOM_DELAY_PRECISION)
- TimeArr (const TimeArr &copy)
- virtual operator std::complex< double > () const
- virtual TimeArr * create (long double custom_delay_prec=TIMEARR_CUSTOM_DELAY_PRECISION) const
- virtual TimeArr * create (TimeArrMap &map, long double custom_delay_prec=TIMEARR_CUSTOM_↩ DELAY_PRECISION) const
- virtual TimeArr * create (const Pressure &pressure, double delay=TIMEARR_PRESSURE_CONVERSION↩ _DELAY, long double custom_delay_prec=TIMEARR_CUSTOM_DELAY_PRECISION) const
- virtual TimeArr * create (const TimeArr &copy) const
- virtual TimeArr * clone () const
- virtual TimeArr * createArray (unsigned int array_size) const
- TimeArr & insertValue (double delay, const Pressure &pressure)
- void sumValue (double delay, const Pressure &pressure)
- TimeArrCIt findValue (double delay) const
- TimeArr & eraseValue (double delay)

- virtual TimeArr * coherentSumSample (double time_delay)
- virtual TimeArr * incoherentSumSample (double time_delay)
- virtual TimeArr * crop (double time_start, double time_end)
- virtual bool checkPressureAttenuation (double distance, double frequency)
- TimeArrCIt begin () const
- TimeArrCIt end () const
- TimeArrCRIt rbegin () const
- TimeArrCRIt rend () const
- TimeArrCIt at (const int i) const
- TimeArrCIt lowerBoundTxLoss (double threshold_db) const
- int size () const
- bool empty () const
- void clear ()
- TimeArr & setDelayPrecision (long double precision)
- double getMaxDelayValue () const
- double getMinDelayValue () const
- long double getDelayPrecision () const
- virtual bool isValid () const
- virtual bool isConvertedFromPressure () const
- TimeArr & operator= (const TimeArr &copy)

Static Public Member Functions

- static TimeArrMap & createNotValid ()
- static TimeArrMap & createImpulse ()
- static void setDebug (bool flag)

Protected Attributes

- long double delay_precision
- TimeArrMap time_arr_map

Static Protected Attributes

- static bool debug = false

Friends

- bool operator== (const TimeArr &left, const TimeArr &right)
- bool operator!= (const TimeArr &left, const TimeArr &right)
- const TimeArr operator+ (const TimeArr &left, const TimeArr &right)
- const TimeArr operator- (const TimeArr &left, const TimeArr &right)
- const TimeArr operator+ (const TimeArr &left, const double right)
- const TimeArr operator- (const TimeArr &left, const double right)
- const TimeArr operator/ (const TimeArr &left, const double right)
- const TimeArr operator* (const TimeArr &left, const double right)
- const TimeArr operator+ (const double left, const TimeArr &right)
- const TimeArr operator- (const double left, const TimeArr &right)
- const TimeArr operator/ (const double left, const TimeArr &right)
- const TimeArr operator* (const double left, const TimeArr &right)
- TimeArr & operator+= (TimeArr &left, const TimeArr &right)
- TimeArr & operator-= (TimeArr &left, const TimeArr &right)
- TimeArr & operator+= (TimeArr &left, double right)
- TimeArr & operator-= (TimeArr &left, double right)
- TimeArr & operator/= (TimeArr &left, double right)
- TimeArr & operator*= (TimeArr &left, double right)
- std::ostream & operator<< (std::ostream &os, const TimeArr &instance)

13.72.1 Detailed Description

Channel power delay profile class.

TimeArr class offers the possibility to store and manipulate channel power delay profiles, e.g. a collection of time delay values associated to a Pressure attenuation value.

13.72.2 Constructor & Destructor Documentation

13.72.2.1 TimeArr() [1/4]

```
TimeArr::TimeArr (
        long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION )
```

Default TimeArr constructor

Parameters

custom_delay_prec	delay precision [s]

Referenced by clone(), and create().

13.72.2.2 TimeArr() [2/4]

```
TimeArr::TimeArr (
        TimeArrMap & map,
        long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION )
```

TimeArr constructor

Parameters

map	custom time arrival map
custom_delay_prec	delay precision [s]

References time_arr_map.

13.72.2.3 TimeArr() [3/4]

```
TimeArr::TimeArr (
        const Pressure & pressure,
        double delay = TIMEARR_PRESSURE_CONVERSION_DELAY,
        long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION )
```

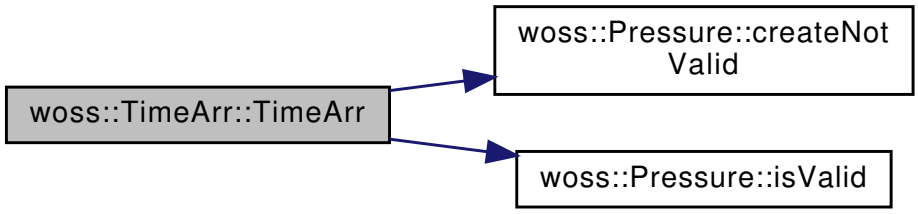
TimeArr constructor. Needed for Pressure to TimeArr conversion

Parameters

pressure	Pressure value
delay	delay value [s]
custom_delay_prec	delay precision [s]

References woss::Pressure::createNotValid(), woss::Pressure::isValid(), and time_arr_map.

Here is the call graph for this function:



13.72.2.4 TimeArr() [4/4] TimeArr::TimeArr (const TimeArr & *copy*)

TimeArr copy constructor

Parameters

copy	TimeArr to be copied

References delay_precision, and time_arr_map.

13.72.3 Member Function Documentation

13.72.3.1 at() TimeArrCIt TimeArr::at (const int *i*) const

Returns a const iterator to the Pressure value at i-th position

Parameters

i	integer should be between 0 and size()

Returns

const iterator to end() if position *i* is not found

References time_arr_map.

13.72.3.2 begin() `TimeArrCIt woss::TimeArr::begin ( ) const [inline]`

Returns a const iterator to the beginning of the time arrival map

Returns

const iterator

References time_arr_map.

Referenced by woss::Pressure::Pressure(), and woss::ResTimeArrBinDb::writeMap().

13.72.3.3 checkPressureAttenuation() `bool TimeArr::checkPressureAttenuation ( double distance, double frequency ) [virtual]`

Check the Pressure value of each delay. If amplitude is >= 1, it replaces with a new Pressure with same phase and new amplitude given by the Thorp absorption process at given frequency and along given distance

Parameters

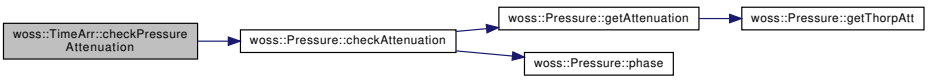
distance	given distance [m]
frequency	given frequency [hz]

Returns

true if correction was applied, *false* otherwise

References woss::Pressure::checkAttenuation(), and time_arr_map.

Here is the call graph for this function:



13.72.3.4 clear() `void woss::TimeArr::clear ( ) [inline]`

Erase all values of Pressure

References time_arr_map.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), and woss::WossManagerResDbMT::getWossTimeArr().

13.72.3.5 clone() `virtual TimeArr * woss::TimeArr::clone ( ) const [inline], [virtual]`

TimeArr virtual factory method

Returns

a heap-created copy of **this** instance

References TimeArr().

Referenced by woss::DefHandler::operator=().

Here is the call graph for this function:

13.72.3.6 coherentSumSample() `TimeArr * TimeArr::coherentSumSample ( double time_delay ) [virtual]`

Sample the TimeArr with sample-time *delay*. Pressure of the new TimeArr are the coherent sum of previous object with equal sampled delay.

Parameters

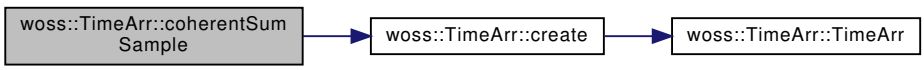
delay	delay value [s]

Returns

a new TimeArr object

References create(), delay_precision, and time_arr_map.

Here is the call graph for this function:



13.72.3.7 create() [1/4] `virtual TimeArr * woss::TimeArr::create ( const Pressure & pressure, double delay = TIMEARR_PRESSURE_CONVERSION_DELAY, long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION ) const [inline], [virtual]`

TimeArr virtual factory method

Parameters

pressure	Pressure value
delay	delay value [s]
custom_delay_prec	delay precision [s]

Returns

a heap-created TimeArr object

References TimeArr().

Here is the call graph for this function:

13.72.3.8 create() [2/4] virtual TimeArr * woss::TimeArr::create (const TimeArr & *copy*) const [inline], [virtual]

TimeArr virtual factory method

Parameters

copy	TimeArr to be copied

Returns

a heap-created TimeArr object

References TimeArr().

Here is the call graph for this function:

13.72.3.9 create() [3/4] virtual TimeArr * woss::TimeArr::create (long double *custom_delay_prec* = *TIMEARR_CUSTOM_DELAY_PRECISION*) const [inline], [virtual]

TimeArr virtual factory method

Parameters

custom_delay_prec	delay precision [s]

Returns

a heap-created TimeArr object

References TimeArr().

Referenced by coherentSumSample(), crop(), woss::ResTimeArrTxtDb::getValue(), and incoherentSumSample().

Here is the call graph for this function:

13.72.3.10 create() [4/4]

```
virtual TimeArr * woss::TimeArr::create (
        TimeArrMap & map,
        long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION ) const  [inline],
[virtual]
```

TimeArr virtual factory method

Parameters

map	custom time arrival map
custom_delay_prec	delay precision [s]

Returns

a heap-created TimeArr object

References TimeArr().

Here is the call graph for this function:

13.72.3.11 createArray()

```
virtual TimeArr * woss::TimeArr::createArray (
        unsigned int array_size ) const  [inline], [virtual]
```

TimeArr virtual factory method

Parameters

array_size	size of array

Returns

a heap-created array of size array_size

13.72.3.12 createImpulse() `TimeArrMap & woss::TimeArr::createImpulse ( ) [inline], [static]`

Creates an impulsive instance

Returns

a new impulsive instance (e.g. delay 0.0 = (+1.0, 0.0))

References time_arr_map.

Referenced by woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), and woss::WossManager::getWossTimeArr().

13.72.3.13 createNotValid() `TimeArrMap & woss::TimeArr::createNotValid ( ) [inline], [static]`

Creates an instance not valid

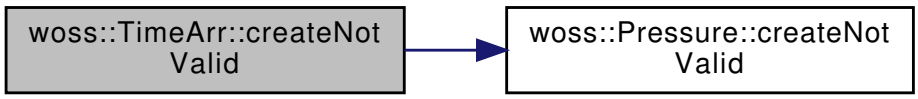
Returns

a new instance not valid (e.g. delay 0.0 = (+inf, +inf))

References woss::Pressure::createNotValid(), and time_arr_map.

Referenced by woss::WossManagerResDb::dbGetTimeArr(), woss::WossDbManager::getTimeArr(), woss::ResTimeArrTxtDb::getVal
woss::WossManager::getWossTimeArr(), woss::ArrAscResReader::readTimeArr(), and woss::ArrBinResReader::readTimeArr().

Here is the call graph for this function:



13.72.3.14 crop() `TimeArr * TimeArr::crop ( double time_start, double time_end ) [virtual]`

Crops the TimeArr between given time values and returns a new heap-allocated object. The new object will have time values in [time_start, time_end)

Parameters

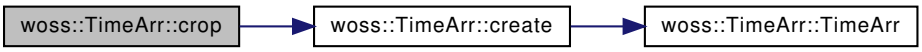
time_start	start time value [s]
time_end	end time value [s]

Returns

a new TimeArr object

References create(), delay_precision, and time_arr_map.

Here is the call graph for this function:



13.72.3.15 empty() `bool woss::TimeArr::empty ( ) const [inline]`

Checks if the instance has stored values

Returns

true if condition applies, *false* otherwise

References time_arr_map.

13.72.3.16 end() `TimeArrCIt woss::TimeArr::end ( ) const [inline]`

Returns a const iterator to the end of the time arrival map

Returns

const iterator

References time_arr_map.

Referenced by woss::Pressure::Pressure(), and woss::ResTimeArrBinDb::writeMap().

13.72.3.17 eraseValue() `TimeArr & woss::TimeArr::eraseValue ( double delay ) [inline]`

Erase the Pressure value with key == *delay* parameter

Parameters

delay	delay value [s]

Returns

reference to ∗**this**

References time_arr_map.

13.72.3.18 findValue() `TimeArrCIt woss::TimeArr::findValue ( double delay ) const [inline]`

Returns a const iterator to the Pressure with key == *delay* parameter

Parameters

delay	delay value [s]

Returns

const iterator to end() if *delay* is not found

References time_arr_map.

13.72.3.19 getDelayPrecision() `long double woss::TimeArr::getDelayPrecision ( ) const [inline]`

Returns the delay precision

Returns

delay precision [s]

References delay_precision.

13.72.3.20 getMaxDelayValue() `double woss::TimeArr::getMaxDelayValue ( ) const [inline]`

Returns the maximum delay value

Returns

maximum delay [s]

References time_arr_map.

13.72.3.21 getMinDelayValue() `double woss::TimeArr::getMinDelayValue ( ) const [inline]`

Returns the maximum delay value

Returns

maximum delay [s]

References time_arr_map.

13.72.3.22 incoherentSumSample() `TimeArr * TimeArr::incoherentSumSample ( double time_delay ) [virtual]`

Sample the TimeArr with sample-time *delay*. Pressure of the new TimeArr are the incoherent sum of previous object with equal sampled delay.

Parameters

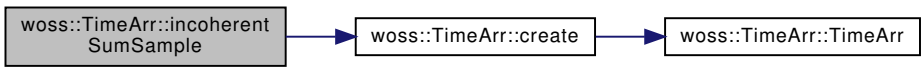
delay	delay value [s]

Returns

a new TimeArr object

References create(), delay_precision, and time_arr_map.

Here is the call graph for this function:



13.72.3.23 insertValue() `TimeArr & woss::TimeArr::insertValue ( double delay, const Pressure & pressure ) [inline]`

Inserts and replace a Pressure value at given delay

Parameters

delay	delay value [s]
pressure	Pressure value

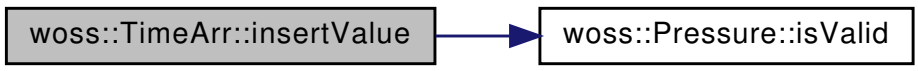
Returns

reference to ∗**this**

References woss::Pressure::isValid(), and time_arr_map.

Referenced by woss::ResTimeArrBinDb::importMap().

Here is the call graph for this function:



13.72.3.24 isConvertedFromPressure() `virtual bool woss::TimeArr::isConvertedFromPressure ( ) const [inline], [virtual]`

Checks if the TimeArr was constructed from a Pressure value, therefore not carrying a valid delay information

Returns

true if it has at least one value, *false* otherwise

References time_arr_map.

13.72.3.25 isValid() `bool TimeArr::isValid ( ) const [virtual]`

Checks the validity of TimeArr

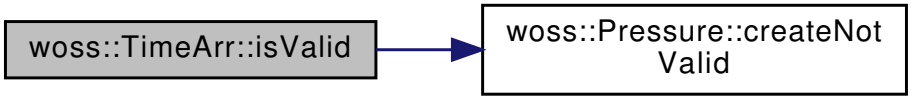
Returns

true if it has at least one value, *false* otherwise

References woss::Pressure::createNotValid(), and time_arr_map.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), and woss::Pressure::Pressure().

Here is the call graph for this function:



13.72.3.26 lowerBoundTxLoss() `TimeArrCIt TimeArr::lowerBoundTxLoss ( double threshold_db ) const`

Returns a const iterator to the Pressure that has txLoss >= threshold_db

Parameters

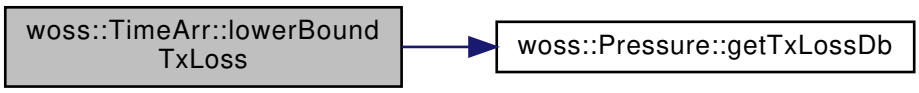
i	threshold in db re upa

Returns

const iterator to end() if position *i* is not found

References woss::Pressure::getTxLossDb(), and time_arr_map.

Here is the call graph for this function:



13.72.3.27 operator std::complex< double >() `TimeArr::operator std::complex< double > ( ) const [virtual]`

complex<double> operator for implicit casting

Returns

the coherent sum

13.72.3.28 operator=() `TimeArr & TimeArr::operator= ( const TimeArr & copy )`

Assignment operator

Parameters

copy	const reference to a TimeArr object to be copied

Returns

TimeArr reference to *this*

References delay_precision, and time_arr_map.

13.72.3.29 rbegin() `TimeArrCRIt woss::TimeArr::rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse beginning of the time arrival map

Returns

const iterator

References time_arr_map.

13.72.3.30 rend() `TimeArrCRIt woss::TimeArr::rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the time arrival map

Returns

const iterator

References time_arr_map.

13.72.3.31 setDebug() `static void woss::TimeArr::setDebug ( bool flag ) [inline], [static]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.72.3.32 setDelayPrecision() `woss::TimeArr & TimeArr::setDelayPrecision ( long double precision )`

Sets the delay precision for all PDouble delay values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

precision	delay precision [m]

Returns

reference to ∗**this**

References time_arr_map.

13.72.3.33 size() `int woss::TimeArr::size ( ) const [inline]`

Returns the number of Pressure stored

Returns

number of Pressure values stored

References time_arr_map.

Referenced by woss::ResTimeArrBinDb::writeMap().

13.72.3.34 sumValue() `void woss::TimeArr::sumValue ( double delay, const Pressure & pressure ) [inline]`

Inserts and (complex) sums a Pressure value at given delay

Parameters

delay	delay value [s]
pressure	Pressure value

References woss::Pressure::isValid(), and time_arr_map.

Referenced by woss::ArrBinResReader::getArrBinFile().

Here is the call graph for this function:

13.72.4 Friends And Related Function Documentation

13.72.4.1 operator"!= `bool operator!= ( const TimeArr & left, const TimeArr & right ) [friend]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

13.72.4.2 operator* [1/2]

```
const TimeArr operator* (
    const double left,
    const TimeArr & right )  [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.3 operator* [2/2]

```
const TimeArr operator* (
    const TimeArr & left,
    const double right )  [friend]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.4 operator*= TimeArr & operator*= (TimeArr & *left*, double *right*) [friend]

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.5 operator+ [1/3] const TimeArr operator+ (const double *left*, const TimeArr & *right*) [friend]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.6 operator+ [2/3] const TimeArr operator+ (const TimeArr & *left*, const double *right*) [friend]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.7 operator+ [3/3]

```
const TimeArr operator+ (
    const TimeArr & left,
    const TimeArr & right )  [friend]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.8 operator+= [1/2]

```
TimeArr & operator+= (
    TimeArr & left,
    const TimeArr & right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.9 operator+= [2/2]

```
TimeArr & operator+= (
    TimeArr & left,
    double right )  [friend]
```

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.10 operator- [1/3]

```
const TimeArr operator- (
    const double left,
    const TimeArr & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.11 operator- [2/3]

```
const TimeArr operator- (
    const TimeArr & left,
    const double right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.12 operator- [3/3]

```
const TimeArr operator- (
    const TimeArr & left,
    const TimeArr & right )  [friend]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.13 operator-= [1/2]

```
TimeArr & operator-= (
    TimeArr & left,
    const TimeArr & right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.14 operator-= [2/2]

```
TimeArr & operator-= (
    TimeArr & left,
    double right )  [friend]
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.15 operator/ [1/2] const TimeArr operator/ (const double *left*, const TimeArr & *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.16 operator/ [2/2] const TimeArr operator/ (const TimeArr & *left*, const double *right*) [friend]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

13.72.4.17 operator/= TimeArr & operator/= (TimeArr & *left*, double *right*) [friend]

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

13.72.4.18 operator<< `std::ostream & operator<< (`

```
std::ostream & os,
const TimeArr & instance )  [friend]
```

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Pressure reference

Returns

os reference after the operation

13.72.4.19 operator== `bool operator== (`

```
const TimeArr & left,
const TimeArr & right )  [friend]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.72.5 Member Data Documentation

13.72.5.1 debug `bool TimeArr::debug = false  [static], [protected]`

Debug flag

Referenced by setDebug().

13.72.5.2 delay_precision `long double woss::TimeArr::delay_precision [protected]`

Stores the precision of all PDouble delay instances [s]

Referenced by coherentSumSample(), crop(), getDelayPrecision(), incoherentSumSample(), operator=(), and TimeArr().

13.72.5.3 time_arr_map `TimeArrMap woss::TimeArr::time_arr_map [protected]`

Pressure values map

Referenced by at(), begin(), checkPressureAttenuation(), clear(), coherentSumSample(), createImpulse(), createNotValid(), crop(), empty(), end(), eraseValue(), findValue(), getMaxDelayValue(), getMinDelayValue(), incoherentSumSample(), insertValue(), isConvertedFromPressure(), isValid(), lowerBoundTxLoss(), woss::operator*=(), woss::operator+=(), woss::operator-=(), woss::operator/=(), operator=(), rbegin(), rend(), setDelayPrecision(), size(), sumValue(), and TimeArr().

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

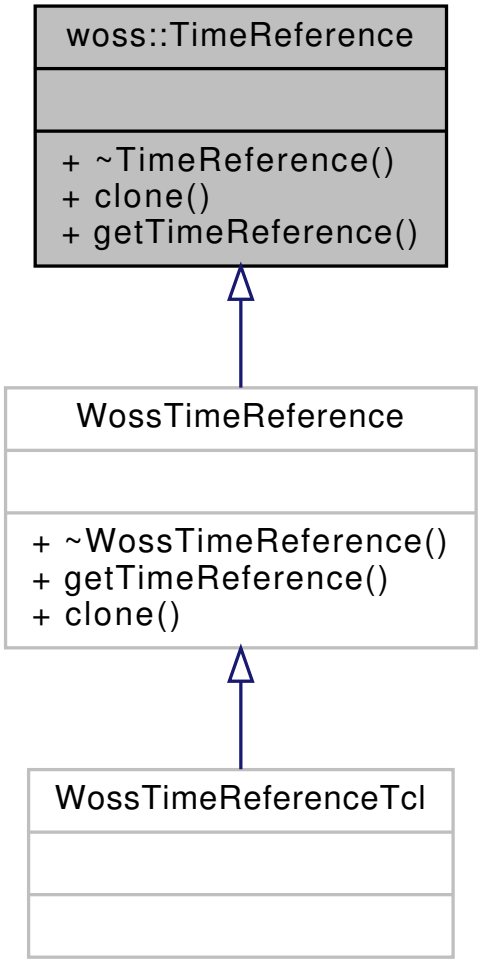
- woss/woss_def/time-arrival-definitions.h
- woss/woss_def/time-arrival-definitions.cpp

13.73 woss::TimeReference Class Reference

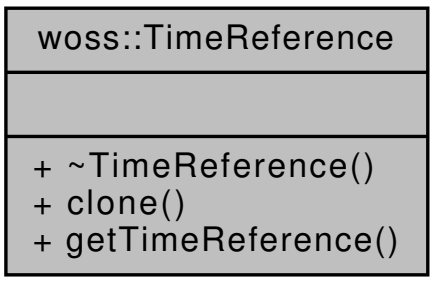
Class for simulation time reference purposes.

`#include <time-definitions.h>`

Inheritance diagram for woss::TimeReference:



Collaboration diagram for woss::TimeReference:



Public Member Functions

- virtual TimeReference * clone ()=0
- virtual double getTimeReference () const =0

13.73.1 Detailed Description

Class for simulation time reference purposes.

Class for simulation time reference purposes

13.73.2 Member Function Documentation

13.73.2.1 clone() `virtual TimeReference * woss::TimeReference::clone ( ) [pure virtual]`

woss::TimeReference virtual factory method

Returns

a heap-allocated copy of **this** instance

Implemented in WossTimeReference.

Referenced by woss::DefHandler::operator=().

13.73.2.2 getTimeReference() `virtual double woss::TimeReference::getTimeReference ( ) const [pure virtual]`

Retuns simulation time

Returns

value in seconds

Implemented in WossTimeReference.

Referenced by woss::DefHandler::getTimeReference().

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

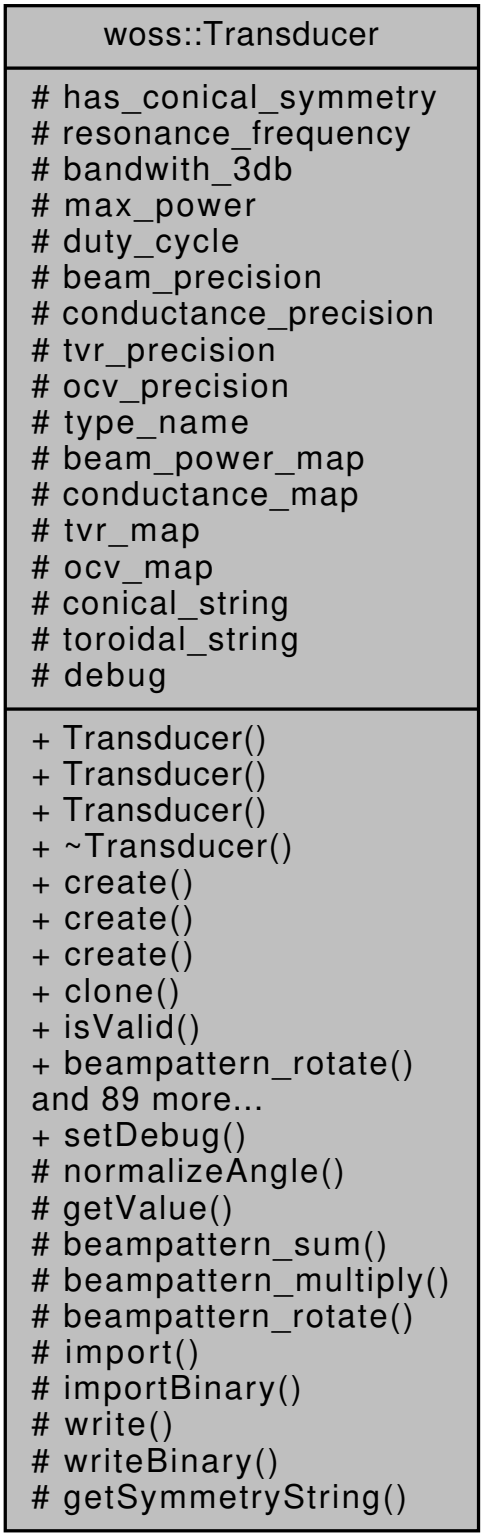
- woss/woss_def/time-definitions.h

13.74 woss::Transducer Class Reference

Transducer class.

`#include <transducer-definitions.h>`

Collaboration diagram for woss::Transducer:



Public Member Functions

- Transducer (long double beam_precision=BEAM_PATTERN_CUSTOM_BEAM_PRECISION, long double conduct_precison=CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION, long double tvr_precision=TVR↩ _CUSTOM_FREQUENCY_PRECISION, long double ocv_precision=OCV_CUSTOM_FREQUENCY_↩ PRECISION)

- Transducer (const Transducer &copy)
- Transducer (BeamPowerMap &beam_map, ConductanceMap &conductance_map, TVRMap &tvr_map, OCVMap &ocv_map)
- virtual Transducer * create (long double beam_precision=BEAM_PATTERN_CUSTOM_BEAM_PRECISION, long double conduct_precison=CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION, long double tvr_precision=TVR_CUSTOM_FREQUENCY_PRECISION, long double ocv_precision=OCV_CUSTOM↩ _FREQUENCY_PRECISION) const
- virtual Transducer * create (BeamPowerMap &beam_map, ConductanceMap &conductance_map, TVRMap &tvr_map, OCVMap &ocv_map) const
- virtual Transducer * create (const Transducer &copy) const
- virtual Transducer * clone () const
- virtual bool isValid () const
- Transducer & beampattern_rotate (double angle)
- Transducer & beampattern_sum (double value)
- Transducer & beampattern_multiply (double value)
- virtual double getSPL (double frequency, double power) const
- double getMaxSPL (double frequency) const
- virtual double getPowerFromSPL (double frequency, double spl) const
- bool beampattern_insert (double angle, double power)
- Transducer & beampattern_replace (double angle, double power)
- BPMCIter beampattern_find (double angle) const
- Transducer & beampattern_erase (double angle)
- int beampattern_size () const
- bool beampattern_empty () const
- Transducer & beampattern_clear ()
- BPMCIter beampattern_begin () const
- BPMCIter beampattern_end () const
- BPMCRIter beampattern_rbegin () const
- BPMCRIter beampattern_rend () const
- BPMCIter beampattern_lower_bound (double angle) const
- BPMCIter beampattern_upper_bound (double angle) const
- bool conductance_insert (double frequency, double conductance)
- bool conductance_insert (double frequency, const ::std::complex< double > &impedance)
- Transducer & conductance_replace (double frequency, double conductance)
- Transducer & conductance_replace (double frequency, const ::std::complex< double > &impedance)
- CMCIter conductance_find (double frequency) const
- Transducer & conductance_erase (double frequency)
- int conductance_size () const
- bool conductance_empty () const
- Transducer & conductance_clear ()
- CMCIter conductance_begin () const
- CMCIter conductance_end () const
- CMCRIter conductance_rbegin () const
- CMCRIter conductance_rend () const
- CMCIter conductance_lower_bound (double frequency) const
- CMCIter conductance_upper_bound (double frequency) const
- bool tvr_insert (double frequency, double tvr)
- Transducer & tvr_replace (double frequency, double tvr)
- TVRMCIter tvr_find (double frequency) const
- Transducer & tvr_erase (double frequency)
- int tvr_size () const
- bool tvr_empty () const
- Transducer & tvr_clear ()
- TVRMCIter tvr_begin () const
- TVRMCIter tvr_end () const

- TVRMCRIter tvr_rbegin () const
- TVRMCRIter tvr_rend () const
- TVRMCIter tvr_lower_bound (double frequency) const
- TVRMCIter tvr_upper_bound (double frequency) const
- bool ocv_insert (double frequency, double ocv)
- Transducer & ocv_replace (double frequency, double ocv)
- OCVMCIter ocv_find (double frequency) const
- Transducer & ocv_erase (double frequency)
- int ocv_size () const
- bool ocv_empty () const
- Transducer & ocv_clear ()
- OCVMCIter ocv_begin () const
- OCVMCIter ocv_end () const
- OCVMCRIter ocv_rbegin () const
- OCVMCRIter ocv_rend () const
- OCVMCIter ocv_lower_bound (double frequency) const
- OCVMCIter ocv_upper_bound (double frequency) const
- Transducer & clearAll ()
- virtual bool import (::std::istream &stream_in)
- virtual bool importBinary (::std::fstream &stream_in)
- bool writeVertBeamPattern (::std::ostream &stream_out, const CoordZ &tx, const CoordZ &rx, double init_↩ bearing, double vert_rot=0, double horiz_rot=0, double mult_costant=1, double add_costant=0) const
- bool writeSPL (::std::ostream &stream_out, double frequency_step, double power) const
- virtual bool write (::std::ostream &stream_out) const
- virtual bool writeBinary (::std::fstream &file_out) const
- Transducer & **setMaxPower** (double power)
- Transducer & **setDutyCycle** (double cycle)
- Transducer & **setResonanceFrequency** (double frequency)
- Transducer & **setBandwith3dB** (double frequency)
- Transducer & **setTypeName** (const ::std::string &name)
- virtual Transducer & setBeamPrecision (long double prec)
- virtual Transducer & setTVRPrecision (long double prec)
- virtual Transducer & setOCVPrecision (long double prec)
- virtual Transducer & setConductancePrecision (long double prec)
- bool **hasToroidalSymmetry** () const
- bool **hasConicalSymmetry** () const
- double getMaxPower () const
- double getDutyCycle () const
- double getResonanceFrequency () const
- double getBandwith3dB () const
- ::std::string getTypeName () const
- long double getBeamPrecision () const
- long double getTVRPrecision () const
- long double getOCVPrecision () const
- long double getConductancePrecision () const
- Transducer & operator= (const Transducer &x)
- friend::std::ostream & operator<< (::std::ostream &os, const Transducer &instance)
- friend::std::ostream & operator>> (::std::istream &is, const Transducer &instance)

Static Public Member Functions

- static void setDebug (bool flag)

Protected Types

- typedef ::std::map< PDouble, double > BeamPowerMap
- typedef BeamPowerMap::iterator **BPMIter**
- typedef BeamPowerMap::reverse_iterator **BPMRIter**
- typedef BeamPowerMap::const_iterator **BPMCIter**
- typedef BeamPowerMap::const_reverse_iterator **BPMCRIter**
- typedef ::std::map< PDouble, double > ConductanceMap
- typedef ConductanceMap::iterator **CMIter**
- typedef ConductanceMap::reverse_iterator **CMRIter**
- typedef ConductanceMap::const_iterator **CMCIter**
- typedef ConductanceMap::const_reverse_iterator **CMCRIter**
- typedef ::std::map< PDouble, double > TVRMap
- typedef TVRMap::iterator **TVRMIter**
- typedef TVRMap::reverse_iterator **TVRMRIter**
- typedef TVRMap::const_iterator **TVRMCIter**
- typedef TVRMap::const_reverse_iterator **TVRMCRIter**
- typedef ::std::map< PDouble, double > OCVMap
- typedef OCVMap::iterator **OCVMIter**
- typedef OCVMap::reverse_iterator **OCVMRIter**
- typedef OCVMap::const_iterator **OCVMCIter**
- typedef OCVMap::const_reverse_iterator **OCVMCRIter**

Protected Member Functions

- virtual double normalizeAngle (double angle) const
- virtual double getValue (double frequency, const ::std::map< PDouble, double > &map, long double precision, bool use_linear=false, double costant=20.0) const
- virtual void beampattern_sum (double value, BeamPowerMap &map)
- virtual void beampattern_multiply (double value, BeamPowerMap &map)
- virtual void beampattern_rotate (double angle, BeamPowerMap &map)
- virtual bool import (::std::istream &stream_in, ::std::map< PDouble, double > &map, long double precision, bool is_angle=false)
- virtual bool importBinary (::std::fstream &file_in, ::std::map< PDouble, double > &map, long double precision, bool is_angle=false)
- virtual bool write (::std::ostream &stream_out, const ::std::map< PDouble, double > &map) const
- virtual bool writeBinary (::std::fstream &file_out, const ::std::map< PDouble, double > &map) const
- virtual const ::std::string & **getSymmetryString** () const

Protected Attributes

- bool has_conical_symmetry
- double resonance_frequency
- double bandwith_3db
- double max_power
- double duty_cycle
- long double beam_precision
- long double conductance_precision
- long double tvr_precision
- long double ocv_precision
- ::std::string type_name
- BeamPowerMap beam_power_map
- ConductanceMap conductance_map
- TVRMap tvr_map
- OCVMap ocv_map

Static Protected Attributes

- static const ::std::string **conical_string** = "CONICAL"
- static const ::std::string **toroidal_string** = "TOROIDAL"
- static bool debug = false

Friends

- bool operator== (const Transducer &left, const Transducer &right)
- bool operator!= (const Transducer &left, const Transducer &right)

13.74.1 Detailed Description

Transducer class.

woss::Transducer

13.74.2 Member Typedef Documentation

13.74.2.1 BeamPowerMap

```
typedef ::std::map< PDouble, double > woss::Transducer::BeamPowerMap [protected]
```

Map that links a angle with its precision to a signed power gain in decibel [db re uPa @ 1m]

13.74.2.2 ConductanceMap

```
typedef ::std::map< PDouble, double > woss::Transducer::ConductanceMap [protected]
```

Map that links a frequency with its precision to a conductance value [uS]

13.74.2.3 OCVMap

```
typedef ::std::map< PDouble, double > woss::Transducer::OCVMap [protected]
```

Map that links a frequency with its precision to an OCV value [db re 1V/uPa]

13.74.2.4 TVRMap

```
typedef ::std::map< PDouble, double > woss::Transducer::TVRMap [protected]
```

Map that links a frequency with its precision to a TVR value [db re uPa/V @ 1m]

13.74.3 Constructor & Destructor Documentation

13.74.3.1 Transducer() [1/3]

```
Transducer::Transducer (
        long double beam_precision = BEAM_PATTERN_CUSTOM_BEAM_PRECISION,
        long double conduct_precison = CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION,
        long double tvr_precision = TVR_CUSTOM_FREQUENCY_PRECISION,
        long double ocv_precision = OCV_CUSTOM_FREQUENCY_PRECISION )
```

Transducer default constructor. The object created is not valid

Parameters

beam_precision	precision of woss::PDouble objects representing angles
conduct_precison	precision of woss::PDouble objects representing frequency
tvr_precision	precision of woss::PDouble objects representing frequency
ocv_precision	precision of woss::PDouble objects representing frequency

Referenced by clone(), and create().

13.74.3.2 Transducer() [2/3] Transducer::Transducer (const Transducer & *copy*)

Transducer copy constructor

Parameters

copy	const reference a Transducer

13.74.3.3 Transducer() [3/3] Transducer::Transducer (BeamPowerMap & *beam_map*, ConductanceMap & *conductance_map*, TVRMap & *tvr_map*, OCVMap & *ocv_map*)

Transducer constructor.

Parameters

beam_map	map linking a woss::PDouble angle [dec degrees] (with precision beam_precision) to a signed power gain [decibel]
conductance_map	map linking a woss::PDouble frequency [hz] (with precision conductance_precision) to conductance value
tvr_map	map linking a woss::PDouble frequency [hz] (with precision tvr_precision) to a TVR value [db re 1 uPa/V @ 1m]
ocv_map	map linking a woss::PDouble frequency [hz] (with precision ocv_precision) to a OCV value [db re 1V/uPa]

References beam_power_map, beam_precision, conductance_map, conductance_precision, ocv_map, ocv_precision, tvr_map, and tvr_precision.

13.74.4 Member Function Documentation

13.74.4.1 beampattern_begin() `Transducer::BPMCIter woss::Transducer::beampattern_begin ( ) const [inline]`

Returns a const iterator to the beginning of the beam pattern map

Returns

const iterator

References beam_power_map.

13.74.4.2 beampattern_clear() `Transducer & woss::Transducer::beampattern_clear ( ) [inline]`

Clears all values

Returns

reference to ∗**this**

References beam_power_map.

13.74.4.3 beampattern_empty() `bool woss::Transducer::beampattern_empty ( ) const [inline]`

Checks if the instance has stored values

Returns

true if condition applies, *false* otherwise

References beam_power_map.

13.74.4.4 beampattern_end() `Transducer::BPMCIter woss::Transducer::beampattern_end ( ) const [inline]`

Returns a const iterator to the end of the beam pattern map

Returns

const iterator

References beam_power_map.

13.74.4.5 beampattern_erase() `Transducer & woss::Transducer::beampattern_erase ( double angle ) [inline]`

Erase the power gain with key == of *angle* parameter

Parameters

angle	const reference to a double angle value

Returns

reference to ∗**this**

References beam_power_map.

13.74.4.6 beampattern_find()

```
Transducer::BPMCIter woss::Transducer::beampattern_find (
    double angle ) const  [inline]
```

Returns a const iterator to the signed power gain with key == of *angle* parameter

Parameters

angle	const reference to a double angle value

Returns

const iterator to end() if *angle* is not found

References beam_power_map.

13.74.4.7 beampattern_insert()

```
bool woss::Transducer::beampattern_insert (
    double angle,
    double power )  [inline]
```

Inserts and doesn't replace a signed power gain at given angle

Parameters

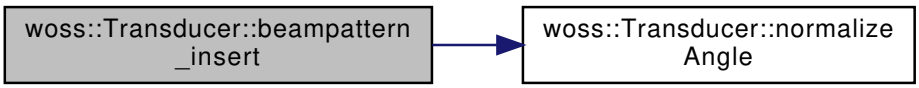
angle	angle value [dec degrees]. The corresponding PDouble will take SSP::beam_precision as precision
power	signed power gain [db re uPa]

Returns

true if inserted, *false* otherwise

References beam_power_map, beam_precision, and normalizeAngle().

Here is the call graph for this function:



13.74.4.8 beampattern_lower_bound()

```
Transducer::BPMCIter woss::Transducer::beampattern_lower↩
_bound (
            double angle ) const  [inline]
```

Returns a const iterator to the signed power gain with key >= of *angle* parameter

Parameters

angle	angle [dec degrees]

Returns

const iterator to end() if *angle* is not found

References beam_power_map, and beam_precision.

13.74.4.9 beampattern_multiply() [1/2]

```
Transducer & woss::Transducer::beampattern_multiply (
            double value )  [inline]
```

Multiplies the beam pattern by given value

Parameters

value	value

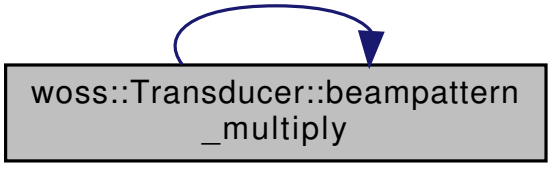
Returns

reference to ∗**this**

References beam_power_map, and beampattern_multiply().

Referenced by beampattern_multiply(), and writeVertBeamPattern().

Here is the call graph for this function:



13.74.4.10 beampattern_multiply() [2/2]

```
void Transducer::beampattern_multiply (
    double value,
    BeamPowerMap & map )  [protected], [virtual]
```

The current beam pattern is multiplied by given value

Parameters

value	value

13.74.4.11 beampattern_rbegin()

```
Transducer::BPMCRIter woss::Transducer::beampattern_rbegin ( )
const  [inline]
```

Returns a const reverse iterator to the reverse beginning of the beam pattern map

Returns

const reverse iterator

References beam_power_map.

13.74.4.12 beampattern_rend()

```
Transducer::BPMCRIter woss::Transducer::beampattern_rend ( )
const  [inline]
```

Returns a const reverse iterator to the reverse end of the beam pattern map

Returns

const reverse iterator

References beam_power_map.

13.74.4.13 beampattern_replace()

```
Transducer & woss::Transducer::beampattern_replace (
    double angle,
    double power )  [inline]
```

Replaces a signed power gain at given angle

Parameters

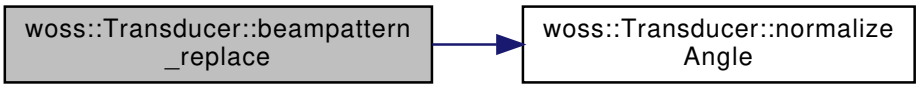
angle	angle value [dec degrees]. The corresponding PDouble will take SSP::beam_precision as precision
power	signed power gain [db re uPa]

Returns

reference to ∗**this**

References beam_power_map, beam_precision, and normalizeAngle().

Here is the call graph for this function:



13.74.4.14 beampattern_rotate() [1/2]

```
Transducer & woss::Transducer::beampattern_rotate (
    double angle )  [inline]
```

Rotate the beam pattern

Parameters

angle	angle [dec degrees]

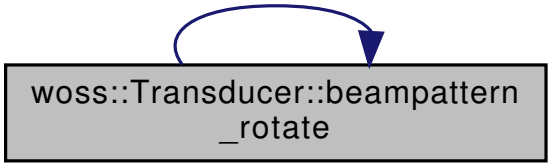
Returns

reference to ∗**this**

References beam_power_map, and beampattern_rotate().

Referenced by beampattern_rotate().

Here is the call graph for this function:



13.74.4.15 beampattern_rotate() [2/2]

```
void Transducer::beampattern_rotate (
    double angle,
    BeamPowerMap & map )  [protected], [virtual]
```

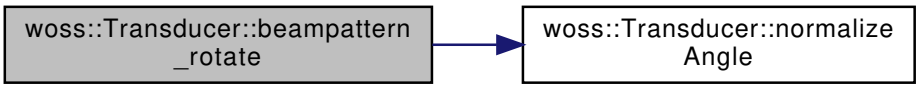
Rotates the current beam pattern of a given angle

Parameters

value	angle [decimal degrees]

References beam_precision, and normalizeAngle().

Here is the call graph for this function:



13.74.4.16 beampattern_size() `int woss::Transducer::beampattern_size ( ) const [inline]`

Returns the number of angles stored

Returns

number of angles stored

References beam_power_map.

13.74.4.17 beampattern_sum() `[1/2]` `Transducer & woss::Transducer::beampattern_sum ( double value ) [inline]`

Adds given value to the beam pattern

Parameters

value	sum costant

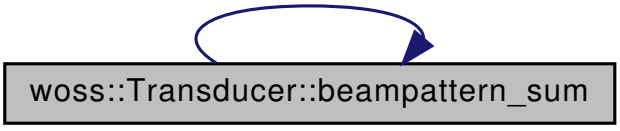
Returns

reference to ∗**this**

References beam_power_map, and beampattern_sum().

Referenced by beampattern_sum(), and writeVertBeamPattern().

Here is the call graph for this function:



13.74.4.18 beampattern_sum() [2/2]

```
void Transducer::beampattern_sum (
        double value,
        BeamPowerMap & map )  [protected], [virtual]
```

Adds a value to the current beam pattern

Parameters

value	value to be added

13.74.4.19 beampattern_upper_bound()

```
Transducer::BPMCIter woss::Transducer::beampattern_upper_bound (
        double angle ) const  [inline]
```

Returns a const iterator to the signed power gain with key $>$ of *angle* parameter

Parameters

angle	angle [dec degrees]

Returns

const iterator to end() if *angle* is not found

References beam_power_map, and beam_precision.

13.74.4.20 clearAll()

```
Transducer & woss::Transducer::clearAll ( )  [inline]
```

Clears all maps

Returns

reference to $*$**this**

References beam_power_map, conductance_map, ocv_map, and tvr_map.

13.74.4.21 clone() `Transducer * Transducer::clone ( ) const [virtual]`

Transducer virtual factory method

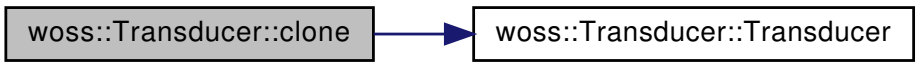
Returns

a heap-created copy of **this** instance

References Transducer().

Referenced by woss::DefHandler::operator=().

Here is the call graph for this function:



13.74.4.22 conductance_begin() `Transducer::CMCIter woss::Transducer::conductance_begin ( ) const [inline]`

Returns a const iterator to the beginning of the conductance map

Returns

const iterator

References conductance_map.

13.74.4.23 conductance_clear() `Transducer & woss::Transducer::conductance_clear ( ) [inline]`

Clears all values

Returns

reference to ∗**this**

References conductance_map.

13.74.4.24 conductance_empty() `bool woss::Transducer::conductance_empty ( ) const [inline]`

Checks if the instance has stored conductance values

Returns

true if condition applies, *false* otherwise

References conductance_map.

13.74.4.25 conductance_end() `Transducer::CMCIter woss::Transducer::conductance_end ( ) const [inline]`

Returns a const iterator to the end of the conductance map

Returns

const iterator

References conductance_map.

13.74.4.26 conductance_erase() `Transducer & woss::Transducer::conductance_erase ( double frequency ) [inline]`

Erase the power gain with key == of *angle* parameter

Parameters

frequency	frequency [hz]

Returns

reference to ∗**this**

References conductance_map.

13.74.4.27 conductance_find() `Transducer::CMCIter woss::Transducer::conductance_find ( double frequency ) const [inline]`

Returns a const iterator to the conductance with key == of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *frequency* is not found

References conductance_map.

13.74.4.28 conductance_insert() `[1/2] bool woss::Transducer::conductance_insert ( double frequency, const ::std::complex< double > & impedance ) [inline]`

Inserts and doesn't replace a complex impedance value at given frequency

Parameters

angle	frequency [hz]. The corresponding PDouble will take Transducer::conductance_precision as precision
conductance	complex impedance [uS + j uF]

Returns

true if inserted, *false* otherwise

References conductance_map, and conductance_precision.

13.74.4.29 conductance_insert() [2/2]

```
bool woss::Transducer::conductance_insert (
        double frequency,
        double conductance ) [inline]
```

Inserts and doesn't replace a conductance value at given frequency

Parameters

angle	frequency [hz]. The corresponding PDouble will take Transducer::conductance_precision as precision
conductance	conductance value [uS]

Returns

true if inserted, *false* otherwise

References conductance_map, and conductance_precision.

13.74.4.30 conductance_lower_bound()

```
Transducer::CMCIter woss::Transducer::conductance_lower_bound (
        double frequency ) const [inline]
```

Returns a const iterator to the conductance with key >= of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *angle* is not found

References conductance_map, and conductance_precision.

13.74.4.31 conductance_rbegin() `Transducer::CMCRIter woss::Transducer::conductance_rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse beginning of the conductance map

Returns

const reverse iterator

References conductance_map.

13.74.4.32 conductance_rend() `Transducer::CMCRIter woss::Transducer::conductance_rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the conductance map

Returns

const reverse iterator

References conductance_map.

13.74.4.33 conductance_replace() [1/2]

```
Transducer & woss::Transducer::conductance_replace (
        double frequency,
        const ::std::complex< double > & impedance ) [inline]
```

Replaces a conductance at given frequency

Parameters

angle	frequency [hz]. The corresponding PDouble will take Transducer::conductance_precision as precision
conductance	complex impedance [uS + j uF]

Returns

reference to ∗**this**

References conductance_map, and conductance_precision.

13.74.4.34 conductance_replace() [2/2]

```
Transducer & woss::Transducer::conductance_replace (
        double frequency,
        double conductance ) [inline]
```

Replaces a conductance at given frequency

Parameters

angle	frequency [hz]. The corresponding PDouble will take Transducer::conductance_precision as precision
conductance	conductance value [uS]

Returns

reference to ∗**this**

References conductance_map, and conductance_precision.

13.74.4.35 conductance_size() `int woss::Transducer::conductance_size ( ) const [inline]`

Returns the number of frequencies stored

Returns

number of frequencies stored

References conductance_map.

13.74.4.36 conductance_upper_bound() `Transducer::CMCIter woss::Transducer::conductance_↩ upper_bound (`

```
double frequency ) const [inline]
```

Returns a const iterator to the conductance with key > of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *angle* is not found

References conductance_map, and conductance_precision.

13.74.4.37 create() [1/3]

```
Transducer * Transducer::create (
    BeamPowerMap & beam_map,
    ConductanceMap & conductance_map,
    TVRMap & tvr_map,
    OCVMap & ocv_map ) const [virtual]
```

Transducer virtual factory method

Parameters

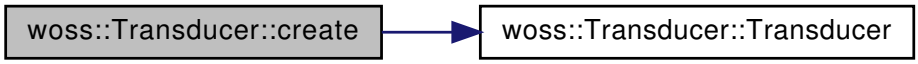
beam_map	map linking a woss::PDouble angle [dec degrees] (with precision beam_precision) to a signed power gain [decibel]
conductance_map	map linking a woss::PDouble frequency [hz] (with precision conductance_precision) to conductance value
tvr_map	map linking a woss::PDouble frequency [hz] (with precision tvr_precision) to a TVR value [db re 1 uPa/V @ 1m]
ocv_map	map linking a woss::PDouble frequency [hz] (with precision ocv_precision) to a OCV value [db re 1V/uPa]

Returns

a heap-created Transducer object

References Transducer().

Here is the call graph for this function:



13.74.4.38 create() [2/3] Transducer * Transducer::create (const Transducer & *copy*) const [virtual]

Transducer virtual factory method

Parameters

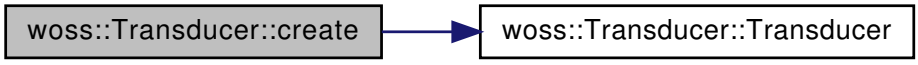
copy	Transducer to be copied

Returns

a heap-created Transducer object

References Transducer().

Here is the call graph for this function:



13.74.4.39 create() [3/3]

```
Transducer * Transducer::create (
        long double beam_precision = BEAM_PATTERN_CUSTOM_BEAM_PRECISION,
        long double conduct_precison = CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION,
        long double tvr_precision = TVR_CUSTOM_FREQUENCY_PRECISION,
        long double ocv_precision = OCV_CUSTOM_FREQUENCY_PRECISION ) const  [virtual]
```

Transducer virtual factory method

Parameters

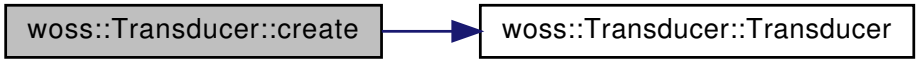
beam_map	map linking a woss::PDouble angle [dec degrees] (with precision beam_precision) to a signed power gain [decibel]
conductance_map	map linking a woss::PDouble frequency [hz] (with precision conductance_precision) to conductance value
tvr_map	map linking a woss::PDouble frequency [hz] (with precision tvr_precision) to a TVR value [db re 1 uPa/V @ 1m]
ocv_map	map linking a woss::PDouble frequency [hz] (with precision ocv_precision) to a OCV value [db re 1V/uPa]

Returns

a heap-created Transducer object

References Transducer().

Here is the call graph for this function:



13.74.4.40 getBandwith3dB() double woss::Transducer::getBandwith3dB ( ) const [inline]

Returns the bandwith at -3db around the resonance frequency

Returns

bandwith [hz]

References bandwith_3db.

13.74.4.41 getBeamPrecision() long double woss::Transducer::getBeamPrecision ( ) const [inline]

Returns the beam pattern beam precision

Returns

beam precision [decimal degrees]

References beam_precision.

13.74.4.42 getConductancePrecision() `long double woss::Transducer::getConductancePrecision ( ) const [inline]`

Returns the conductance frequency precision

Returns

precision [hz]

References conductance_precision.

13.74.4.43 getDutyCycle() `double woss::Transducer::getDutyCycle ( ) const [inline]`

Returns the recommended duty cycle

Returns

duty cycle [between 0 and 1]

References duty_cycle.

13.74.4.44 getMaxPower() `double woss::Transducer::getMaxPower ( ) const [inline]`

Returns max allowed input Power for this transducer

Returns

power [W]

References max_power.

13.74.4.45 getMaxSPL() `double woss::Transducer::getMaxSPL ( double frequency ) const [inline]`

Returns the max SPL (Sound Pressure Level) for given frequency

Parameters

frequency	frequency [hz]

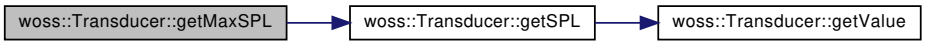
Returns

reference to ∗**this**

References getSPL(), and max_power.

Referenced by WossMPhyBpsk::getTxPower().

Here is the call graph for this function:



13.74.4.46 getOCVPrecision() `long double woss::Transducer::getOCVPrecision ( ) const [inline]`

Returns the OCV frequency precision

Returns

precision [hz]

References ocv_precision.

13.74.4.47 getPowerFromSPL() `double Transducer::getPowerFromSPL ( double frequency, double spl ) const [virtual]`

Returns the input power for given frequency and SPL

Parameters

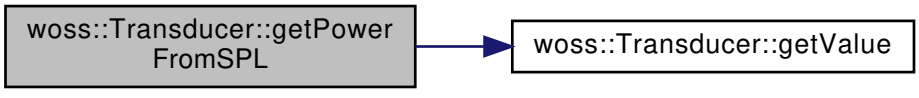
frequency	frequency [hz]
spl	[db re uPa]

Returns

power [W]

References conductance_map, conductance_precision, debug, getValue(), max_power, tvr_map, and tvr_precision.

Here is the call graph for this function:



13.74.4.48 getResonanceFrequency() `double woss::Transducer::getResonanceFrequency ( ) const [inline]`

Returns the resonance frequency of the transducer

Returns

frequency [hz]

References resonance_frequency.

13.74.4.49 getSPL() `double Transducer::getSPL ( double frequency, double power ) const [virtual]`

Returns the SPL (Sound Pressure Level) for given frequency and input power

Parameters

frequency	frequency [hz]
power	input power [W]

Returns

reference to ∗**this**

References conductance_map, conductance_precision, debug, getValue(), max_power, tvr_map, and tvr_precision.

Referenced by getMaxSPL(), and writeSPL().

Here is the call graph for this function:

13.74.4.50 getTVRPrecision() `long double woss::Transducer::getTVRPrecision ( ) const [inline]`

Returns the TVR frequency precision

Returns

precision [hz]

References tvr_precision.

13.74.4.51 getTypeName() `std::string woss::Transducer::getTypeName ( ) const [inline]`

Returns the transducer typename

Returns

name

References type_name.

13.74.4.52 getValue()

```
double Transducer::getValue (
    double frequency,
    const ::std::map< PDouble, double > & map,
    long double precision,
    bool use_linear = false,
    double costant = 20.0 ) const [protected], [virtual]
```

Returns a value for given frequency in the given map and with given precision.

Parameters

frequency	frequency [decimal degrees]
map	one of the transducer's map
map	precision

Returns

value found

References debug, and tvr_map.

Referenced by getPowerFromSPL(), and getSPL().

13.74.4.53 import() [1/2]

```
virtual bool woss::Transducer::import (
    ::std::istream & stream_in ) [virtual]
```

Imports values in from the given stream

Parameters

stream_in	const reference to an istream instance

Returns

true if method was successful, false otherwise

Referenced by woss::TransducerHandler::importValueAscii().

13.74.4.54 import() [2/2]

```
bool Transducer::import (
    ::std::istream & stream_in,
    ::std::map< PDouble, double > & map,
    long double precision,
    bool is_angle = false )  [protected], [virtual]
```

Imports values in the given map with given precision from the given stream

Parameters

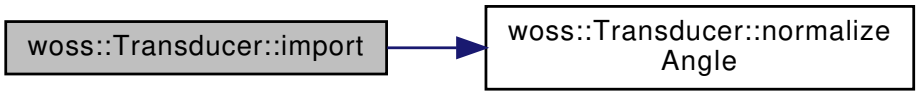
stream_in	const reference to an istream instance
map	any of the transducer's map
precision	map PDouble keys precision
is_angle	flag to signal if the input map is a beam pattern map

Returns

true if method was successful, false otherwise

References debug, and normalizeAngle().

Here is the call graph for this function:



13.74.4.55 importBinary() [1/2]

```
bool Transducer::importBinary (
    ::std::fstream & file_in,
    ::std::map< PDouble, double > & map,
    long double precision,
    bool is_angle = false )  [protected], [virtual]
```

Imports values in the given map with given precision from the given binary stream

Parameters

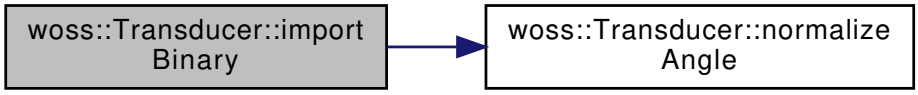
stream_in	const reference to an istream instance
map	any of the transducer's map
precision	map PDouble keys precision
is_angle	flag to signal if the input map is a beam pattern map

Returns

true if method was successful, false otherwise

References debug, and normalizeAngle().

Here is the call graph for this function:



13.74.4.56 importBinary() [2/2] virtual bool woss::Transducer::importBinary (::std::fstream & *stream_in*) [virtual]

Imports values in from the given binary stream

Parameters

stream_in	const reference to an istream instance

Returns

true if method was successful, false otherwise

Referenced by woss::TransducerHandler::importValueBinary().

13.74.4.57 isValid() bool Transducer::isValid ( ) const [virtual]

Checks the validity of the object

Returns

true if it has at least one value, *false* otherwise

References bandwith_3db, beam_power_map, conductance_map, duty_cycle, max_power, ocv_map, resonance_frequency, and tvr_map.

Referenced by WossMPhyBpsk::getTxPower(), and woss::BellhopWoss::writeRayOptions().

13.74.4.58 normalizeAngle() double Transducer::normalizeAngle (double *angle*) const [protected], [virtual]

Returns an angle in [-180.0 , 180.0]

Parameters

angle	angle [decimal degrees]

Returns

angle [decimal degrees]

Referenced by beampattern_insert(), beampattern_replace(), beampattern_rotate(), import(), and importBinary().

13.74.4.59 ocv_begin() `Transducer::CMCIter woss::Transducer::ocv_begin ( ) const [inline]`

Returns a const iterator to the beginning of the OCV map

Returns

const iterator

References ocv_map.

13.74.4.60 ocv_clear() `Transducer & woss::Transducer::ocv_clear ( ) [inline]`

Clears all OCV values

Returns

reference to ∗**this**

References ocv_map.

13.74.4.61 ocv_empty() `bool woss::Transducer::ocv_empty ( ) const [inline]`

Checks if the instance has any OCV stored values

Returns

true if condition applies, *false* otherwise

References ocv_map.

13.74.4.62 ocv_end() `Transducer::CMCIter woss::Transducer::ocv_end ( ) const [inline]`

Returns a const iterator to the end of the OCV map

Returns

const iterator

References ocv_map.

13.74.4.63 ocv_erase() `Transducer & woss::Transducer::ocv_erase ( double frequency ) [inline]`

Erase the OCV with key == of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

reference to ∗**this**

References ocv_map.

13.74.4.64 ocv_find() `Transducer::CMCIter woss::Transducer::ocv_find ( double frequency ) const [inline]`

Returns a const iterator to the OCV with key == of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *frequency* is not found

References ocv_map.

13.74.4.65 ocv_insert() `bool woss::Transducer::ocv_insert ( double frequency, double ocv ) [inline]`

Inserts and doesn't replace an OCV value at given frequency

Parameters

frequency	frequency value [hz]. The corresponding PDouble will take Transducer::ocv_precision as precision
ocv	OCV [db re 1V/uPa]

Returns

true if inserted, *false* otherwise

References ocv_map, and ocv_precision.

13.74.4.66 ocv_lower_bound() `Transducer::CMCIter woss::Transducer::ocv_lower_bound ( double frequency ) const [inline]`

Returns a const iterator to the OCV with key >= of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *angle* is not found

References ocv_map, and ocv_precision.

13.74.4.67 ocv_rbegin() `Transducer::CMCRIter woss::Transducer::ocv_rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse beginning of the OCV map

Returns

const reverse iterator

References ocv_map.

13.74.4.68 ocv_rend() `Transducer::CMCRIter woss::Transducer::ocv_rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the OCV map

Returns

const reverse iterator

References ocv_map.

13.74.4.69 ocv_replace() `Transducer & woss::Transducer::ocv_replace ( double frequency, double ocv ) [inline]`

Replaces an OCV value at given frequency

Parameters

frequency	frequency value [hz]. The corresponding PDouble will take Transducer::ocv_precision as precision
ocv	OCV [db re 1V/uPa]

Returns

reference to ∗**this**

References ocv_map, and ocv_precision.

13.74.4.70 ocv_size() `int woss::Transducer::ocv_size ( ) const [inline]`

Returns the number of OCV value stored

Returns

number of angles stored

References ocv_map.

13.74.4.71 ocv_upper_bound() `Transducer::CMCIter woss::Transducer::ocv_upper_bound ( double frequency ) const [inline]`

Returns a const iterator to the OCV with key > of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *angle* is not found

References ocv_map, and ocv_precision.

13.74.4.72 operator<<() `friend::std::ostream & woss::Transducer::operator<< ( ::std::ostream & os, const Transducer & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Transducer reference

Returns

os reference after the operation

13.74.4.73 operator=() `Transducer & Transducer::operator= ( const Transducer & x )`

Assignment operator

Parameters

copy	const reference to a Transducer object to be copied

Returns

Transducer reference to *this*

References bandwith_3db, beam_power_map, beam_precision, conductance_map, conductance_precision, duty_cycle, has_conical_symmetry, max_power, ocv_map, ocv_precision, resonance_frequency, tvr_map, and tvr_precision.

13.74.4.74 operator>>() `friend::std::ostream & woss::Transducer::operator>> ( ::std::istream & is, const Transducer & instance )`

operator

Parameters

os	left operand ostream reference
instance	right operand const Transducer reference

Returns

os reference after the operation

13.74.4.75 setBeamPrecision() `Transducer & Transducer::setBeamPrecision ( long double prec ) [virtual]`

Sets the beam pattern precision for all PDouble angle values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

prec	beam precision [decimal degrees]

References beam_power_map, and beam_precision.

13.74.4.76 setConductancePrecision() `Transducer & Transducer::setConductancePrecision ( long double prec ) [virtual]`

Sets the conductance precision for all PDouble frequency values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

prec	frequency [hz]

References conductance_map, and conductance_precision.

13.74.4.77 setDebug() `static void woss::Transducer::setDebug ( bool flag ) [inline], [static]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.74.4.78 setOCVPrecision() `Transducer & Transducer::setOCVPrecision ( long double prec ) [virtual]`

Sets the OCV precision for all PDouble frequency values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

prec	frequency [hz]

References beam_precision, and ocv_map.

13.74.4.79 setTVRPrecision() `Transducer & Transducer::setTVRPrecision ( long double prec ) [virtual]`

Sets the TVR precision for all PDouble frequency values. If the given precision is different from current value, the profile will be modified accordingly

Parameters

prec	frequency [hz]

References tvr_map, and tvr_precision.

13.74.4.80 tvr_begin() `Transducer::CMCIter woss::Transducer::tvr_begin ( ) const [inline]`

Returns a const iterator to the beginning of the tvr map

Returns

const iterator

References tvr_map.

13.74.4.81 tvr_clear() `Transducer & woss::Transducer::tvr_clear ( ) [inline]`

Clears all values

Returns

reference to ∗**this**

References tvr_map.

13.74.4.82 tvr_empty() `bool woss::Transducer::tvr_empty ( ) const [inline]`

Checks if the instance has any stored tvr values

Returns

true if condition applies, *false* otherwise

References tvr_map.

13.74.4.83 tvr_end() `Transducer::CMCIter woss::Transducer::tvr_end ( ) const [inline]`

Returns a const iterator to the end of the tvr map

Returns

const iterator

References tvr_map.

13.74.4.84 tvr_erase() `Transducer & woss::Transducer::tvr_erase ( double frequency ) [inline]`

Erase the tvr with key == to *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

reference to ∗**this**

References tvr_map.

13.74.4.85 tvr_find() `Transducer::CMCIter woss::Transducer::tvr_find ( double frequency ) const [inline]`

Returns a const iterator to the tvr with key == to *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *angle* is not found

References tvr_map.

13.74.4.86 tvr_insert() `bool woss::Transducer::tvr_insert ( double frequency, double tvr ) [inline]`

Inserts and doesn't replace a tvr at given frequency

Parameters

frequency	frequency value [hz]. The corresponding PDouble will take Transducer::tvr_precision as precision
tvr	tvr [db re uPa/V @ 1m]

Returns

true if inserted, *false* otherwise

References tvr_map, and tvr_precision.

13.74.4.87 tvr_lower_bound() `Transducer::CMCIter woss::Transducer::tvr_lower_bound ( double frequency ) const [inline]`

Returns a const iterator to the signed power gain with key >= of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *frequency* is not found

References tvr_map, and tvr_precision.

13.74.4.88 tvr_rbegin() `Transducer::CMCRIter woss::Transducer::tvr_rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse beginning of the tvr map

Returns

const reverse iterator

References tvr_map.

13.74.4.89 tvr_rend() `Transducer::CMCRIter woss::Transducer::tvr_rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the tvr map

Returns

const reverse iterator

References tvr_map.

13.74.4.90 tvr_replace() `Transducer & woss::Transducer::tvr_replace ( double frequency, double tvr ) [inline]`

Replaces a tvr at given frequency

Parameters

frequency	frequency value [hz]. The corresponding PDouble will take Transducer::tvr_precision as precision
tvr	tvr [db re uPa/V @ 1m]

Returns

reference to ∗**this**

References tvr_map, and tvr_precision.

13.74.4.91 tvr_size() `int woss::Transducer::tvr_size ( ) const [inline]`

Returns the number of frequencies stored

Returns

number of frequencies stored

References tvr_map.

13.74.4.92 tvr_upper_bound() `Transducer::CMCIter woss::Transducer::tvr_upper_bound ( double frequency ) const [inline]`

Returns a const iterator to the signed power gain with key $>$ of *frequency* parameter

Parameters

frequency	frequency [hz]

Returns

const iterator to end() if *frequency* is not found

References tvr_map, and tvr_precision.

13.74.4.93 write() [1/2] `bool Transducer::write ( ::std::ostream & stream_out ) const [virtual]`

Write values out to the given stream

Parameters

stream_out	const reference to an ostream instance

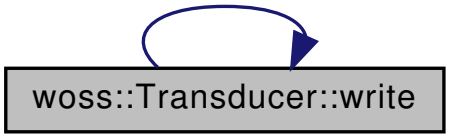
Returns

true if method was successful, false otherwise

References bandwith_3db, beam_power_map, beam_precision, conductance_map, conductance_precision, duty_cycle, max_power, ocv_map, ocv_precision, resonance_frequency, tvr_map, tvr_precision, type_name, and write().

Referenced by write().

Here is the call graph for this function:



13.74.4.94 write() [2/2] bool Transducer::write (::std::ostream & *stream_out*, const ::std::map< PDouble, double > & *map*) const [protected], [virtual]

Writes values in the given map with given precision to the given stream

Parameters

stream_out	const reference to an ostream instance
map	any of the transducer's map

Returns

true if method was successful, false otherwise

13.74.4.95 writeBinary() [1/2] bool Transducer::writeBinary (::std::fstream & *file_out*) const [virtual]

Write values out to the given binary stream

Parameters

stream_out	const reference to an ostream instance

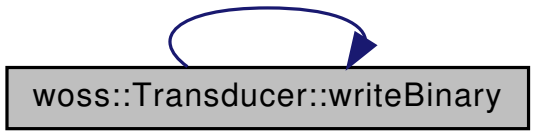
Returns

true if method was successful, false otherwise

References bandwith_3db, beam_power_map, beam_precision, conductance_map, conductance_precision, duty_cycle, max_power, ocv_map, ocv_precision, resonance_frequency, tvr_map, tvr_precision, type_name, and writeBinary().

Referenced by writeBinary().

Here is the call graph for this function:



13.74.4.96 writeBinary() [2/2]

```
bool Transducer::writeBinary (
    ::std::fstream & file_out,
    const ::std::map< PDouble, double > & map ) const  [protected], [virtual]
```

Writes values in the given map with given precision to the given binary stream

Parameters

stream_out	const reference to an ostream instance
map	any of the transducer's map

Returns

true if method was successful, false otherwise

13.74.4.97 writeSPL()

```
bool Transducer::writeSPL (
    ::std::ostream & stream_out,
    double frequency_step,
    double power ) const
```

Write SPL values for all frequencies to the given steram

Parameters

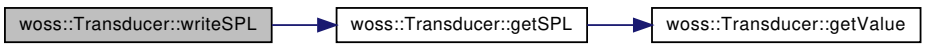
stream_out	const reference to an ostream instance

Returns

true if method was successful, false otherwise

References conductance_map, getSPL(), max_power, and tvr_map.

Here is the call graph for this function:



13.74.4.98 writeVertBeamPattern()

```
bool Transducer::writeVertBeamPattern (
    ::std::ostream & stream_out,
    const CoordZ & tx,
    const CoordZ & rx,
    double init_bearing,
    double vert_rot = 0,
    double horiz_rot = 0,
    double mult_costant = 1,
    double add_costant = 0 ) const
```

Writes the beam pattern to the given stream

Parameters

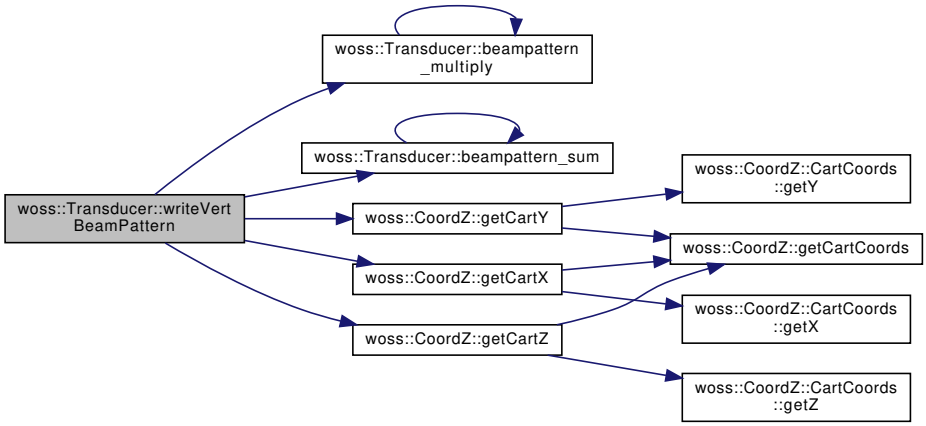
stream_out	const reference to an ostream instance
rotation	beam pattern rotation [decimal degrees]
mult_costant	value to be added
add_costant	value to be multiplied by

Returns

true if method was successful, false otherwise

References beampattern_multiply(), beampattern_sum(), debug, woss::CoordZ::getCartX(), woss::CoordZ::getCartY(), and woss::CoordZ::getCartZ().

Here is the call graph for this function:



13.74.5 Friends And Related Function Documentation

13.74.5.1 operator"!=

```
bool operator!= (
    const Transducer & left,
    const Transducer & right )  [friend]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.74.5.2 operator== `bool operator== ( const Transducer & left, const Transducer & right ) [friend]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

13.74.6 Member Data Documentation

13.74.6.1 bandwith_3db `double woss::Transducer::bandwith_3db [protected]`

the bandwith @ -3dB around the resonance frequency [hz]

Referenced by getBandwith3dB(), isValid(), operator=(), write(), and writeBinary().

13.74.6.2 beam_power_map `BeamPowerMap woss::Transducer::beam_power_map [protected]`

vertical beam pattern map

Referenced by beampattern_begin(), beampattern_clear(), beampattern_empty(), beampattern_end(), beampattern_erase(), beampattern_find(), beampattern_insert(), beampattern_lower_bound(), beampattern_multiply(), beampattern_rbegin(), beampattern_rend(), beampattern_replace(), beampattern_rotate(), beampattern_size(), beampattern_sum(), beampattern_upper_bound(), clearAll(), isValid(), operator=(), setBeamPrecision(), Transducer(), write(), and writeBinary().

13.74.6.3 beam_precision `long double woss::Transducer::beam_precision [protected]`

angle precision [decimal degrees]

Referenced by beampattern_insert(), beampattern_lower_bound(), beampattern_replace(), beampattern_rotate(), beampattern_upper_bound(), getBeamPrecision(), operator=(), setBeamPrecision(), setOCVPrecision(), Transducer(), write(), and writeBinary().

13.74.6.4 conductance_map `ConductanceMap woss::Transducer::conductance_map [protected]`

conductance map

Referenced by clearAll(), conductance_begin(), conductance_clear(), conductance_empty(), conductance_end(), conductance_erase(), conductance_find(), conductance_insert(), conductance_lower_bound(), conductance_rbegin(), conductance_rend(), conductance_replace(), conductance_size(), conductance_upper_bound(), getPowerFromSPL(), getSPL(), isValid(), operator=(), setConductancePrecision(), Transducer(), write(), writeBinary(), and writeSPL().

13.74.6.5 conductance_precision `long double woss::Transducer::conductance_precision [protected]`

frequency precision [hz]

Referenced by conductance_insert(), conductance_lower_bound(), conductance_replace(), conductance_upper_bound(), getConductancePrecision(), getPowerFromSPL(), getSPL(), operator=(), setConductancePrecision(), Transducer(), write(), and writeBinary().

13.74.6.6 debug `bool Transducer::debug = false [static], [protected]`

debug flag

Referenced by getPowerFromSPL(), getSPL(), getValue(), import(), importBinary(), setDebug(), and writeVertBeamPattern().

13.74.6.7 duty_cycle `double woss::Transducer::duty_cycle [protected]`

recommended duty cycle [between 0 and 1]

Referenced by getDutyCycle(), isValid(), operator=(), write(), and writeBinary().

13.74.6.8 has_conical_symmetry `bool woss::Transducer::has_conical_symmetry [protected]`

set to true if transducer has conical symmetry along axis through angle = 0°

Referenced by operator=().

13.74.6.9 max_power `double woss::Transducer::max_power [protected]`

max allowed input power [W]

Referenced by getMaxPower(), getMaxSPL(), getPowerFromSPL(), getSPL(), isValid(), operator=(), write(), writeBinary(), and writeSPL().

13.74.6.10 ocv_map `OCVMap woss::Transducer::ocv_map [protected]`

OCV map

Referenced by clearAll(), isValid(), ocv_begin(), ocv_clear(), ocv_empty(), ocv_end(), ocv_erase(), ocv_find(), ocv_insert(), ocv_lower_bound(), ocv_rbegin(), ocv_rend(), ocv_replace(), ocv_size(), ocv_upper_bound(), operator=(), setOCVPrecision(), Transducer(), write(), and writeBinary().

13.74.6.11 ocv_precision `long double woss::Transducer::ocv_precision [protected]`

frequency precision [hz]

Referenced by getOCVPrecision(), ocv_insert(), ocv_lower_bound(), ocv_replace(), ocv_upper_bound(), operator=(), Transducer(), write(), and writeBinary().

13.74.6.12 resonance_frequency `double woss::Transducer::resonance_frequency [protected]`

resonance frequency of the transducer [hz]

Referenced by getResonanceFrequency(), isValid(), operator=(), write(), and writeBinary().

13.74.6.13 tvr_map `TVRMap woss::Transducer::tvr_map [protected]`

TVR map

Referenced by clearAll(), getPowerFromSPL(), getSPL(), getValue(), isValid(), operator=(), setTVRPrecision(), Transducer(), tvr_begin(), tvr_clear(), tvr_empty(), tvr_end(), tvr_erase(), tvr_find(), tvr_insert(), tvr_lower_bound(), tvr_rbegin(), tvr_rend(), tvr_replace(), tvr_size(), tvr_upper_bound(), write(), writeBinary(), and writeSPL().

13.74.6.14 tvr_precision `long double woss::Transducer::tvr_precision [protected]`

frequency precision [hz]

Referenced by getPowerFromSPL(), getSPL(), getTVRPrecision(), operator=(), setTVRPrecision(), Transducer(), tvr_insert(), tvr_lower_bound(), tvr_replace(), tvr_upper_bound(), write(), and writeBinary().

13.74.6.15 type_name ::std::string woss::Transducer::type_name [protected]

transducer's model name

Referenced by getTypeName(), write(), and writeBinary().

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

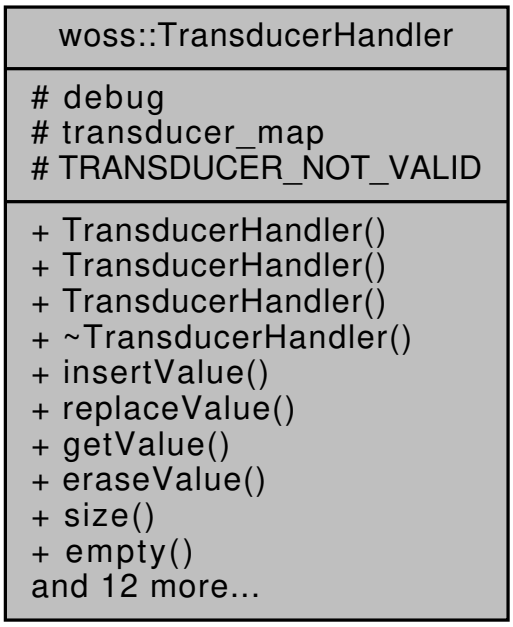
- woss/woss_def/transducer-definitions.h
- woss/woss_def/transducer-definitions.cpp

13.75 woss::TransducerHandler Class Reference

Transducer creator and handler class.

`#include <transducer-handler.h>`

Collaboration diagram for woss::TransducerHandler:



Public Member Functions

- TransducerHandler ()
- TransducerHandler (const TransducerHandler &copy)
- TransducerHandler (TransducerMap &transduc_map)
- bool insertValue (const ::std::string &name, Transducer *const transducer)
- TransducerHandler & replaceValue (const ::std::string &name, Transducer *const transducer)
- const Transducer *const getValue (const ::std::string &name) const
- TransducerHandler & eraseValue (const ::std::string &name)
- int size () const
- bool empty () const
- TransducerHandler & clear ()
- TMCIter begin () const

- TMCIter end () const
- TMCRIter rbegin () const
- TMCRIter rend () const
- virtual bool importValueAscii (const ::std::string &type_name, const ::std::string &file_name)
- virtual bool importValueBinary (const ::std::string &type_name, const ::std::string &file_name)
- virtual bool writeValueAscii (const ::std::string &type_name, const ::std::string &file_name)
- virtual bool writeValueBinary (const ::std::string &type_name, const ::std::string &file_name)
- TransducerHandler & operator= (const TransducerHandler &x)
- TransducerHandler & setDebug (bool flag)
- bool **getDebug** ()

Protected Types

- typedef ::std::map< ::std::string, Transducer * > TransducerMap
- typedef TransducerMap::iterator **TMIter**
- typedef TransducerMap::reverse_iterator **TMRIter**
- typedef TransducerMap::const_iterator **TMCIter**
- typedef TransducerMap::const_reverse_iterator **TMCRIter**

Protected Attributes

- bool debug
- TransducerMap transducer_map

Static Protected Attributes

- static const ::std::string TRANSDUCER_NOT_VALID = "TRANSDUCER_NOT_VALID"

13.75.1 Detailed Description

Transducer creator and handler class.

woss::TransducerHandler class imports and saves woss::Transducer. It provides access to them via string name.

13.75.2 Member Typedef Documentation

13.75.2.1 TransducerMap `typedef ::std::map< ::std::string, Transducer* > woss::TransducerHandler::Transducer [protected]`

Map that links a string to a pointer to a woss::Transducer

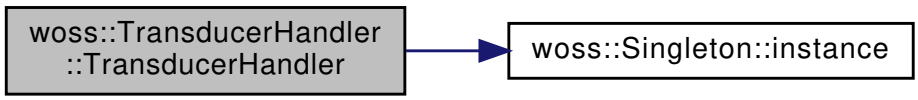
13.75.3 Constructor & Destructor Documentation

13.75.3.1 TransducerHandler() [1/3] TransducerHandler::TransducerHandler ( )

TransducerHandler default constructor

References woss::Singleton< T >::instance(), transducer_map, and TRANSDUCER_NOT_VALID.

Here is the call graph for this function:



13.75.3.2 TransducerHandler() [2/3] TransducerHandler::TransducerHandler (const TransducerHandler & *copy*)

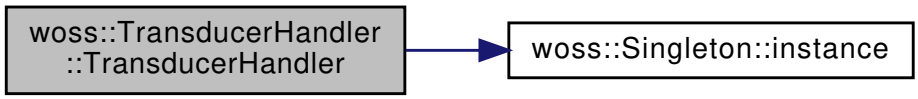
TransducerHandler copy constructor

Parameters

copy	const reference a TransducerHandler

References woss::Singleton< T >::instance(), transducer_map, and TRANSDUCER_NOT_VALID.

Here is the call graph for this function:



13.75.3.3 TransducerHandler() [3/3] TransducerHandler::TransducerHandler (TransducerMap & *transduc_map*)

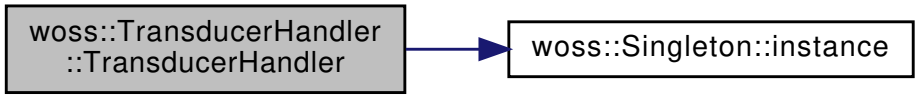
TransducerHandler constructor.

Parameters

transduc_map	TransducerMap to be inserted

References woss::Singleton< T >::instance(), transducer_map, and TRANSDUCER_NOT_VALID.

Here is the call graph for this function:



13.75.4 Member Function Documentation

13.75.4.1 begin() `TransducerHandler::TMCIter woss::TransducerHandler::begin ( ) const [inline]`

Returns a const iterator to the beginning of the tranducer map

Returns

const iterator

References transducer_map.

13.75.4.2 clear() `TransducerHandler & TransducerHandler::clear ( )`

Deletes all pointers and clears the map

Returns

reference to ∗**this**

References transducer_map.

13.75.4.3 empty() `bool woss::TransducerHandler::empty ( ) const [inline]`

Checks if the instance has any stored values

Returns

true if condition applies, *false* otherwise

References transducer_map.

13.75.4.4 end() `TransducerHandler::TMCIter woss::TransducerHandler::end ( ) const [inline]`

Returns a const iterator to the end of the tranducer map

Returns

const iterator

References transducer_map.

13.75.4.5 eraseValue() `TransducerHandler & TransducerHandler::eraseValue ( const ::std::string & name )`

Erases and deletes the pointer to a woss::Transducer for given string

Parameters

name	const reference to string

Returns

reference to ∗**this**

References transducer_map.

13.75.4.6 getValue()

```
const Transducer *const TransducerHandler::getValue (
    const ::std::string & name ) const
```

Returns a const iterator to the woss::Transducer for given string.

Parameters

name	const reference to string

Returns

constant pointer to a constant woss::Transducer that is invalid if the given key is not found.

References transducer_map, and TRANSDUCER_NOT_VALID.

Referenced by woss::BellhopCreator::initializeBhWoss().

13.75.4.7 importValueAscii()

```
bool TransducerHandler::importValueAscii (
    const ::std::string & type_name,
    const ::std::string & file_name ) [virtual]
```

Imports a woss::Transducer from the given file and with given string key

Parameters

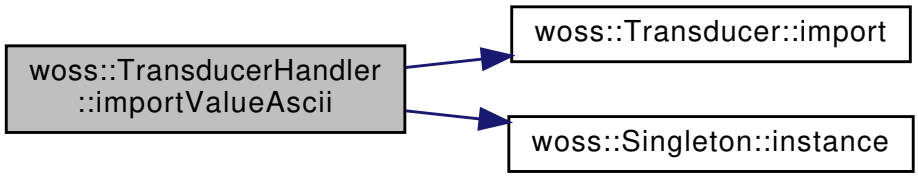
type_name	transducer type name
file_name	file path

Returns

true if method was successful, false otherwise

References woss::Transducer::import(), woss::Singleton< T >::instance(), and transducer_map.

Here is the call graph for this function:



13.75.4.8 importValueBinary()

```
bool TransducerHandler::importValueBinary (
    const ::std::string & type_name,
    const ::std::string & file_name )  [virtual]
```

Imports a woss::Transducer from the given binary file and with given string key

Parameters

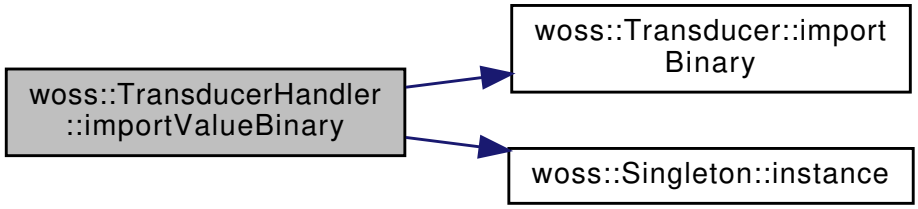
type_name	transducer type name
file_name	file path

Returns

true if method was successful, false otherwise

References woss::Transducer::importBinary(), woss::Singleton< T >::instance(), and transducer_map.

Here is the call graph for this function:



13.75.4.9 insertValue()

```
bool TransducerHandler::insertValue (
    const ::std::string & name,
    Transducer *const transducer )
```

Inserts and doesn't replace if another pointer to a woss::Transducer is found for given key; in this case the given pointer is deleted.

Parameters

name	type name
transducer	constant pointer to a woss::Transducer

Returns

true if inserted, *false* otherwise

References transducer_map.

13.75.4.10 operator=() `TransducerHandler & TransducerHandler::operator= ( const TransducerHandler & x )`

Assignment operator

Parameters

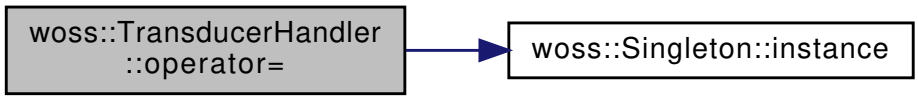
copy	const reference to a TransducerHandler object to be copied

Returns

TransducerHandler reference to *this*

References debug, woss::Singleton< T >::instance(), transducer_map, and TRANSDUCER_NOT_VALID.

Here is the call graph for this function:



13.75.4.11 rbegin() `TransducerHandler::TMCRIter woss::TransducerHandler::rbegin ( ) const [inline]`

Returns a const reverse iterator to the reverse beginning of the tranducer map

Returns

const reverse iterator

References transducer_map.

13.75.4.12 rend() `TransducerHandler::TMCRIter woss::TransducerHandler::rend ( ) const [inline]`

Returns a const reverse iterator to the reverse end of the tranducer map

Returns

const reverse iterator

References transducer_map.

13.75.4.13 replaceValue() `TransducerHandler & TransducerHandler::replaceValue ( const ::std::string & name, Transducer *const transducer )`

Replaces a pointer to a woss::Transducer for given key. The old value is deleted

Parameters

name	type name
transducer	constant pointer to a woss::Transducer

Returns

true if inserted, *false* otherwise

References transducer_map.

13.75.4.14 setDebug() `TransducerHandler & woss::TransducerHandler::setDebug ( bool flag ) [inline]`

Sets debug flag for all instances

Parameters

flag	debug bool

References debug.

13.75.4.15 size() `int woss::TransducerHandler::size ( ) const [inline]`

Returns the number of pointers stored

Returns

number of pointers stored

References transducer_map.

13.75.4.16 writeValueAscii() `bool TransducerHandler::writeValueAscii (`

```
const ::std::string & type_name,
const ::std::string & file_name ) [virtual]
```

Writes a woss::Transducer to the given file

Parameters

type_name	transducer type name
file_name	file path

Returns

true if method was successful, false otherwise

References transducer_map.

13.75.4.17 writeValueBinary() `bool TransducerHandler::writeValueBinary (`

```
const ::std::string & type_name,
const ::std::string & file_name ) [virtual]
```

Writes a woss::Transducer to the given binary file

Parameters

type_name	transducer type name
file_name	file path

Returns

true if method was successful, false otherwise

References transducer_map.

13.75.5 Member Data Documentation

13.75.5.1 debug `bool woss::TransducerHandler::debug [protected]`

debug flag

Referenced by operator=(), and setDebug().

13.75.5.2 transducer_map `TransducerMap woss::TransducerHandler::transducer_map [protected]`

beam pattern map

Referenced by begin(), clear(), empty(), end(), eraseValue(), getValue(), importValueAscii(), importValueBinary(), insertValue(), operator=(), rbegin(), rend(), replaceValue(), size(), TransducerHandler(), writeValueAscii(), and writeValueBinary().

13.75.5.3 TRANSDUCER_NOT_VALID `const ::std::string TransducerHandler::TRANSDUCER_NOT_VALID = "TRANSDUCER_NOT_VALID" [static], [protected]`

string key that represents an invalid woss::Transducer

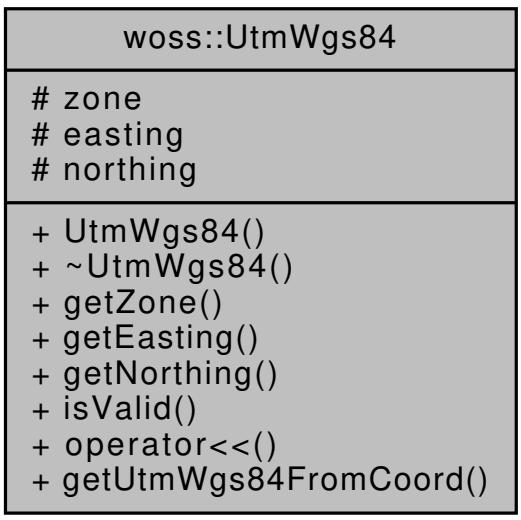
Referenced by getValue(), operator=(), and TransducerHandler().

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

- woss/woss_def/transducer-handler.h
- woss/woss_def/transducer-handler.cpp

13.76 woss::UtmWgs84 Class Reference

Collaboration diagram for woss::UtmWgs84:



Public Member Functions

- UtmWgs84 (int utmZone=COORD_NOT_SET_VALUE, double east=COORD_NOT_SET_VALUE, double north=COORD_NOT_SET_VALUE)
- int **getZone** () const
- double **getEasting** () const
- double **getNorthing** () const
- bool **isValid** () const
- friend::std::ostream & operator<< (::std::ostream &os, const UtmWgs84 &instance)

Static Public Member Functions

- static const UtmWgs84 getUtmWgs84FromCoord (const Coord &coords)

Protected Attributes

- int **zone**
- double easting
- double northing

13.76.1 Constructor & Destructor Documentation

13.76.1.1 UtmWgs84()

```
UtmWgs84::UtmWgs84 (
    int utmZone = COORD_NOT_SET_VALUE,
    double east = COORD_NOT_SET_VALUE,
    double north = COORD_NOT_SET_VALUE )
```

UtmWgs84 constructor.

Parameters

utmZone	zone number
east	Easting value
north	Northing value

Referenced by getUtmWgs84FromCoord().

13.76.2 Member Function Documentation

13.76.2.1 getUtmWgs84FromCoord()

```
const UtmWgs84 UtmWgs84::getUtmWgs84FromCoord (
    const Coord & coords )  [static]
```

Gets destination easting and northing in UTM - WGS84 coordinates from a given Coord

Parameters

coords	reference to a valid Coord object

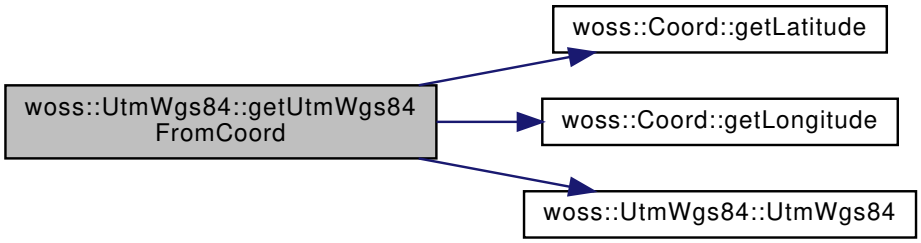
Returns

UTM - WGS84 coordinates

References woss::Coord::EARTH_SEMIMAJOR_AXIS, woss::Coord::getLatitude(), woss::Coord::getLongitude(), and UtmWgs84().

Referenced by woss::BathyUtmCsvDb::getBathyIndex().

Here is the call graph for this function:



13.76.2.2 operator<<() `friend::std::ostream & woss::UtmWgs84::operator<< ( ::std::ostream & os, const UtmWgs84 & instance )`

<< operator

Parameters

os	left operand ostream reference
instance	right operand const CoordZ reference

Returns

os reference after the operation

13.76.3 Member Data Documentation

13.76.3.1 easting `double woss::UtmWgs84::easting [protected]`

Zone number

13.76.3.2 northing `double woss::UtmWgs84::northing [protected]`

Easting value

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

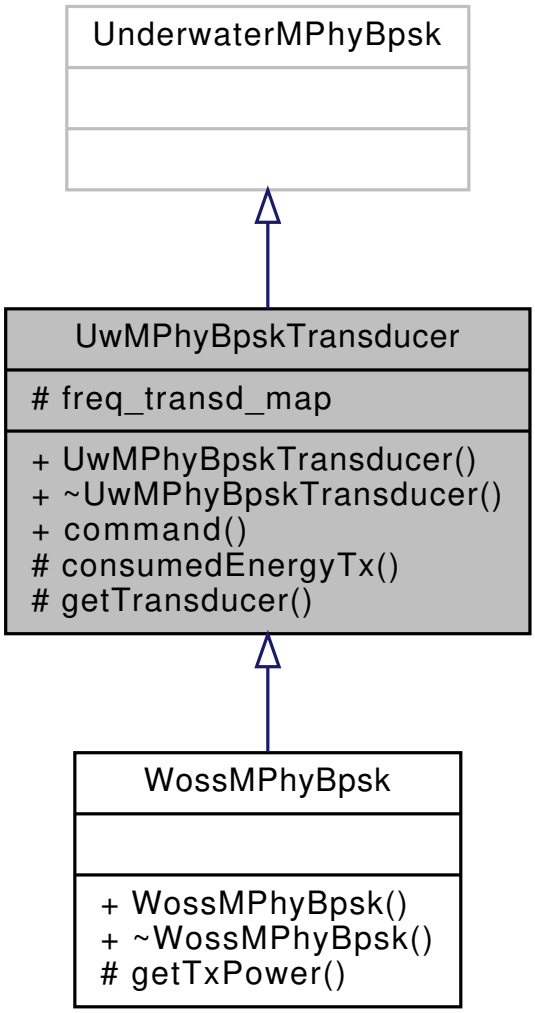
- woss/woss_def/coordinates-definitions.h
- woss/woss_def/coordinates-definitions.cpp

13.77 UwMPhyBpskTransducer Class Reference

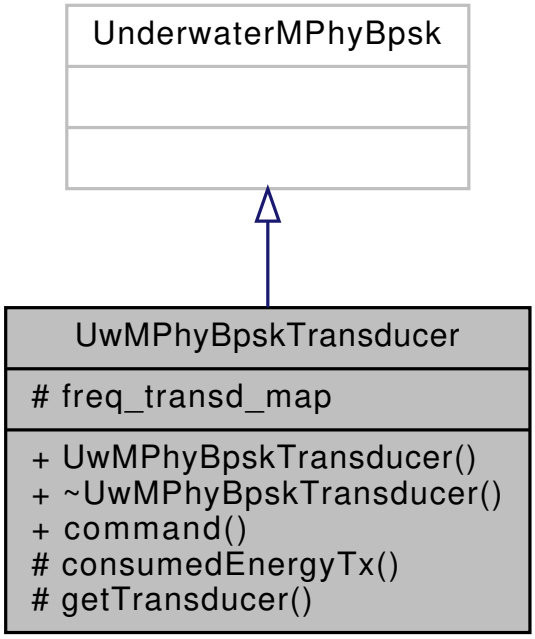
BPSK modulation class with woss::Transducer tx power control.

`#include <uw-woss-bpsk.h>`

Inheritance diagram for UwMPhyBpskTransducer:



Collaboration diagram for UwMPhyBpskTransducer:



Public Member Functions

- virtual int **command** (int argc, const char ∗const ∗argv)

Protected Types

- typedef ::std::map< ::std::pair< double, double >, const woss::Transducer ∗ > **FreqTransducerMap**
- typedef FreqTransducerMap::iterator **FTMIter**
- typedef FreqTransducerMap::const_iterator **FTMCIter**
- typedef FreqTransducerMap::reverse_iterator **FTMRIter**
- typedef FreqTransducerMap::const_reverse_iterator **FTMRCIter**

Protected Member Functions

- virtual double **consumedEnergyTx** (double Ptx, double duration)
- virtual const woss::Transducer ∗const **getTransducer** (double frequency) const

Protected Attributes

- FreqTransducerMap **freq_transd_map**

13.77.1 Detailed Description

BPSK modulation class with woss::Transducer tx power control.

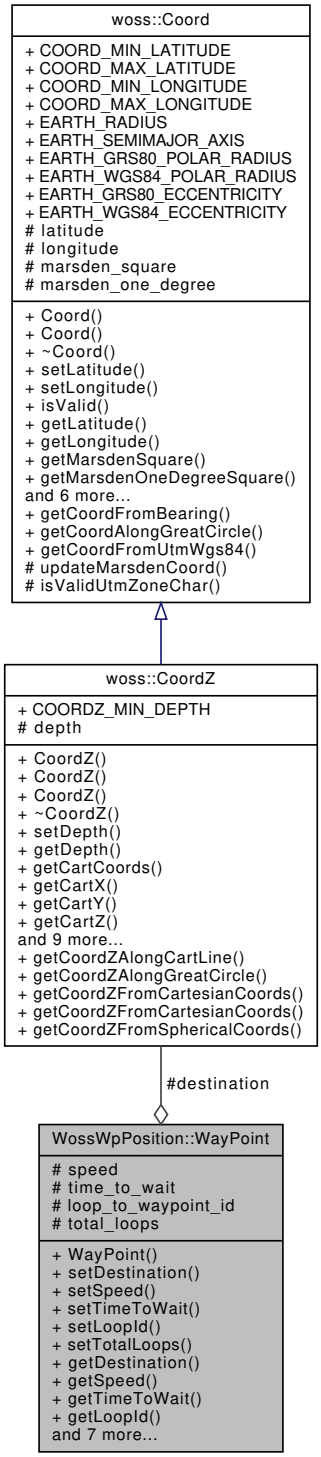
UwMPhyBpskTransducer extends UnderwaterMPhyBpsk adding transducer power computations capabilities.

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

- woss_phy/uw-woss-bpsk.h
- woss_phy/uw-woss-bpsk.cpp

13.78 WossWpPosition::WayPoint Class Reference

Collaboration diagram for WossWpPosition::WayPoint:



Public Member Functions

- **WayPoint** (const woss::CoordZ &destination, double speed=0.0, double time=HUGE_VAL, int loop_id=INT↩ _MAX, int total_loops=0)
- void **setDestination** (const woss::CoordZ &dest)

- void **setSpeed** (double s)
- void **setTimeToWait** (double t)
- void **setLoopId** (int id)
- void **setTotalLoops** (int loops)
- const woss::CoordZ & **getDestination** () const
- double **getSpeed** () const
- double **getTimeToWait** () const
- int **getLoopId** () const
- int **getTotalLoops** () const
- bool **isValid** () const
- bool **hasToLoop** () const
- bool **hasToWait** () const
- bool **hasToStop** () const
- virtual double **getTimeOfArrival** (const WayPoint &dest_waypoint) const
- virtual woss::CoordZ **getCurrentPosition** (const WayPoint &dest_waypoint, double time_elapsed) const

Protected Attributes

- woss::CoordZ **destination**
- double **speed**
- double **time_to_wait**
- int **loop_to_waypoint_id**
- int **total_loops**

Friends

- std::ostream & **operator**<< (std::ostream &os, const WayPoint &instance)

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

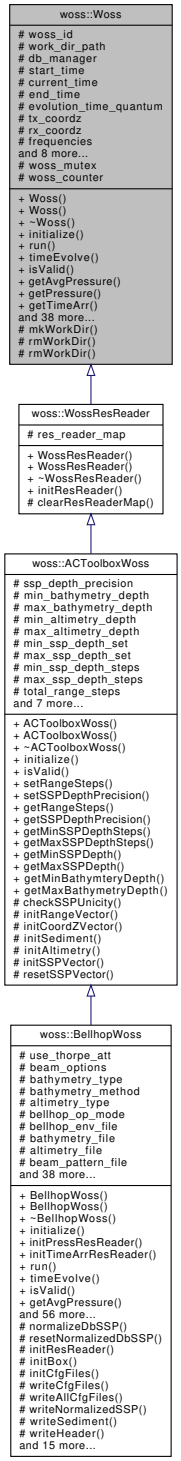
- woss_phy/uw-woss-waypoint-position.h
- woss_phy/uw-woss-waypoint-position.cpp

13.79 woss::Woss Class Reference

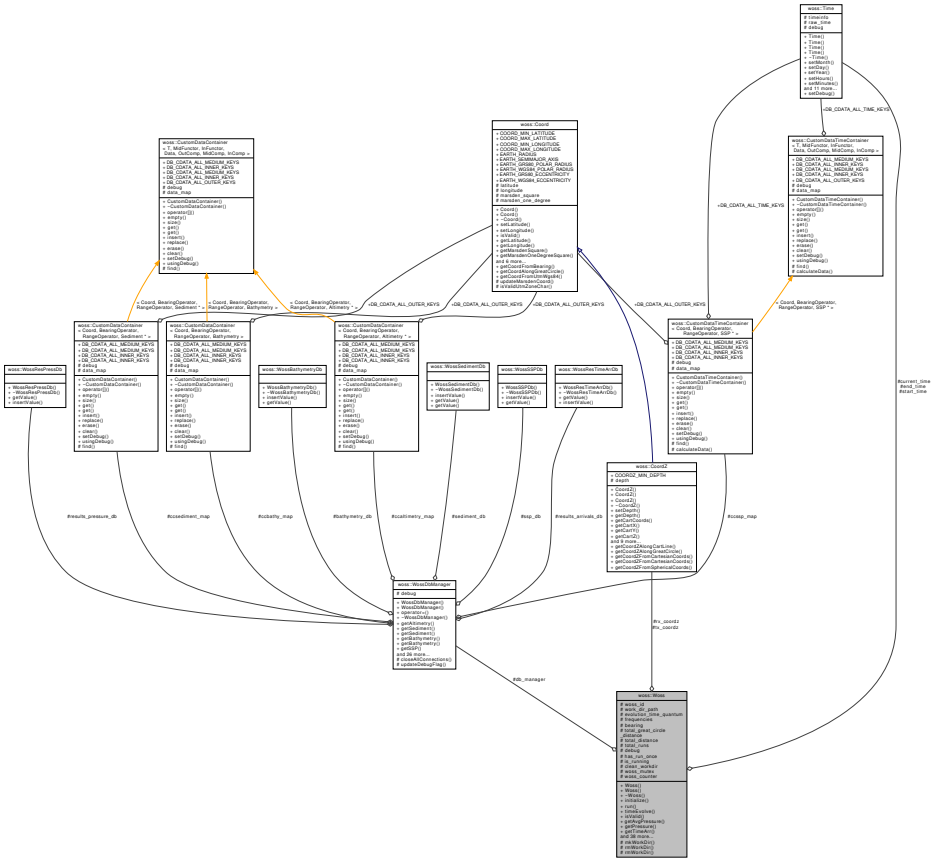
Abstract class that provides the interface for initializing and running a channel simulator.

`#include <woss.h>`

Inheritance diagram for woss::Woss:



Collaboration diagram for woss::Woss:



Public Member Functions

- Woss ()
- Woss (const CoordZ &tx, const CoordZ &rx, const Time &start_t, const Time &end_t, double start_freq, double end_freq, double freq_step)
- virtual bool initialize ()=0
- virtual bool run ()=0
- virtual bool timeEvolve (const Time &time_value)=0
- virtual bool isValid () const =0
- virtual Pressure * getAvgPressure (double frequency, double tx_depth, double start_rx_depth=WOSS_MIN↩ _DEPTH, double start_rx_range=WOSS_MIN_RANGE, double end_rx_depth=WOSS_MAX_DEPTH, double end_rx_range=WOSS_MAX_RANGE) const =0
- virtual Pressure * getPressure (double frequency, double tx_depth, double rx_depth, double rx_range) const =0
- virtual TimeArr * getTimeArr (double frequency, double tx_depth, double rx_depth, double rx_range) const =0
- Woss & setDebug (bool flag)
- Woss & setCleanWorkDir (bool flag)
- Woss & setWorkDirPath (const ::std::string &path)
- Woss & setWossDbManager (const WossDbManager *const ptr)
- Woss & insertFrequency (double freq)
- Woss & insertFrequencies (double freq_start, double freq_end, double freq_step)
- Woss & setFrequencies (const FreqSet &freq_set)
- Woss & eraseFrequency (double freq)
- Woss & clearFrequencies ()

- Woss & setTotalRuns (int runs)
- Woss & setTxCoordZ (const CoordZ &coordz)
- Woss & setRxCoordZ (const CoordZ &coordz)
- Woss & setStartTime (const Time &start_t)
- Woss & setEndTime (const Time &end_t)
- Woss & setEvolutionTimeQuantum (double value)
- int getWossId () const
- ::std::string getWorkDirPath () const
- const FreqSet & getFrequencies () const
- double getMinFrequency () const
- double getMaxFrequency () const
- FreqSCIt freq_begin () const
- FreqSCIt freq_end () const
- FreqSCRIt freq_rbegin () const
- FreqSCRIt freq_rend () const
- FreqSCIt freq_lower_bound (double frequency) const
- FreqSCIt freq_upper_bound (double frequency) const
- int getTotalRuns () const
- CoordZ getTxCoordZ () const
- CoordZ getRxCoordZ () const
- Time getStartTime () const
- Time getCurrentTime () const
- Time getEndTime () const
- double getEvolutionTimeQuantum () const
- double getGreatCircleDistance () const
- double getDistance () const
- double getBearing () const
- bool usingDebug () const
- virtual bool isRunning () const

Protected Member Functions

- virtual bool mkWorkDir (double curr_frequency, int curr_run=0)
- virtual bool rmWorkDir (double curr_frequency, int curr_run=0)
- virtual bool **rmWorkDir** ()

Protected Attributes

- int woss_id
- ::std::string work_dir_path
- const WossDbManager * db_manager
- Time start_time
- Time current_time
- Time end_time
- double evolution_time_quantum
- CoordZ tx_coordz
- CoordZ rx_coordz
- FreqSet frequencies
- double bearing
- double total_great_circle_distance
- double total_distance
- int total_runs
- bool debug
- bool **has_run_once**
- volatile bool is_running
- bool clean_workdir

Static Protected Attributes

- static pthread_spinlock_t woss_mutex = woss::initWossSpinlock(&Woss::woss_mutex , 0)
- static int woss_counter = 0

Friends

- void destroyWossSpinlock ()

13.79.1 Detailed Description

Abstract class that provides the interface for initializing and running a channel simulator.

Woss class has the task to properly initialize and run a channel simulator and to provide its results (with the optional aid of ResReader class).

13.79.2 Constructor & Destructor Documentation

13.79.2.1 Woss() [1/2] Woss::Woss ( )

Woss default constructor. Default constructed objects are not valid

References woss_counter, woss_id, and woss_mutex.

13.79.2.2 Woss() [2/2] Woss::Woss (

```
const CoordZ & tx,
const CoordZ & rx,
const Time & start_t,
const Time & end_t,
double start_freq,
double end_freq,
double freq_step )
```

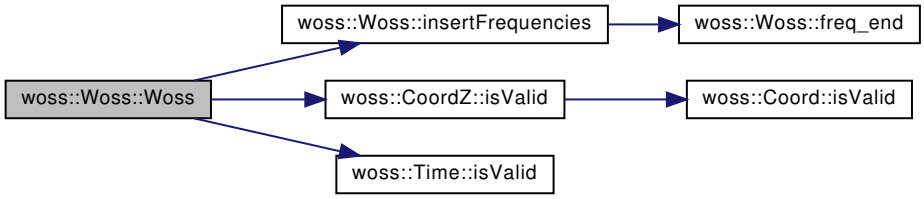
Woss constructor

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_t	const reference to a valid Time object for SSP's averaging purposes (start date time)
end_t	const reference to a valid Time object for SSP's averaging purposes (end date time)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
freq_step	frequency step [Hz]

References end_time, insertFrequencies(), woss::CoordZ::isValid(), woss::Time::isValid(), start_time, woss_counter, woss_id, and woss_mutex.

Here is the call graph for this function:



13.79.3 Member Function Documentation

13.79.3.1 clearFrequencies() `Woss & woss::Woss::clearFrequencies ( ) [inline]`

Erases all frequencies

Returns

reference to ∗**this**

References frequencies.

13.79.3.2 eraseFrequency() `Woss & woss::Woss::eraseFrequency ( double freq ) [inline]`

Erases the given frequency from the FreqSet

Parameters

freq	frequency value [Hz]

Returns

reference to ∗**this**

References frequencies.

13.79.3.3 freq_begin() `FreqSCIt woss::Woss::freq_begin ( ) const [inline]`

Returns a const iterator to the beginning of FreqSet in use

Returns

const FreqSet iterator

References frequencies.

13.79.3.4 freq_end() `FreqSCIt woss::Woss::freq_end ( ) const [inline]`

Returns a const iterator to the end of FreqSet in use

Returns

const FreqSet iterator

References frequencies.

Referenced by insertFrequencies().

13.79.3.5 freq_lower_bound() `FreqSCIt woss::Woss::freq_lower_bound ( double frequency ) const [inline]`

Returns a const iterator to the value >= *frequency* parameter

Parameters

frequency	const reference to a frequency value [Hz]

Returns

const iterator to end() if *frequency* is not found

References frequencies.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWoss and woss::WossManager::getWossTimeArr().

13.79.3.6 freq_rbegin() `FreqSCRIt woss::Woss::freq_rbegin ( ) const [inline]`

Returns a const iterator to the reverse beginning of FreqSet in use

Returns

const FreqSet reverse iterator

References frequencies.

13.79.3.7 freq_rend() `FreqSCRIt woss::Woss::freq_rend ( ) const [inline]`

Returns a const iterator to the reverse of FreqSet in use

Returns

const FreqSet reverse iterator

References frequencies.

13.79.3.8 freq_upper_bound() `FreqSCIt woss::Woss::freq_upper_bound ( double frequency ) const [inline]`

Returns a const iterator to the value > *frequency* parameter

Parameters

frequency	const reference to a frequency value [Hz]

Returns

const iterator to end() if *frequency* is not found

References frequencies.

13.79.3.9 getAvgPressure() `virtual Pressure * woss::Woss::getAvgPressure ( double frequency, double tx_depth, double start_rx_depth = WOSS_MIN_DEPTH, double start_rx_range = WOSS_MIN_RANGE, double end_rx_depth = WOSS_MAX_DEPTH, double end_rx_range = WOSS_MAX_RANGE ) const [pure virtual]`

Gets the average Pressure value in given rx range-depth box

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
start_rx_depth	start receiver depth [m]
start_rx_range	start receiver range [m]
end_rx_depth	end receiver depth [m]
end_rx_range	end receiver range [m]

Returns

a valid Pressure value

Implemented in woss::BellhopWoss.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), and woss::WossManager::getWossPressure().

13.79.3.10 getBearing() `double woss::Woss::getBearing ( ) const [inline]`

Gets the bearing between transmitter and receiver

Returns

bearing [radians]

References bearing.

13.79.3.11 getCurrentTime() `Time woss::Woss::getCurrentTime ( ) const [inline]`

Gets start date time

Returns

a valid Time object

References current_time.

13.79.3.12 getDistance() `double woss::Woss::getDistance ( ) const [inline]`

Gets the distance between transmitter and receiver

Returns

distance value [m]

References total_distance.

13.79.3.13 getEndTime() `Time woss::Woss::getEndTime ( ) const [inline]`

Gets end date time

Returns

a valid Time object

References end_time.

13.79.3.14 getEvolutionTimeQuantum() `double woss::Woss::getEvolutionTimeQuantum ( ) const [inline]`

Gets the evolution time threshold

Returns

time [s]

References evolution_time_quantum.

13.79.3.15 getFrequencies() `const FreqSet & woss::Woss::getFrequencies ( ) const [inline]`

Returns the FreqSet in use

Returns

const reference to frequencies

References frequencies.

13.79.3.16 getGreatCircleDistance() `double woss::Woss::getGreatCircleDistance ( ) const [inline]`

Gets the surface great-circle distance between transmitter and receiver

Returns

distance value [m]

See also

Coord::getGreatCircleDistance()

References total_great_circle_distance.

13.79.3.17 getMaxFrequency() double woss::Woss::getMaxFrequency ( ) const [inline]

Returns the maximum frequency in use

Returns

frequency value [Hz]

References frequencies.

13.79.3.18 getMinFrequency() double woss::Woss::getMinFrequency ( ) const [inline]

Returns the minimum frequency in use

Returns

frequency value [Hz]

References frequencies.

13.79.3.19 getPressure() virtual Pressure * woss::Woss::getPressure (double *frequency*, double *tx_depth*, double *rx_depth*, double *rx_range*) const [pure virtual]

Gets a Pressure value of given range, depths

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

a valid Pressure value

Implemented in woss::BellhopWoss.

Referenced by woss::WossManager::getWossPressure().

13.79.3.20 getRxCoordZ() `CoordZ woss::Woss::getRxCoordZ ( ) const [inline]`

Gets receiver CoordZ

Returns

valid CoordZ object

References rx_coordz.

Referenced by woss::BellhopCreator::initializeBhWoss(), and woss::WossCreator::initializeWoss().

13.79.3.21 getStartTime() `Time woss::Woss::getStartTime ( ) const [inline]`

Gets start date time

Returns

a valid Time object

References start_time.

13.79.3.22 getTimeArr()

```
virtual TimeArr * woss::Woss::getTimeArr (
        double frequency,
        double tx_depth,
        double rx_depth,
        double rx_range ) const [pure virtual]
```

Gets a TimeArr value of given range, depths

Parameters

frequency	frequency [Hz]
tx_depth	transmitter depth [m]
rx_depth	receiver depth [m]
rx_range	receiver range [m]

Returns

a valid Pressure value

Implemented in woss::BellhopWoss.

Referenced by woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), and woss::WossManager::getWossTimeArr().

13.79.3.23 getTotalRuns() `int woss::Woss::getTotalRuns ( ) const [inline]`

Gets the total number of channel simulator's runs

Returns

total number of runs

References total_runs.

13.79.3.24 getTxCoordZ() `CoordZ woss::Woss::getTxCoordZ ( ) const [inline]`

Gets transmitter CoordZ

Returns

valid CoordZ object

References tx_coordz.

Referenced by woss::BellhopCreator::initializeBhWoss(), and woss::WossCreator::initializeWoss().

13.79.3.25 getWorkDirPath() `::std::string woss::Woss::getWorkDirPath ( ) const [inline]`

Returns the work pathname

Returns

string

References work_dir_path.

13.79.3.26 getWossId() `int woss::Woss::getWossId ( ) const [inline]`

Returns the instance identifier

Returns

id number

References woss_id.

Referenced by woss::ArrAscResReader::getArrAscHeader(), woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getA
woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), woss::ArrAscResReader::readMapAvgPressure(),
woss::ArrBinResReader::readMapAvgPressure(), and woss::ShdResReader::readMapAvgPressure().

13.79.3.27 initialize() `bool Woss::initialize ( ) [pure virtual]`

Initializes the channel simulator

Returns

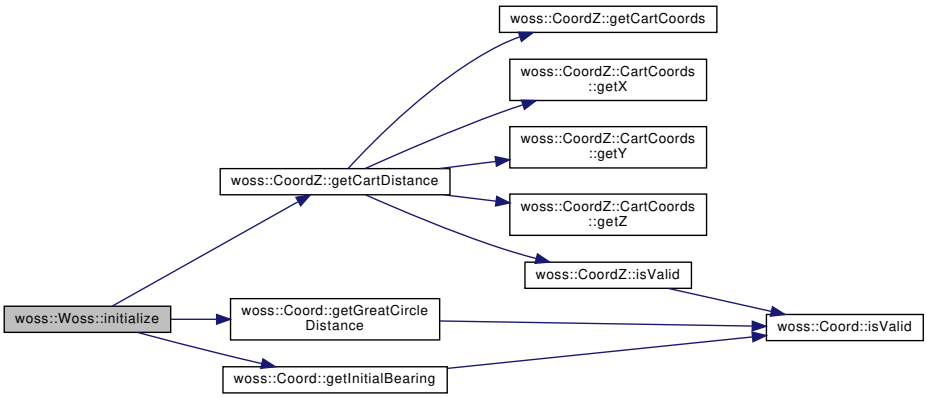
true if method was successful, *false* otherwise

Implemented in woss::ACToolboxWoss, and woss::BellhopWoss.

References bearing, debug, woss::CoordZ::getCartDistance(), woss::Coord::getGreatCircleDistance(), woss::Coord::getInitialBearing rx_coordz, total_distance, total_great_circle_distance, tx_coordz, and woss_id.

Referenced by woss::ACToolboxWoss::initialize(), and woss::BellhopCreator::initializeWoss().

Here is the call graph for this function:



13.79.3.28 insertFrequencies() `Woss & Woss::insertFrequencies (`

```
double freq_start,
double freq_end,
double freq_step )
```

Insert a range of valid frequencies

Parameters

freq_start	a valid frequency value [Hz]
freq_end	a valid frequency value [Hz] >= freq_start
freq_step	a valid frequency value [Hz]

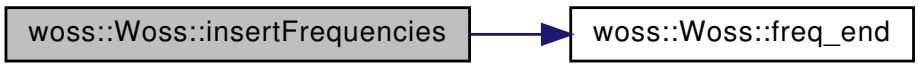
Returns

reference to ∗**this**

References freq_end(), and frequencies.

Referenced by Woss().

Here is the call graph for this function:



13.79.3.29 insertFrequency()

```
Woss & woss::Woss::insertFrequency (
    double freq ) [inline]
```

Insert a frequency value into the FreqSet

Parameters

freq	a valid frequency value [Hz]

Returns

reference to ∗**this**

References frequencies.

13.79.3.30 isRunning()

```
bool Woss::isRunning ( ) const [virtual]
```

Checks if instance is already running the channel simulator

References is_running, and woss_mutex.

Referenced by woss::WossManagerResDbMT::getWossPressure(), and woss::WossManagerResDbMT::getWossTimeArr().

13.79.3.31 isValid()

```
virtual bool woss::Woss::isValid ( ) const [pure virtual]
```

Checks the validity of Woss

Returns

true if it's valid, *false* otherwise

Implemented in woss::ACToolboxWoss, and woss::BellhopWoss.

13.79.3.32 mkWorkDir()

```
bool Woss::mkWorkDir (
    double curr_frequency,
    int curr_run = 0 ) [protected], [virtual]
```

Creates the temporary work directory

Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run value < total_runs

Returns

true if method succeeded, *false* otherwise

References current_time, debug, work_dir_path, and woss_id.

Referenced by woss::BellhopWoss::writeCfgFiles().

13.79.3.33 rmWorkDir()

```
bool Woss::rmWorkDir (
        double curr_frequency,
        int curr_run = 0 )  [protected], [virtual]
```

Removes the temporary work directory

Parameters

curr_frequency	frequency in use [Hz]
curr_run	current run value < total_runs

Returns

true if method succeeded, *false* otherwise

References current_time, work_dir_path, and woss_id.

Referenced by woss::BellhopWoss::removeCfgFiles().

13.79.3.34 run()

```
virtual bool woss::Woss::run ( )  [pure virtual]
```

Runs the channel simulator. It is mandatory to set **is_running** to *true* at the beginning of this function and set it to *false* before returning.

Returns

true if method was successful, *false* otherwise

Implemented in woss::BellhopWoss.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWoss and woss::WossManager::getWossTimeArr().

13.79.3.35 setCleanWorkDir()

```
Woss & woss::Woss::setCleanWorkDir (
        bool flag )  [inline]
```

Sets clean work dir flag

Parameters

flag	debug flag

Returns

reference to ∗**this**

References clean_workdir.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.36 setDebug() `Woss & woss::Woss::setDebug ( bool flag ) [inline]`

Sets debug flag

Parameters

flag	debug flag

Returns

reference to ∗**this**

References debug.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.37 setEndTime() `Woss & woss::Woss::setEndTime ( const Time & end_t ) [inline]`

Sets end date time

Parameters

coordz	const reference to a valid Time object

Returns

reference to ∗**this**

References end_time.

13.79.3.38 setEvolutionTimeQuantum() `Woss & woss::Woss::setEvolutionTimeQuantum ( double value ) [inline]`

Sets time evolution threshold

Parameters

value	time threshold in seconds.

Returns

reference to ∗**this**

References evolution_time_quantum.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.39 setFrequencies() `Woss & woss::Woss::setFrequencies ( const FreqSet & freq_set ) [inline]`

Copys the given FreqSet

Parameters

freq_set	const reference to a FreqSet

Returns

reference to ∗**this**

References frequencies.

13.79.3.40 setRxCoordZ() `Woss & woss::Woss::setRxCoordZ ( const CoordZ & coordz ) [inline]`

Sets receiver CoordZ

Parameters

coordz	const reference to a valid CoordZ object

Returns

reference to ∗**this**

References rx_coordz.

13.79.3.41 setStartTime() Woss & woss::Woss::setStartTime (const Time & *start_t*) [inline]

Sets start date time

Parameters

coordz	const reference to a valid Time object

Returns

reference to ∗**this**

References start_time.

13.79.3.42 setTotalRuns() Woss & woss::Woss::setTotalRuns (int *runs*) [inline]

Sets the total number of channel simulator's runs

Parameters

runs	number of runs

Returns

reference to ∗**this**

References total_runs.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.43 setTxCoordZ() Woss & woss::Woss::setTxCoordZ (const CoordZ & *coordz*) [inline]

Sets transmitter CoordZ

Parameters

coordz	const reference to a valid CoordZ object

Returns

reference to ∗**this**

References tx_coordz.

13.79.3.44 setWorkDirPath() `Woss & woss::Woss::setWorkDirPath ( const ::std::string & path ) [inline]`

Sets the work pathname

Parameters

path	valid pathname

Returns

reference to ∗**this**

References work_dir_path.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.45 setWossDbManager() `Woss & woss::Woss::setWossDbManager ( const WossDbManager *const ptr ) [inline]`

Sets the WossDbManager pointer

Parameters

path	const pointer to a const WossDbManager object

Returns

reference to ∗**this**

References db_manager.

Referenced by woss::WossCreator::initializeWoss().

13.79.3.46 timeEvolve() `virtual bool woss::Woss::timeEvolve ( const Time & time_value ) [pure virtual]`

Performs a time evoulion of all time-dependant parameters

Parameters

time_value	constant reference to a valid Time object (between start_time and end_time)

Returns

true if method was successful, *false* otherwise

Implemented in woss::BellhopWoss.

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWoss and woss::WossManager::getWossTimeArr().

13.79.3.47 usingDebug() `bool woss::Woss::usingDebug ( ) const [inline]`

Checks if instance is using debug

References debug.

Referenced by woss::ArrAscResReader::getArrAscHeader(), woss::ArrBinResReader::getArrBinFile(), woss::ArrBinResReader::getA woss::ShdResReader::getShdFile(), woss::ShdResReader::getShdHeader(), woss::ArrAscResReader::readMapAvgPressure(), woss::ArrBinResReader::readMapAvgPressure(), and woss::ShdResReader::readMapAvgPressure().

13.79.4 Friends And Related Function Documentation

13.79.4.1 destroyWossSpinlock `void destroyWossSpinlock ( ) [friend]`

Function used to destroy the static pthread_spin_t woss_mutex

13.79.5 Member Data Documentation

13.79.5.1 bearing `double woss::Woss::bearing [protected]`

Initial bearing between tx_coordz and rx_coordz [radians]

See also

Coord::getInitialBearing()

Referenced by getBearing(), woss::ACToolboxWoss::initCoordZVector(), and initialize().

13.79.5.2 clean_workdir `bool woss::Woss::clean_workdir [protected]`

flag for removing working dir

Referenced by setCleanWorkDir().

13.79.5.3 current_time `Time woss::Woss::current_time [protected]`

Valid current simulated time (between start and end time)

Referenced by getCurrentTime(), woss::BellhopWoss::initCfgFiles(), woss::ACToolboxWoss::initSSPVector(), mkWorkDir(), rmWorkDir(), and woss::BellhopWoss::timeEvolve().

13.79.5.4 db_manager `const WossDbManager* woss::Woss::db_manager [protected]`

Constant pointer to the DbManager instance

Referenced by woss::ACToolboxWoss::initAltimetry(), woss::ACToolboxWoss::initCoordZVector(), woss::ACToolboxWoss::initSedimer
woss::ACToolboxWoss::initSSPVector(), and setWossDbManager().

13.79.5.5 debug `bool woss::Woss::debug [protected]`

Debug flag

Referenced by woss::BellhopWoss::checkAngles(), woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeOffse
woss::BellhopWoss::getTimeArr(), woss::ACToolboxWoss::initAltimetry(), woss::ACToolboxWoss::initCoordZVector(),
initialize(), woss::ACToolboxWoss::initRangeVector(), woss::ACToolboxWoss::initSediment(), woss::ACToolboxWoss::initSSPVector(),
mkWorkDir(), woss::BellhopWoss::normalizeDbSSP(), woss::BellhopWoss::run(), setDebug(), woss::BellhopWoss::timeEvolve(),
and usingDebug().

13.79.5.6 end_time `Time woss::Woss::end_time [protected]`

Valid end date time value, for SSP averaging purposes

Referenced by getEndTime(), woss::ACToolboxWoss::isValid(), setEndTime(), woss::BellhopWoss::timeEvolve(), and Woss().

13.79.5.7 evolution_time_quantum `double woss::Woss::evolution_time_quantum [protected]`

Time threshold in seconds. For time evolution purposes. A value < 0 ==> evoulion is off A value == 0 ==> no threshold

Referenced by getEvolutionTimeQuantum(), setEvolutionTimeQuantum(), and woss::BellhopWoss::timeEvolve().

13.79.5.8 frequencies `FreqSet woss::Woss::frequencies [protected]`

Frequencies in use

Referenced by woss::BellhopWoss::checkBoundaries(), clearFrequencies(), eraseFrequency(), freq_begin(), freq_end(), freq_lower_bound(), freq_rbegin(), freq_rend(), freq_upper_bound(), getFrequencies(), getMaxFrequency(), getMinFrequency(), insertFrequencies(), insertFrequency(), woss::ACToolboxWoss::isValid(), woss::BellhopWoss::removeAllCfgFiles(woss::BellhopWoss::run(), setFrequencies(), and woss::BellhopWoss::writeAllCfgFiles().

13.79.5.9 is_running `volatile bool woss::Woss::is_running [protected]`

Running flag

Referenced by isRunning(), and woss::BellhopWoss::run().

13.79.5.10 rx_coordz `CoordZ woss::Woss::rx_coordz [protected]`

Receiver CoordZ

Referenced by woss::BellhopWoss::checkBoundaries(), woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeO getRxCoordZ(), woss::ACToolboxWoss::initAltimetry(), woss::ACToolboxWoss::initCoordZVector(), initialize(), woss::ACToolboxWoss::isValid(), setRxCoordZ(), and woss::BellhopWoss::writeReceiver().

13.79.5.11 start_time `Time woss::Woss::start_time [protected]`

Valid start date time value, for SSP averaging purposes

Referenced by getStartTime(), woss::ACToolboxWoss::isValid(), setStartTime(), woss::BellhopWoss::timeEvolve(), and Woss().

13.79.5.12 total_distance `double woss::Woss::total_distance [protected]`

Cartesian distance between tx_coordz and rx_coordz [m]

See also

CoordZ::getDistance()

Referenced by woss::BellhopWoss::checkAngles(), getDistance(), woss::BellhopWoss::initialize(), and initialize().

13.79.5.13 total_great_circle_distance `double woss::Woss::total_great_circle_distance [protected]`

Surface great-circle distance between tx_coordz and rx_coordz [m]

See also

Coord::getGreatCircleDistance()

Referenced by woss::BellhopWoss::checkAngles(), woss::BellhopWoss::checkBoundaries(), woss::BellhopWoss::checkRangeOffsets getGreatCircleDistance(), woss::BellhopWoss::initialize(), initialize(), woss::ACToolboxWoss::initRangeVector(), and woss::BellhopWoss::writeReceiver().

13.79.5.14 total_runs `int woss::Woss::total_runs [protected]`

Total number of channel simulator's runs

Referenced by woss::BellhopWoss::getAvgPressure(), woss::BellhopWoss::getPressure(), woss::BellhopWoss::getTimeArr(), getTotalRuns(), woss::BellhopWoss::removeAllCfgFiles(), woss::BellhopWoss::run(), setTotalRuns(), and woss::BellhopWoss::writeAllCfgFiles().

13.79.5.15 tx_coordz `CoordZ woss::Woss::tx_coordz [protected]`

Transmitter CoordZ

Referenced by woss::BellhopWoss::checkBoundaries(), woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeO getTxCoordZ(), woss::ACToolboxWoss::initAltimetry(), woss::ACToolboxWoss::initCoordZVector(), initialize(), woss::ACToolboxWoss::initSediment(), woss::ACToolboxWoss::initSSPVector(), woss::ACToolboxWoss::isValid(), woss::BellhopWoss::normalizeDbSSP(), setTxCoordZ(), and woss::BellhopWoss::writeTransmitter().

13.79.5.16 work_dir_path `::std::string woss::Woss::work_dir_path [protected]`

Directory path for temporary files (e.g. channel simulator files)

Referenced by getWorkDirPath(), woss::BellhopWoss::initCfgFiles(), mkWorkDir(), rmWorkDir(), and setWorkDirPath().

13.79.5.17 woss_counter `int Woss::woss_counter = 0 [static], [protected]`

Unique instances id-counter

Referenced by Woss().

13.79.5.18 woss_id `int woss::Woss::woss_id [protected]`

id of specific instance

Referenced by woss::BellhopWoss::checkAngles(), woss::BellhopWoss::checkDepthOffsets(), woss::BellhopWoss::checkRangeOffse
woss::BellhopWoss::getTimeArr(), getWossId(), woss::ACToolboxWoss::initAltimetry(), woss::BellhopWoss::initCfgFiles(),
woss::ACToolboxWoss::initCoordZVector(), initialize(), woss::ACToolboxWoss::initRangeVector(), woss::ACToolboxWoss::initSedimen
woss::ACToolboxWoss::initSSPVector(), mkWorkDir(), rmWorkDir(), woss::BellhopWoss::run(), woss::BellhopWoss::setBhMode(),
woss::BellhopWoss::timeEvolve(), and Woss().

13.79.5.19 woss_mutex `pthread_spinlock_t Woss::woss_mutex = woss::initWossSpinlock( &Woss↩ ::woss_mutex , 0 ) [static], [protected]`

Spinlock for syncronization purposes

Referenced by isRunning(), and Woss().

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

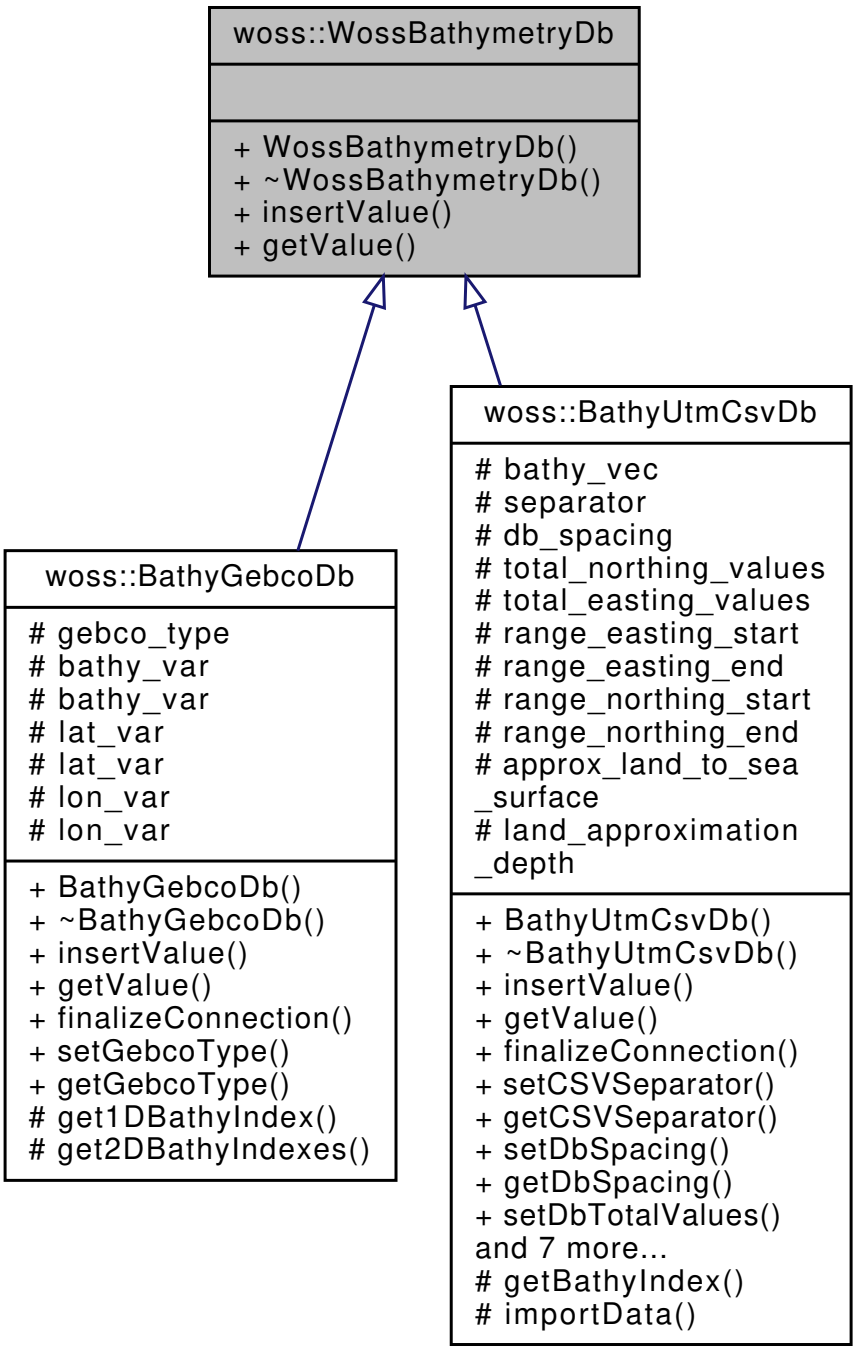
- woss/woss.h
- woss/woss.cpp

13.80 woss::WossBathymetryDb Class Reference

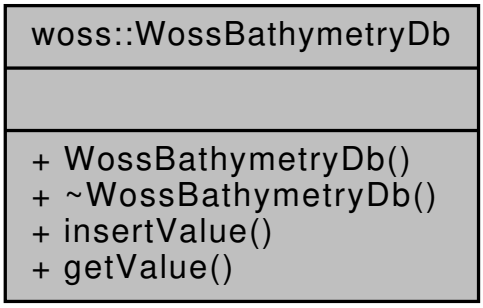
Data behaviour class for bathymetry database.

`#include <woss-db.h>`

Inheritance diagram for woss::WossBathymetryDb:



Collaboration diagram for woss::WossBathymetryDb:



Public Member Functions

- virtual bool insertValue (const Coord &coordinates, const Bathymetry &bathymetry_value)=0
- virtual Bathymetry getValue (const Coord &coords) const =0

13.80.1 Detailed Description

Data behaviour class for bathymetry database.

WossBathymetryDb is the prototype of any bathymetry dabase in WOSS

See also

BathyGebcoDb

13.80.2 Member Function Documentation

13.80.2.1 getValue() `virtual Bathymetry woss::WossBathymetryDb::getValue ( const Coord & coords ) const [pure virtual]`

Returns the positive depth value (woss::Bathymetry) for given coordinates, if present in the database

Parameters

coords	const reference to a valid Coord object

Returns

positive depth value [m] if coordinates are found, *HUGE_VAL* otherwise

Implemented in woss::BathyGebcoDb, and woss::BathyUtmCsvDb.

Referenced by woss::WossDbManager::getBathymetry().

13.80.2.2 insertValue()

```
virtual bool woss::WossBathymetryDb::insertValue (
        const Coord & coordinates,
        const Bathymetry & bathymetry_value )  [pure virtual]
```

Inserts the given woss::Bathymetry value in the database for given coordinates

Parameters

coordinates	const reference to a valid woss::Coord object
bathymetry_value	const reference to woss::Bathymetry value to be inserted

Returns

true if method was successful, *false* otherwise

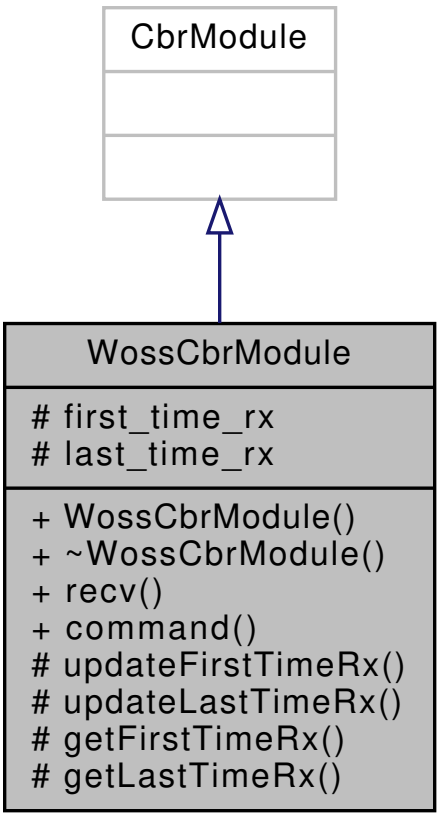
Implemented in woss::BathyGebcoDb, and woss::BathyUtmCsvDb.

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

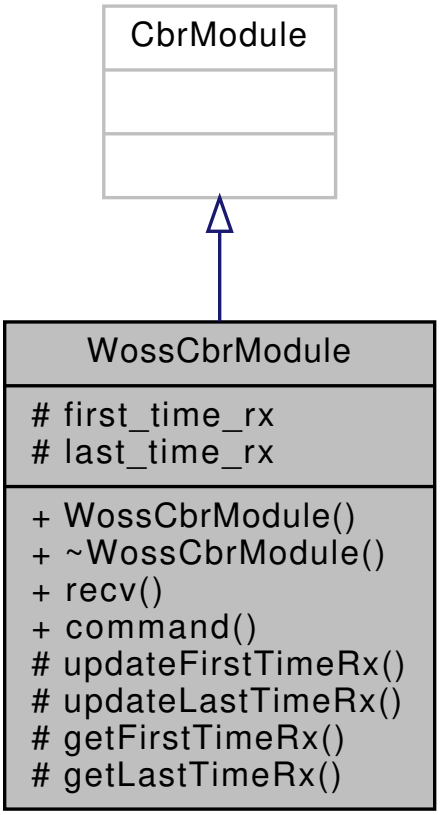
- woss/woss_db/woss-db.h

13.81 WossCbrModule Class Reference

Inheritance diagram for WossCbrModule:



Collaboration diagram for WossCbrModule:



Public Member Functions

- virtual void **recv** (Packet ∗)
- virtual int **command** (int argc, const char ∗const ∗argv)

Protected Member Functions

- void **updateFirstTimeRx** (double time)
- void **updateLastTimeRx** (double time)
- double **getFirstTimeRx** ()
- double **getLastTimeRx** ()

Protected Attributes

- double **first_time_rx**
- double **last_time_rx**

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

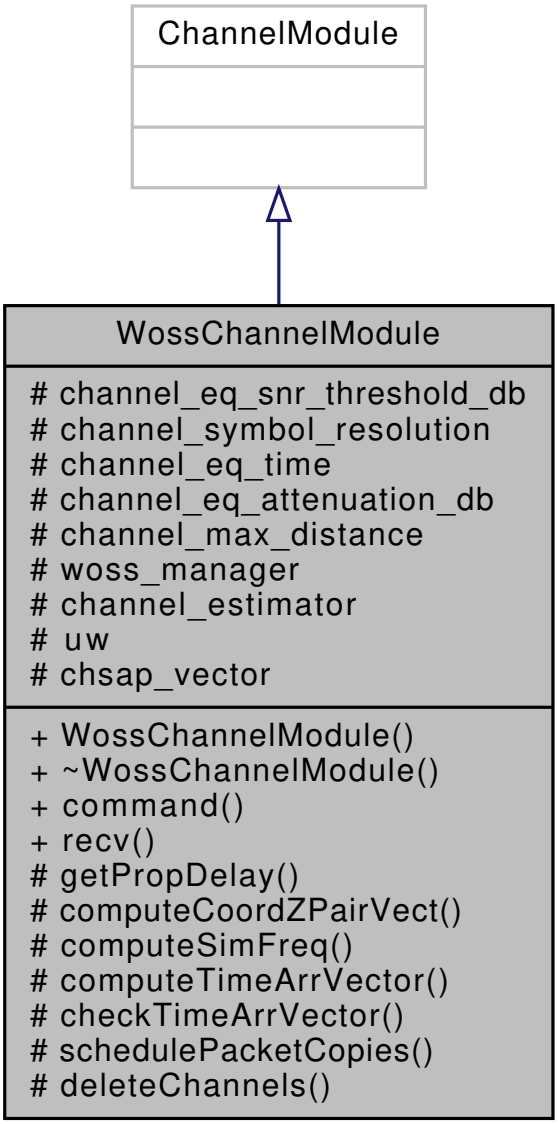
- woss_phy/uw-woss-cbr.h
- woss_phy/uw-woss-cbr.cpp

13.82 WossChannelModule Class Reference

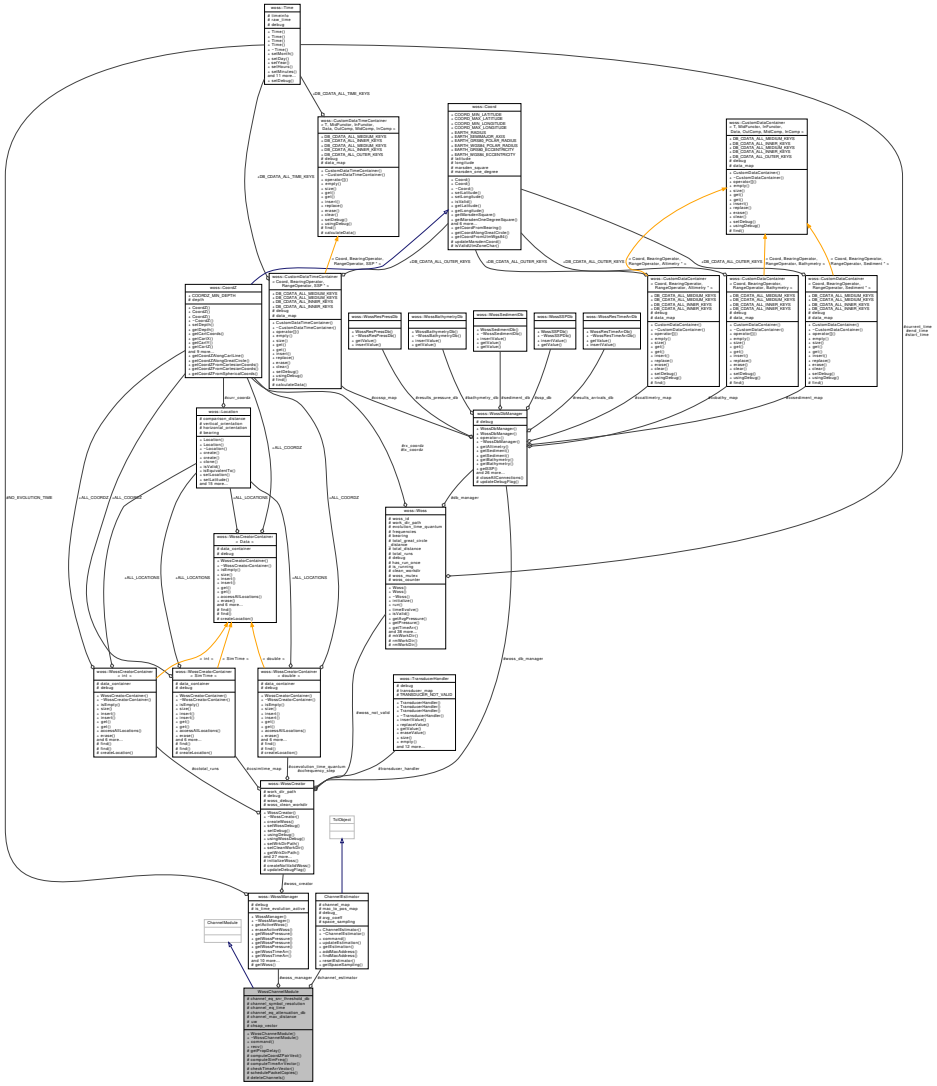
WossChannelModule class for channel calculations with WOSS.

`#include <uw-woss-channel.h>`

Inheritance diagram for WossChannelModule:



Collaboration diagram for WossChannelModule:



Public Member Functions

- virtual int **command** (int argc, const char ∗const ∗argv)
- virtual void **recv** (Packet ∗p, ChSAP ∗chsap)

Protected Types

- typedef ::std::vector< ChSAP ∗ > **ChSAPVector**

Protected Member Functions

- double **getPropDelay** (const woss::CoordZ &s, const woss::CoordZ &d)
- woss::CoordZPairVect **computeCoordZPairVect** (ChSAP ∗dest)
- woss::SimFreq **computeSimFreq** (Packet ∗p)
- woss::TimeArrVector **computeTimeArrVector** (const woss::CoordZPairVect &coords, const woss::SimFreq &sim_freq)
- void **checkTimeArrVector** (const woss::CoordZPairVect &coords, woss::TimeArrVector &channels)
- void **schedulePacketCopies** (const woss::CoordZPairVect &coords, const woss::TimeArrVector &channels, const woss::SimFreq &sim_freq, Packet ∗p)
- void **deleteChannels** (woss::TimeArrVector &channels)

Protected Attributes

- double **channel_eq_snr_threshold_db**
- double **channel_symbol_resolution**
- double **channel_eq_time**
- double **channel_eq_attenuation_db**
- double **channel_max_distance**
- woss::WossManager * **woss_manager**
- ChannelEstimator * **channel_estimator**
- Underwater **uw**
- ChSAPVector **chsap_vector**

13.82.1 Detailed Description

WossChannelModule class for channel calculations with WOSS.

WossChannelModule extends ChannelModule for channel calculations with WOSS

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

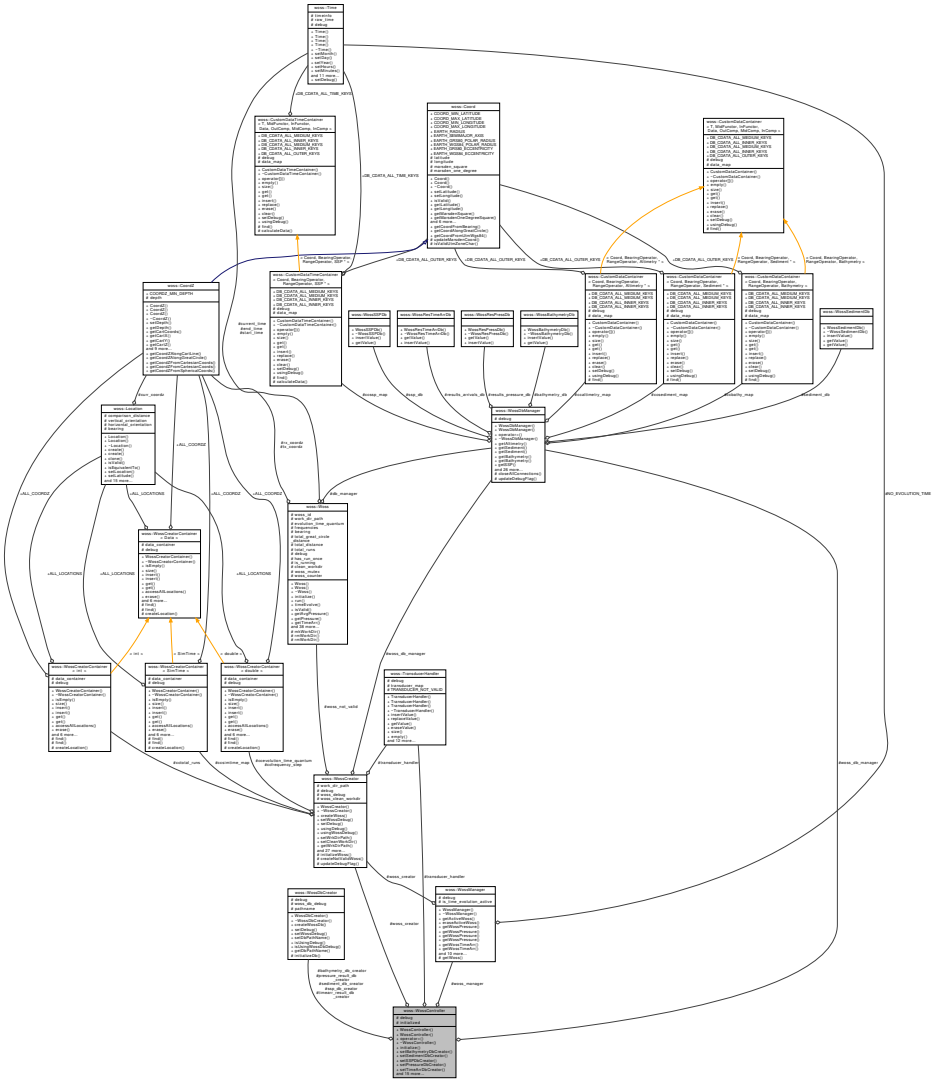
- woss_phy/uw-woss-channel.h
- woss_phy/uw-woss-channel.cpp

13.83 woss::WossController Class Reference

Class for managing all WOSS classes involved.

`#include <woss-controller.h>`

Collaboration diagram for woss::WossController:



Public Member Functions

- WossController ()
- WossController (WossController &copy)
- WossController & operator= (WossController &copy)
- virtual ~WossController ()
- bool initialize ()
- WossController & **setBathymetryDbCreator** (WossDbCreator *const ptr)
- WossController & **setSedimentDbCreator** (WossDbCreator *const ptr)
- WossController & **setSSPDbCreator** (WossDbCreator *const ptr)
- WossController & **setPressureDbCreator** (WossDbCreator *const ptr)
- WossController & **setTimeArrDbCreator** (WossDbCreator *const ptr)
- WossController & **setWossCreator** (WossCreator *const ptr)
- WossController & **setWossDbManager** (WossDbManager *const ptr)
- WossController & **setWossManager** (WossManager *const ptr)
- WossController & **setTransducerHandler** (TransducerHandler *const ptr)
- void **setDebug** (bool flag)
- const WossDbCreator *const **getBathymetryDbCreator** () const

- const WossDbCreator *const **getSedimentDbCreator** () const
- const WossDbCreator *const **getSSPDbCreator** () const
- const WossDbCreator *const **getPressureDbCreator** () const
- const WossDbCreator *const **getTimeArrDbCreator** () const
- const WossCreator *const **getWossCreator** () const
- const WossDbManager *const **getWossDbManager** () const
- WossManager *const **getWossManager** () const
- TransducerHandler *const **getTransducerHandler** () const
- bool **getDebug** () const

Protected Attributes

- double debug
- bool initialized
- WossDbCreator * **bathymetry_db_creator**
- WossDbCreator * **sediment_db_creator**
- WossDbCreator * **ssp_db_creator**
- WossDbCreator * **pressure_result_db_creator**
- WossDbCreator * **timearr_result_db_creator**
- WossCreator * **woss_creator**
- WossDbManager * **woss_db_manager**
- WossManager * **woss_manager**
- TransducerHandler * **transducer_handler**

13.83.1 Detailed Description

Class for managing all WOSS classes involved.

woss::WossController is a class that sets all needed connections between primary WOSS classes. It should be used with woss::Singleton for safety reasons. (e.g. woss::Singleton<woss::WossController>)

13.83.2 Constructor & Destructor Documentation

13.83.2.1 WossController() [1/2] `WossController::WossController ( )`

Default constructor

13.83.2.2 WossController() [2/2] `WossController::WossController ( WossController & copy )`

Copy constructor (no const here, we have to modify the copy)

References initialized.

13.83.2.3 ~WossController() `WossController::~WossController ( ) [virtual]`

Destructor

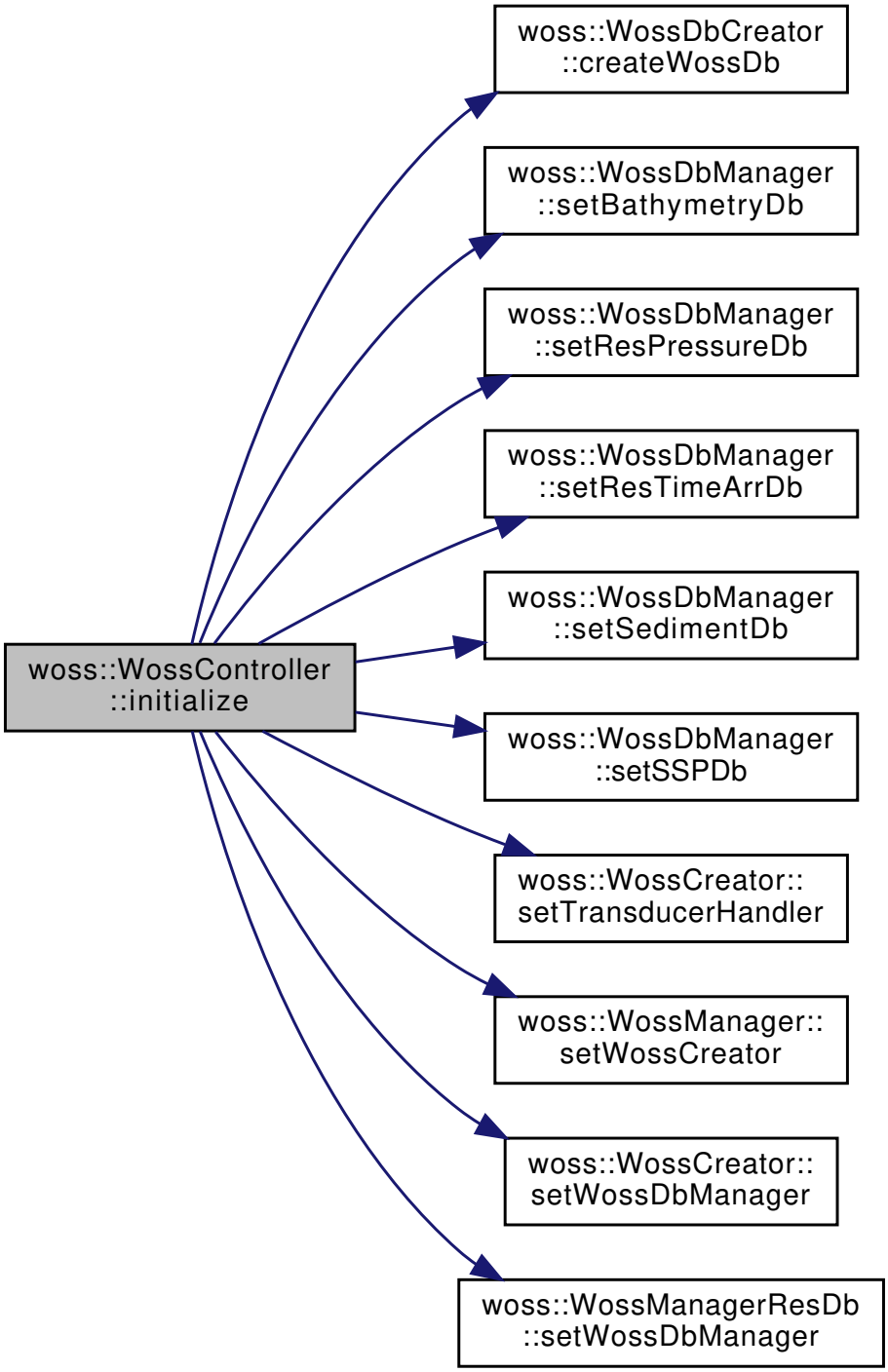
13.83.3 Member Function Documentation

13.83.3.1 initialize() `bool WossController::initialize ( )`

Initializes all connections

References woss::WossDbCreator::createWossDb(), initialized, woss::WossDbManager::setBathymetryDb(), woss::WossDbManager::setResPressureDb(), woss::WossDbManager::setResTimeArrDb(), woss::WossDbManager::setSedimentDb woss::WossDbManager::setSSPDb(), woss::WossCreator::setTransducerHandler(), woss::WossManager::setWossCreator(), woss::WossCreator::setWossDbManager(), and woss::WossManagerResDb::setWossDbManager().

Here is the call graph for this function:



13.83.3.2 operator=() `WossController & WossController::operator= ( WossController & copy )`

Assignment operator (no const here, we have to modify the copy)

References debug, and initialized.

13.83.4 Member Data Documentation

13.83.4.1 debug `double woss::WossController::debug [protected]`

Debug flag

Referenced by operator=().

13.83.4.2 initialized `bool woss::WossController::initialized [protected]`

Initialized flag

Referenced by initialize(), operator=(), and WossController().

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

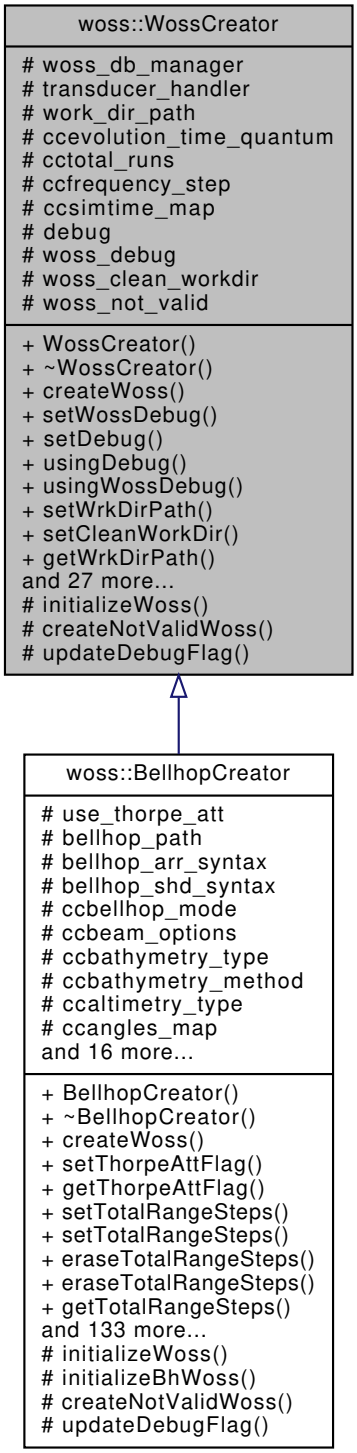
- woss/woss-controller.h
- woss/woss-controller.cpp

13.84 woss::WossCreator Class Reference

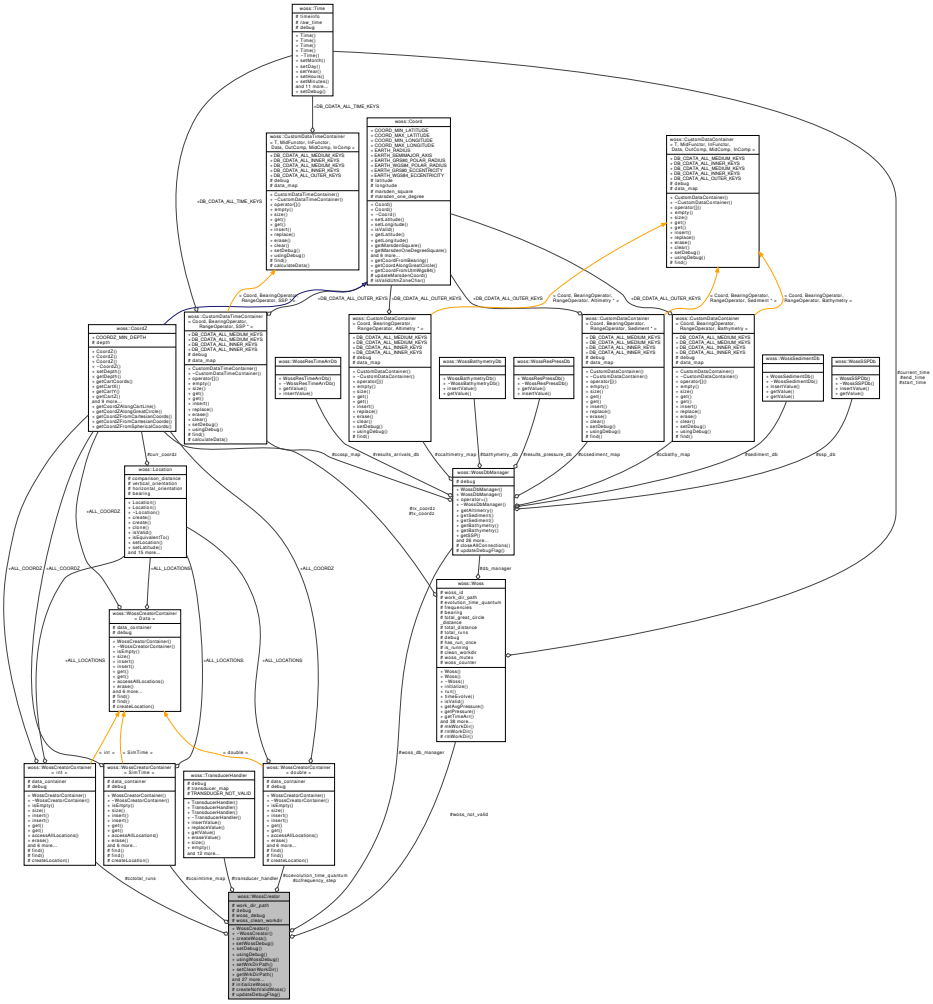
Abstract class that provides correctly initialized Woss objects.

`#include <woss-creator.h>`

Inheritance diagram for woss::WossCreator:



Collaboration diagram for woss::WossCreator:



Public Member Functions

- WossCreator ()
- virtual Woss *const createWoss (const CoordZ &tx, const CoordZ &rx, double start_freq, double end_freq) const =0
- WossCreator & setWossDebug (bool flag)
- WossCreator & setDebug (bool flag)
- bool usingDebug () const
- bool usingWossDebug () const
- WossCreator & setWrkDirPath (const ::std::string &path)
- WossCreator & setCleanWorkDir (bool flag)
- ::std::string getWrkDirPath () const
- WossCreator & setFrequencyStep (double f_step, const CoordZ &tx, const CoordZ &rx)
- WossCreator & setFrequencyStep (double f_step, Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)
- double getFrequencyStep (const CoordZ &tx, const CoordZ &rx) const
- double getFrequencyStep (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS) const
- WossCreator & eraseFrequencyStep (const CoordZ &tx, const CoordZ &rx)
- WossCreator & eraseFrequencyStep (Location *const tx=CCDouble::ALL_LOCATIONS, Location *const rx=CCDouble::ALL_LOCATIONS)

- WossCreator & setEvolutionTimeQuantum (double value, const CoordZ &tx, const CoordZ &rx)
- WossCreator & setEvolutionTimeQuantum (double value, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- double getEvolutionTimeQuantum (const CoordZ &tx, const CoordZ &rx) const
- double getEvolutionTimeQuantum (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS) const
- WossCreator & eraseEvolutionTimeQuantum (const CoordZ &tx, const CoordZ &rx)
- WossCreator & eraseEvolutionTimeQuantum (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- WossCreator & setTotalRuns (int runs, const CoordZ &tx, const CoordZ &rx)
- WossCreator & setTotalRuns (int runs, Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- int getTotalRuns (const CoordZ &tx, const CoordZ &rx) const
- int getTotalRuns (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS) const
- WossCreator & eraseTotalRuns (const CoordZ &tx, const CoordZ &rx)
- WossCreator & eraseTotalRuns (Location *const tx=CCInt::ALL_LOCATIONS, Location *const rx=CCInt::ALL_LOCATIONS)
- WossCreator & setSimTime (const SimTime &simtime, const CoordZ &tx, const CoordZ &rx)
- WossCreator & setSimTime (const SimTime &simtime, Location *const tx=CCSimTime::ALL_LOCATIONS, Location *const rx=CCSimTime::ALL_LOCATIONS)
- SimTime getSimTime (const CoordZ &tx, const CoordZ &rx) const
- SimTime getSimTime (Location *const tx=CCSimTime::ALL_LOCATIONS, Location *const rx=CCSimTime::ALL_LOCATIONS) const
- WossCreator & eraseSimTime (const CoordZ &tx, const CoordZ &rx)
- WossCreator & eraseSimTime (Location *const tx=CCSimTime::ALL_LOCATIONS, Location *const rx=CCSimTime::ALL_LOCATIONS)
- WossCreator & setWossDbManager (const WossDbManager *const ptr)
- WossCreator & setTransducerHandler (const TransducerHandler *const ptr)
- const Woss & **getWossNotValid** () const

Protected Types

- typedef WossCreatorContainer< SimTime > CCSimTime
- typedef WossCreatorContainer< double > CCDouble
- typedef WossCreatorContainer< int > CCInt

Protected Member Functions

- virtual bool initializeWoss (Woss *const woss_ptr) const =0
- virtual const Woss * **createNotValidWoss** () const =0
- virtual void updateDebugFlag ()

Protected Attributes

- const WossDbManager * **woss_db_manager**
- const TransducerHandler * **transducer_handler**
- ::std::string work_dir_path
- CCDouble ccevolution_time_quantum
- CCInt cctotal_runs
- CCDouble ccfrequency_step
- CCSimTime ccsimtime_map
- bool debug
- bool woss_debug
- bool woss_clean_workdir

Static Protected Attributes

- static const Woss * **woss_not_valid** = NULL

13.84.1 Detailed Description

Abstract class that provides correctly initialized Woss objects.

WossCreator provides interface for creation and initializiation of Woss objects, relieving the user from this task

13.84.2 Member Typedef Documentation

13.84.2.1 CCDouble `typedef WossCreatorContainer< double > woss::WossCreator::CCDouble [protected]`

double WossCreatorContainer

13.84.2.2 CCInt `typedef WossCreatorContainer< int > woss::WossCreator::CCInt [protected]`

int WossCreatorContainer

13.84.2.3 CCSimTime `typedef WossCreatorContainer< SimTime > woss::WossCreator::CCSimTime [protected]`

SimTime WossCreatorContainer

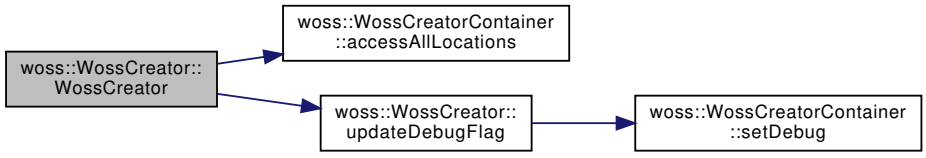
13.84.3 Constructor & Destructor Documentation

13.84.3.1 WossCreator() `WossCreator::WossCreator ( )`

WossCreator default constructor

References woss::WossCreatorContainer< Data >::accessAllLocations(), ccfrequency_step, and updateDebugFlag().

Here is the call graph for this function:



13.84.4 Member Function Documentation

13.84.4.1 createWoss() `virtual Woss *const woss::WossCreator::createWoss ( const CoordZ & tx, const CoordZ & rx, double start_freq, double end_freq ) const [pure virtual]`

Returns a pointer to valid Woss for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

pointer to properly initialized Woss object

Implemented in woss::BellhopCreator.

13.84.4.2 eraseEvolutionTimeQuantum() [1/2]

```
WossCreator & woss::WossCreator::eraseEvolutionTimeQuantum (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases the time evolution threshold

Parameters

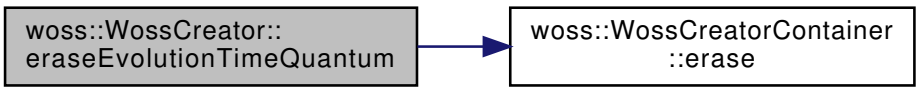
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.3 eraseEvolutionTimeQuantum() [2/2]

```
WossCreator & woss::WossCreator::eraseEvolutionTimeQuantum (
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Erases the time evolution threshold

Parameters

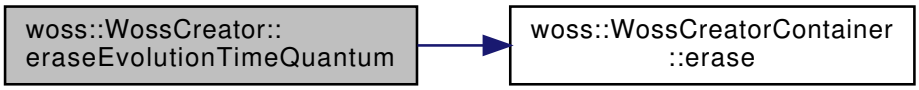
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.4 eraseFrequencyStep() [1/2]

```
WossCreator & woss::WossCreator::eraseFrequencyStep (
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Erases the frequency step for given transmitter, receiver woss::CoordZ

Parameters

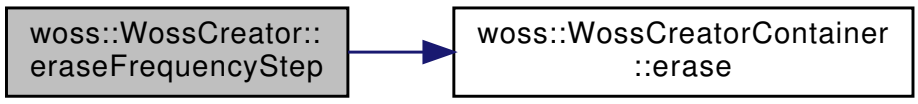
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccfrequency_step, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.5 eraseFrequencyStep() [2/2]

```
WossCreator & woss::WossCreator::eraseFrequencyStep (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Erases the frequency step for given transmitter, receiver woss::Location

Parameters

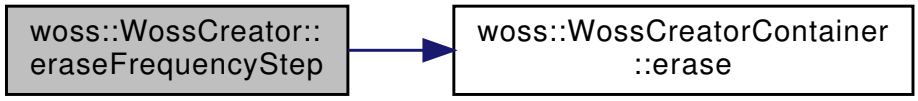
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccfrequency_step, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.6 eraseSimTime() [1/2]

```
WossCreator & woss::WossCreator::eraseSimTime (
    const CoordZ & tx,
    const CoordZ & rx ) [inline]
```

Erases the SimTime for given transmitter, receiver woss::CoordZ

Parameters

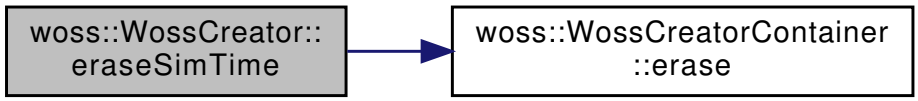
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccsimtime_map, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.7 eraseSimTime() [2/2]

```
WossCreator & woss::WossCreator::eraseSimTime (
    Location *const tx = CCSimTime::ALL_LOCATIONS,
    Location *const rx = CCSimTime::ALL_LOCATIONS ) [inline]
```

Erases the SimTime for given transmitter, receiver woss::Location

Parameters

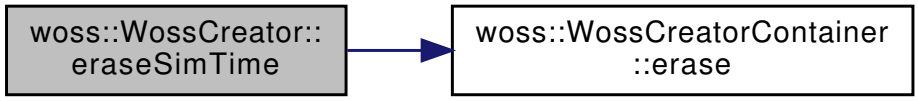
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccsimtime_map, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.8 eraseTotalRuns() [1/2]

```
WossCreator & woss::WossCreator::eraseTotalRuns (
    const CoordZ & tx,
    const CoordZ & rx ) [inline]
```

Erases the total number of channel simulator's runs

Parameters

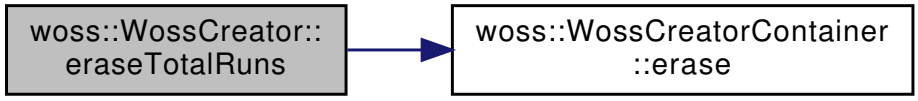
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_runs, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.9 eraseTotalRuns() [2/2]

```
WossCreator & woss::WossCreator::eraseTotalRuns (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS ) [inline]
```

Erases the total number of channel simulator's runs

Parameters

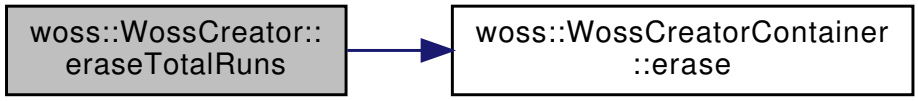
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_runs, and woss::WossCreatorContainer< Data >::erase().

Here is the call graph for this function:



13.84.4.10 getEvolutionTimeQuantum() [1/2]

```
double woss::WossCreator::getEvolutionTimeQuantum (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the time evolution threshold

Parameters

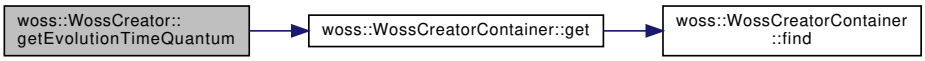
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

time evolution threshold in seconds

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.84.4.11 getEvolutionTimeQuantum() [2/2]

```
double woss::WossCreator::getEvolutionTimeQuantum (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS ) const  [inline]
```

Returns the time evolution threshold

Parameters

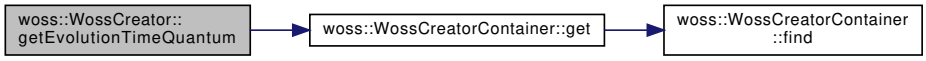
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

time evolution threshold in seconds

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:



13.84.4.12 getFrequencyStep() [1/2]

```
double woss::WossCreator::getFrequencyStep (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the frequency step for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

frequency step in Hz

References ccfrequency_step, and woss::WossCreatorContainer< Data >::get().

Referenced by woss::BellhopCreator::createWoss(), woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT
woss::WossManagerResDb::getWossTimeArr(), and woss::WossManagerResDbMT::getWossTimeArr().

Here is the call graph for this function:

13.84.4.13 getFrequencyStep() [2/2]

```
double woss::WossCreator::getFrequencyStep (
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS ) const  [inline]
```

Returns the frequency step for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

frequency step in Hz

References ccfrequency_step, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.84.4.14 getSimTime() [1/2]

```
SimTime woss::WossCreator::getSimTime (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the SimTime for given transmitter, receiver woss::CoordZ

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

SimTime instance

References ccsimtime_map, and woss::WossCreatorContainer< Data >::get().

Referenced by woss::BellhopCreator::createWoss(), woss::WossManagerResDbMT::getWossPressure(), woss::WossManager::getWossPressure(), woss::WossManagerResDbMT::getWossTimeArr(), woss::WossManager::getWossTimeA woss::WMSMTcreateThreadPressure(), and woss::WMSMTcreateThreadTimeArr().

Here is the call graph for this function:

13.84.4.15 getSimTime() [2/2]

```
SimTime woss::WossCreator::getSimTime (
    Location *const tx = CCSimTime::ALL_LOCATIONS,
    Location *const rx = CCSimTime::ALL_LOCATIONS ) const  [inline]
```

Returns the SimTime for given transmitter, receiver woss::Location

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

SimTime instance

References ccsimtime_map, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.84.4.16 getTotalRuns() [1/2]

```
int woss::WossCreator::getTotalRuns (
    const CoordZ & tx,
    const CoordZ & rx ) const  [inline]
```

Returns the total number of channel simulator's runs

Parameters

tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

number of runs

References cctotal_runs, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.84.4.17 getTotalRuns() [2/2]

```
int woss::WossCreator::getTotalRuns (
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS ) const  [inline]
```

Returns the total number of channel simulator's runs

Parameters

tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

number of runs

References cctotal_runs, and woss::WossCreatorContainer< Data >::get().

Here is the call graph for this function:

13.84.4.18 getWrkDirPath() `::std::string woss::WossCreator::getWrkDirPath ( ) const [inline]`

Gets the work pathname

Returns

valid pathname

References work_dir_path.

13.84.4.19 initializeWoss() `bool WossCreator::initializeWoss ( Woss *const woss_ptr ) const [protected], [pure virtual]`

Initializes given Woss object

Parameters

woss_ptr	const pointer to an unitialized Woss

Returns

true if method succeeded, *false* otherwise

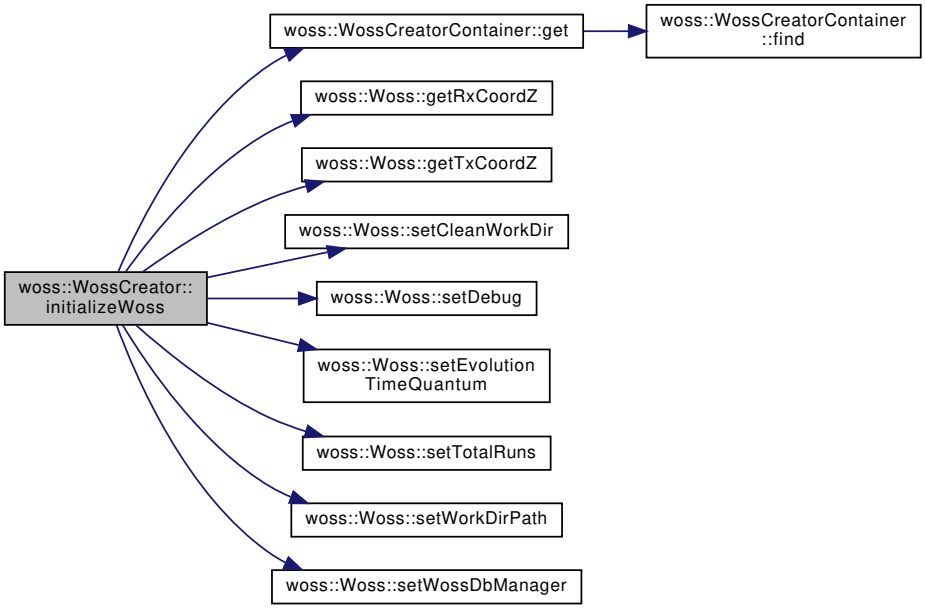
Implemented in woss::BellhopCreator.

References ccevolution_time_quantum, cctotal_runs, woss::WossCreatorContainer< Data >::get(), woss::Woss::getRxCoordZ(), woss::Woss::getTxCoordZ(), woss::Woss::setCleanWorkDir(), woss::Woss::setDebug(), woss::Woss::setEvolutionTimeQuantum(),

woss::Woss::setTotalRuns(), woss::Woss::setWorkDirPath(), woss::Woss::setWossDbManager(), work_dir_path, woss_clean_workdir, and woss_debug.

Referenced by woss::BellhopCreator::initializeWoss().

Here is the call graph for this function:



13.84.4.20 setCleanWorkDir() `WossCreator & woss::WossCreator::setCleanWorkDir ( bool flag ) [inline]`

Sets clean work dir flag

Parameters

flag	debug flag

Returns

reference to ∗**this**

References woss_clean_workdir.

13.84.4.21 setDebug() `WossCreator & woss::WossCreator::setDebug ( bool flag ) [inline]`

Sets debug flag

Parameters

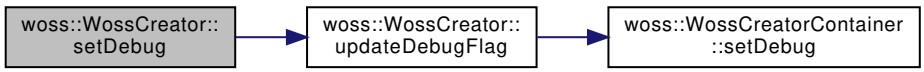
flag	debug flag

Returns

reference to ∗**this**

References debug, and updateDebugFlag().

Here is the call graph for this function:



13.84.4.22 setEvolutionTimeQuantum() [1/2]

```
WossCreator & woss::WossCreator::setEvolutionTimeQuantum (
            double value,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the time evolution threshold in seconds

Parameters

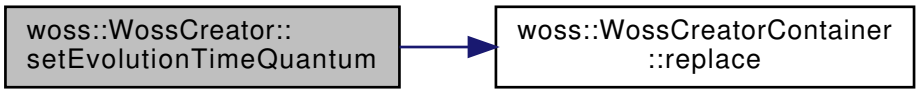
value	time evolution threshold in seconds
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.23 setEvolutionTimeQuantum() [2/2]

```
WossCreator & woss::WossCreator::setEvolutionTimeQuantum (
            double value,
            Location *const tx = CCInt::ALL_LOCATIONS,
            Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets time threshold for time evolution purposes

Parameters

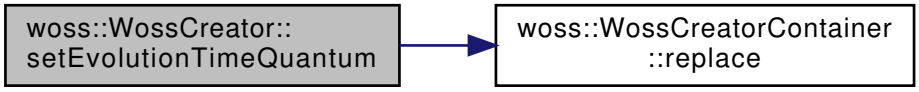
value	time threshold in seconds
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccevolution_time_quantum, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.24 setFrequencyStep() [1/2]

```
WossCreator & woss::WossCreator::setFrequencyStep (
            double f_step,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Sets the frequency step for given transmitter, receiver woss::CoordZ

Parameters

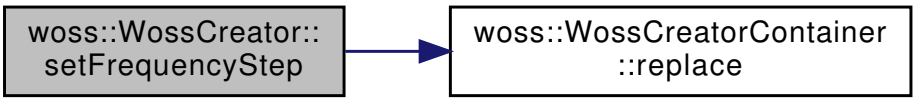
f_step	valid frequency step [Hz]. If step <= 0 start frequency is used and no steps are done
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccfrequency_step, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.25 setFrequencyStep() [2/2]

```
WossCreator & woss::WossCreator::setFrequencyStep (
    double f_step,
    Location *const tx = CCDouble::ALL_LOCATIONS,
    Location *const rx = CCDouble::ALL_LOCATIONS )  [inline]
```

Sets the frequency step for given transmitter, receiver woss::Location

Parameters

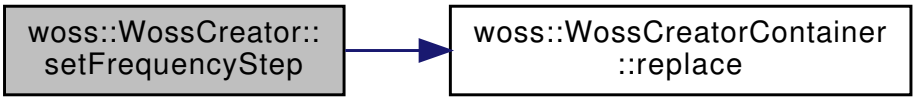
f_step	valid frequency step [Hz]. If step <= 0 start frequency is used and no steps are done
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccfrequency_step, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.26 setSimTime() [1/2]

```
WossCreator & woss::WossCreator::setSimTime (
    const SimTime & simtime,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the SimTime for given transmitter, receiver woss::CoordZ

Parameters

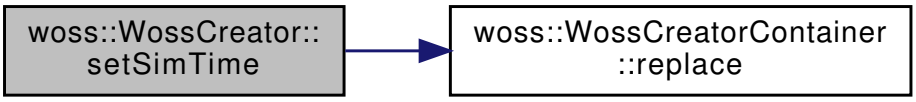
simtime	const reference to a valid SimTime
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References ccsimtime_map, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.27 setSimTime() [2/2]

```
WossCreator & woss::WossCreator::setSimTime (
    const SimTime & simtime,
    Location *const tx = CCSimTime::ALL_LOCATIONS,
    Location *const rx = CCSimTime::ALL_LOCATIONS )  [inline]
```

Sets the SimTime for given transmitter, receiver woss::Location

Parameters

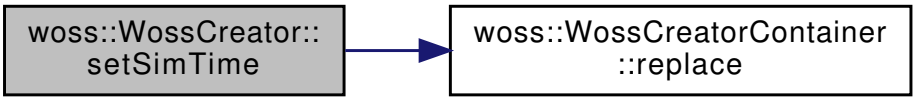
simtime	const reference to a valid SimTime
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References ccsimtime_map, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.28 setTotalRuns() [1/2]

```
WossCreator & woss::WossCreator::setTotalRuns (
    int runs,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Sets the total number of channel simulator's runs

Parameters

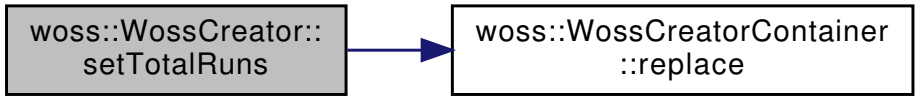
runs	number of runs
tx	const reference to a valid woss::CoordZ instance
rx	const reference to a valid woss::CoordZ instance

Returns

reference to ∗**this**

References cctotal_runs, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.29 setTotalRuns() [2/2]

```
WossCreator & woss::WossCreator::setTotalRuns (
    int runs,
    Location *const tx = CCInt::ALL_LOCATIONS,
    Location *const rx = CCInt::ALL_LOCATIONS )  [inline]
```

Sets the total number of channel simulator's runs

Parameters

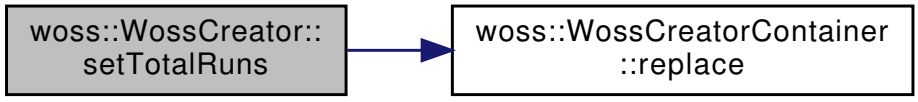
runs	number of runs
tx	const pointer to a valid woss::Location instance
rx	const pointer to a valid woss::Location instance

Returns

reference to ∗**this**

References cctotal_runs, and woss::WossCreatorContainer< Data >::replace().

Here is the call graph for this function:



13.84.4.30 setTransducerHandler()

```
WossCreator & woss::WossCreator::setTransducerHandler (
        const TransducerHandler *const ptr )  [inline]
```

Sets the WossDbManager pointer

Parameters

path	const pointer to a const WossDbManager object

Returns

reference to ∗**this**

Referenced by woss::WossController::initialize().

13.84.4.31 setWossDbManager() `WossCreator & woss::WossCreator::setWossDbManager ( const WossDbManager *const ptr ) [inline]`

Sets the WossDbManager pointer

Parameters

path	const pointer to a const WossDbManager object

Returns

reference to ∗**this**

Referenced by woss::WossController::initialize().

13.84.4.32 setWossDebug() `WossCreator & woss::WossCreator::setWossDebug ( bool flag ) [inline]`

Sets debug flag of every Woss object created

Parameters

flag	debug flag

Returns

reference to ∗**this**

References woss_debug.

13.84.4.33 setWrkDirPath() `WossCreator & woss::WossCreator::setWrkDirPath ( const ::std::string & path ) [inline]`

Sets the work pathname

Parameters

path	valid pathname

Returns

reference to ∗**this**

References work_dir_path.

13.84.4.34 updateDebugFlag() `void WossCreator::updateDebugFlag ( ) [protected], [virtual]`

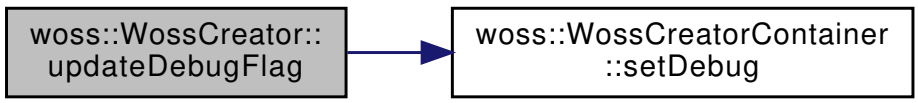
Propagates the debug flag

Reimplemented in woss::BellhopCreator.

References ccevolution_time_quantum, ccfrequency_step, ccsimtime_map, cctotal_runs, debug, and woss::WossCreatorContainer<

Referenced by setDebug(), woss::BellhopCreator::updateDebugFlag(), and WossCreator().

Here is the call graph for this function:



13.84.4.35 usingDebug() `bool woss::WossCreator::usingDebug ( ) const [inline]`

Returns debug flag

Returns

true if using debug, *false* otherwise

References debug.

13.84.4.36 usingWossDebug() `bool woss::WossCreator::usingWossDebug ( ) const [inline]`

Returns Woss debug flag

Returns

true if Woss objects will be using debug, *false* otherwise

References woss_debug.

13.84.5 Member Data Documentation

13.84.5.1 ccevolution_time_quantum `CCDouble woss::WossCreator::ccevolution_time_quantum [protected]`

Time evolution threshold in seconds

Referenced by eraseEvolutionTimeQuantum(), getEvolutionTimeQuantum(), initializeWoss(), setEvolutionTimeQuantum(), and updateDebugFlag().

13.84.5.2 ccfrequency_step `CCDouble woss::WossCreator::ccfrequency_step [protected]`

Frequency step [Hz]

Referenced by eraseFrequencyStep(), getFrequencyStep(), setFrequencyStep(), updateDebugFlag(), and WossCreator().

13.84.5.3 ccsimtime_map `CCSimTime woss::WossCreator::ccsimtime_map [protected]`

SimTime containter for user-given transmitter CoordZ

Referenced by eraseSimTime(), getSimTime(), setSimTime(), and updateDebugFlag().

13.84.5.4 cctotal_runs `CCInt woss::WossCreator::cctotal_runs [protected]`

Total number of channel simulator's runs

Referenced by eraseTotalRuns(), getTotalRuns(), initializeWoss(), setTotalRuns(), and updateDebugFlag().

13.84.5.5 debug `bool woss::WossCreator::debug [protected]`

Debug flag

Referenced by setDebug(), woss::BellhopCreator::updateDebugFlag(), updateDebugFlag(), and usingDebug().

13.84.5.6 work_dir_path `::std::string woss::WossCreator::work_dir_path [protected]`

Directory path for temporary files (e.g. channel simulator files)

Referenced by getWrkDirPath(), initializeWoss(), and setWrkDirPath().

13.84.5.7 woss_clean_workdir `bool woss::WossCreator::woss_clean_workdir [protected]`

Debug flag for all Woss instances

Referenced by initializeWoss(), and setCleanWorkDir().

13.84.5.8 woss_debug `bool woss::WossCreator::woss_debug [protected]`

Debug flag for all Woss instances

Referenced by initializeWoss(), setWossDebug(), and usingWossDebug().

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

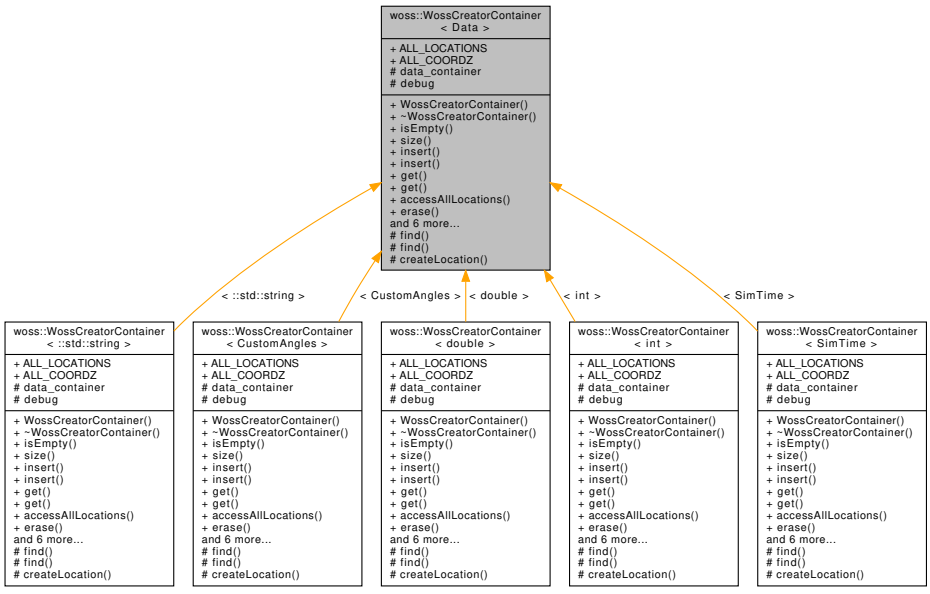
- woss/woss-creator.h
- woss/woss-creator.cpp

13.85 woss::WossCreatorContainer< Data > Class Template Reference

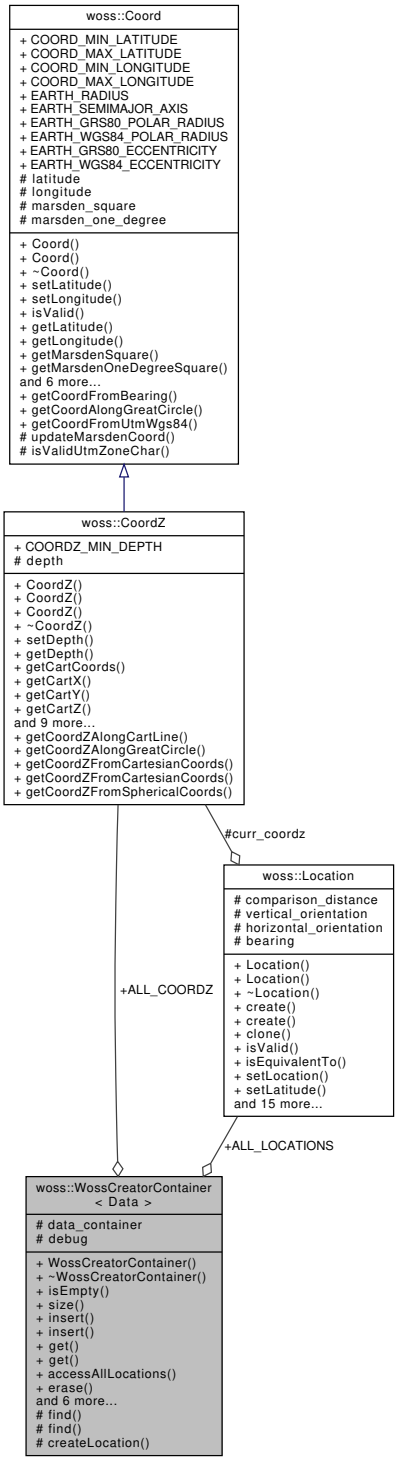
Class that stores WossCreator parameters.

`#include <woss-creator-container.h>`

Inheritance diagram for woss::WossCreatorContainer< Data >:



Collaboration diagram for woss::WossCreatorContainer< Data >:



Public Member Functions

- WossCreatorContainer ()
- ~WossCreatorContainer ()
- bool isEmpty () const
- int size () const
- bool insert (const Data &data, Location *const tx, Location *const rx)

- bool insert (const Data &data, const CoordZ &tx, const CoordZ &rx)
- Data get (Location *const tx, Location *const rx) const
- Data get (const CoordZ &tx, const CoordZ &rx) const
- Data & accessAllLocations ()
- void erase (Location *const tx, Location *const rx)
- void erase (const CoordZ &tx, const CoordZ &rx)
- void replace (const Data &data, Location *const tx, Location *const rx)
- void replace (const Data &data, const CoordZ &tx, const CoordZ &rx)
- void clear ()
- void setDebug (bool flag)
- bool isUsingDebug () const

Static Public Attributes

- static Location *const ALL_LOCATIONS = NULL
- static const CoordZ ALL_COORDZ = CoordZ()

Protected Types

- typedef ::std::map< Location *, Data > InnerContainer
- typedef InnerContainer::iterator **ICIter**
- typedef InnerContainer::reverse_iterator **ICRIter**
- typedef InnerContainer::const_iterator **ICCIter**
- typedef InnerContainer::const_reverse_iterator **ICCRIter**
- typedef ::std::map< Location *, InnerContainer > DataContainer
- typedef DataContainer::iterator **DCIter**
- typedef DataContainer::const_iterator **DCCIter**
- typedef DataContainer::reverse_iterator **DCRIter**
- typedef DataContainer::const_reverse_iterator **DCRCIter**

Protected Member Functions

- DCIter find (const CoordZ &coordinates)
- ICIter find (const CoordZ &coordinates, const DCIter &iter)
- Location * createLocation (const CoordZ &coordinates)

Protected Attributes

- DataContainer data_container
- bool debug

13.85.1 Detailed Description

template<typename Data>
class woss::WossCreatorContainer< Data >

Class that stores WossCreator parameters.

WossCreatorContainer provides interface for storing and retrieving custom WossCreator data, indexed by transmitter and receiver woss::Location*

13.85.2 Member Typedef Documentation

13.85.2.1 DataContainer `template<typename Data >`

`typedef ::std::map< Location*, InnerContainer > woss::WossCreatorContainer< Data >::Data↩ Container [protected]`

The outer container. It associates a transmitter Location pointer to a InnerContainer object

13.85.2.2 InnerContainer `template<typename Data >`

`typedef ::std::map< Location*, Data > woss::WossCreatorContainer< Data >::InnerContainer [protected]`

The innermost container. It associates a receiver Location pointer to the Data object

13.85.3 Constructor & Destructor Documentation

13.85.3.1 WossCreatorContainer() `template<typename Data >`

`woss::WossCreatorContainer< Data >::WossCreatorContainer`

Default constructor

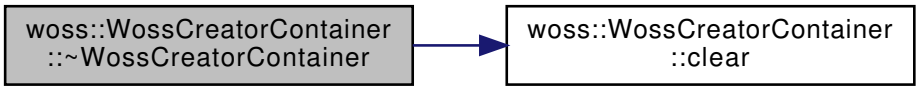
13.85.3.2 ~WossCreatorContainer() `template<typename Data >`

`woss::WossCreatorContainer< Data >::~WossCreatorContainer`

Destructor

References woss::WossCreatorContainer< Data >::clear().

Here is the call graph for this function:



13.85.4 Member Function Documentation

13.85.4.1 accessAllLocations() template<typename Data >

```
Data & woss::WossCreatorContainer< Data >::accessAllLocations  [inline]
```

Returns a reference to the Data object associated to transmitter and receiver Location equal to ALL_LOCATIONS. If no keys were present, the Data object is default constructed and a reference to it is returned

Returns

a reference to a Data object

Referenced by woss::WossCreator::WossCreator().

13.85.4.2 clear() template<typename Data >

```
void woss::WossCreatorContainer< Data >::clear  [inline]
```

Clears the container

Referenced by woss::WossCreatorContainer< Data >::~WossCreatorContainer().

13.85.4.3 createLocation() template<typename Data >

```
Location * woss::WossCreatorContainer< Data >::createLocation (
            const CoordZ & coordinates )  [inline], [protected]
```

Creates a Location for a given CoordZ

Parameters

coordinates	a const reference to a valid receiver CoordZ

Returns

a heap-allocated Location pointer

Referenced by woss::WossCreatorContainer< CustomTransducer >::insert(), woss::WossCreatorContainer< Data * >::insert(), woss::WossCreatorContainer< CustomTransducer >::replace(), and woss::WossCreatorContainer< Data * >::replace().

13.85.4.4 erase() [1/2] template<typename Data >

```
void woss::WossCreatorContainer< Data >::erase (
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Erases an object into the container

Parameters

tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

13.85.4.5 erase() [2/2] template<typename Data >

```
void woss::WossCreatorContainer< Data >::erase (
            Location *const tx,
            Location *const rx )  [inline]
```

Erases an object into the container

Parameters

tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Referenced by woss::BellhopCreator::eraseAltimetryType(), woss::BellhopCreator::eraseAngles(), woss::BellhopCreator::eraseBathyr
woss::BellhopCreator::eraseBathymetryType(), woss::BellhopCreator::eraseBeamOptions(), woss::BellhopCreator::eraseBhMode(),
woss::BellhopCreator::eraseBoxDepth(), woss::BellhopCreator::eraseBoxRange(), woss::BellhopCreator::eraseCustomTransducer(),
woss::WossCreator::eraseEvolutionTimeQuantum(), woss::WossCreator::eraseFrequencyStep(), woss::BellhopCreator::eraseRaysN
woss::BellhopCreator::eraseRxMaxDepthOffset(), woss::BellhopCreator::eraseRxMaxRangeOffset(), woss::BellhopCreator::eraseRx
woss::BellhopCreator::eraseRxMinRangeOffset(), woss::BellhopCreator::eraseRxTotalDepths(), woss::BellhopCreator::eraseRxTotall
woss::WossCreator::eraseSimTime(), woss::BellhopCreator::eraseSspDepthPrecision(), woss::BellhopCreator::eraseSspDepthSteps
woss::BellhopCreator::eraseTotalRangeSteps(), woss::WossCreator::eraseTotalRuns(), woss::BellhopCreator::eraseTotalTransmitters
woss::BellhopCreator::eraseTxMaxDepthOffset(), and woss::BellhopCreator::eraseTxMinDepthOffset().

13.85.4.6 find() [1/2] template<typename Data >

```
WossCreatorContainer< Data >::DCIter woss::WossCreatorContainer< Data >::find (
            const CoordZ & coordinates )  [protected]
```

Returns an DataContainer iterators that points to the given transmitter coordinates. An iterator to the end of the DataContainer is returned if no coordinates are found

Parameters

coordinates	a const reference to a valid transmitter CoordZ

Returns

a DataContainer::iterator

References woss::WossCreatorContainer< Data >::ALL_COORDZ, woss::WossCreatorContainer< Data >::ALL_LOCATIONS, woss::WossCreatorContainer< Data >::data_container, and woss::WossCreatorContainer< Data >::debug.

Referenced by woss::WossCreatorContainer< Data >::get(), woss::WossCreatorContainer< Data * >::get(), woss::WossCreatorContainer< CustomTransducer >::insert(), and woss::WossCreatorContainer< Data * >::insert().

13.85.4.7 find() [2/2] template<typename Data >

```
WossCreatorContainer< Data >::ICIter woss::WossCreatorContainer< Data >::find (
            const CoordZ & coordinates,
            const DCIter & iter )  [protected]
```

Returns an InnerContainer iterators for the DataContainter iterator that points to the given receiver coordinates. An iterator to the relative end of the InnerContainer is returned if no coordinates are found

Parameters

coordinates	a const reference to a valid receiver CoordZ

Returns

a DataContainer::iterator

References woss::WossCreatorContainer< Data >::ALL_COORDZ, woss::WossCreatorContainer< Data >::ALL_LOCATIONS, and woss::WossCreatorContainer< Data >::debug.

13.85.4.8 get() [1/2] template<typename Data >

```
Data woss::WossCreatorContainer< Data >::get (
            const CoordZ & tx,
            const CoordZ & rx ) const  [inline]
```

Returns an object into the container. If the keys are NOT found, a default constructed object is returned

Parameters

tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

a copy of the original object

References woss::WossCreatorContainer< Data >::ALL_COORDZ, woss::WossCreatorContainer< Data >::data_container, and woss::WossCreatorContainer< Data >::debug.

13.85.4.9 get() [2/2] template<typename Data >

```
Data woss::WossCreatorContainer< Data >::get (
            Location *const tx,
            Location *const rx ) const  [inline]
```

Returns an object into the container. If the keys are NOT found, a default constructed object is returned

Parameters

tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

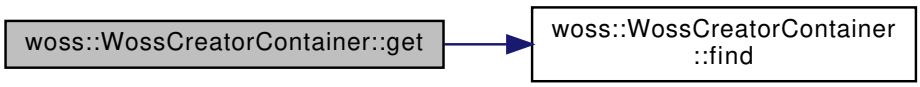
Returns

a copy of the original object

References woss::WossCreatorContainer< Data >::ALL_LOCATIONS, woss::WossCreatorContainer< Data >::data_container, woss::WossCreatorContainer< Data >::debug, and woss::WossCreatorContainer< Data >::find().

Referenced by woss::BellhopCreator::getAltimetryType(), woss::BellhopCreator::getAngles(), woss::BellhopCreator::getBathymetryMe
woss::BellhopCreator::getBathymetryType(), woss::BellhopCreator::getBeamOptions(), woss::BellhopCreator::getBhMode(),
woss::BellhopCreator::getBoxDepth(), woss::BellhopCreator::getBoxRange(), woss::WossCreator::getEvolutionTimeQuantum(),
woss::WossCreator::getFrequencyStep(), woss::BellhopCreator::getRaysNumber(), woss::BellhopCreator::getRxMaxDepthOffset(),
woss::BellhopCreator::getRxMaxRangeOffset(), woss::BellhopCreator::getRxMinDepthOffset(), woss::BellhopCreator::getRxMinRang
woss::BellhopCreator::getRxTotalDepths(), woss::BellhopCreator::getRxTotalRanges(), woss::WossCreator::getSimTime(),
woss::BellhopCreator::getSspDepthPrecision(), woss::BellhopCreator::getSspDepthSteps(), woss::BellhopCreator::getTotalRangeSte
woss::WossCreator::getTotalRuns(), woss::BellhopCreator::getTotalTransmitters(), woss::BellhopCreator::getTxMaxDepthOffset(),
woss::BellhopCreator::getTxMinDepthOffset(), woss::BellhopCreator::initializeBhWoss(), and woss::WossCreator::initializeWoss().

Here is the call graph for this function:



13.85.4.10 insert() [1/2] template<typename Data >

```
bool woss::WossCreatorContainer< Data >::insert (
            const Data & data,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted

Parameters

data	const reference to a Data object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

13.85.4.11 insert() [2/2] template<typename Data >

```
bool woss::WossCreatorContainer< Data >::insert (
            const Data & data,
            Location *const tx,
            Location *const rx )  [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted

Parameters

data	const reference to a Data object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

13.85.4.12 isEmpty() template<typename Data >

```
bool woss::WossCreatorContainer< Data >::isEmpty  [inline]
```

Checks if the container is empty

Returns

true if it's empty, *false* otherwise

Referenced by woss::BellhopCreator::initializeBhWoss().

13.85.4.13 isUsingDebug() template<typename Data >

```
bool woss::WossCreatorContainer< Data >::isUsingDebug ( ) const  [inline]
```

Retuns the debug flag

Returns

true if is using the debug flag, *false* otherwise

References woss::WossCreatorContainer< Data >::debug.

13.85.4.14 replace() [1/2] template<typename Data >

```
void woss::WossCreatorContainer< Data >::replace (
            const Data & data,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Replaces an object into the container. If the key are not present, the object is still inserted

Parameters

data	const reference to a Data object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

13.85.4.15 replace() [2/2] template<typename Data >

```
void woss::WossCreatorContainer< Data >::replace (
            const Data & data,
            Location *const tx,
            Location *const rx )  [inline]
```

Replaces an object into the container. If the key are not present, the object is still inserted

Parameters

data	const reference to a Data object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

Referenced by woss::BellhopCreator::setAltimetryType(), woss::BellhopCreator::setAngles(), woss::BellhopCreator::setBathymetryMe
woss::BellhopCreator::setBathymetryType(), woss::BellhopCreator::setBeamOptions(), woss::BellhopCreator::setBhMode(),
woss::BellhopCreator::setBoxDepth(), woss::BellhopCreator::setBoxRange(), woss::WossCreator::setEvolutionTimeQuantum(),
woss::WossCreator::setFrequencyStep(), woss::BellhopCreator::setRaysNumber(), woss::BellhopCreator::setRxMaxDepthOffset(),
woss::BellhopCreator::setRxMaxRangeOffset(), woss::BellhopCreator::setRxMinDepthOffset(), woss::BellhopCreator::setRxMinRang
woss::BellhopCreator::setRxTotalDepths(), woss::BellhopCreator::setRxTotalRanges(), woss::WossCreator::setSimTime(),
woss::BellhopCreator::setSspDepthPrecision(), woss::BellhopCreator::setSspDepthSteps(), woss::BellhopCreator::setTotalRangeSte
woss::WossCreator::setTotalRuns(), woss::BellhopCreator::setTotalTransmitters(), woss::BellhopCreator::setTxMaxDepthOffset(),
and woss::BellhopCreator::setTxMinDepthOffset().

13.85.4.16 setDebug() template<typename Data >

```
void woss::WossCreatorContainer< Data >::setDebug (
            bool flag )  [inline]
```

Sets or unsets the debug flag

Parameters

flag	boolean flag

References woss::WossCreatorContainer< Data >::debug.

Referenced by woss::BellhopCreator::updateDebugFlag(), and woss::WossCreator::updateDebugFlag().

13.85.4.17 size() template<typename Data >

int woss::WossCreatorContainer< Data >::size [inline]

Returns the size of the container

Returns

container's size

13.85.5 Member Data Documentation

13.85.5.1 ALL_COORDZ template<typename Data >

const CoordZ woss::WossCreatorContainer< Data >::ALL_COORDZ = CoordZ() [static]

Pointer that represents the index for all possible coordinates

Referenced by woss::WossCreatorContainer< Data >::find(), woss::WossCreatorContainer< Data >::get(), and woss::WossCreatorContainer< Data * >::get().

13.85.5.2 ALL_LOCATIONS template<typename Data >

Location *const woss::WossCreatorContainer< Data >::ALL_LOCATIONS = NULL [static]

Pointer that represents the index for all possible Locations

Referenced by woss::WossCreatorContainer< Data * >::accessAllLocations(), woss::WossCreatorContainer< CustomTransducer >
woss::WossCreatorContainer< Data >::find(), woss::WossCreatorContainer< Data >::get(), and woss::WossCreatorContainer< Da

13.85.5.3 data_container template<typename Data >

DataContainer woss::WossCreatorContainer< Data >::data_container [protected]

The Data container

Referenced by woss::WossCreatorContainer< Data * >::accessAllLocations(), woss::WossCreatorContainer< CustomTransducer >
woss::WossCreatorContainer< Data >::find(), woss::WossCreatorContainer< Data >::get(), woss::WossCreatorContainer< Data *
woss::WossCreatorContainer< CustomTransducer >::insert(), woss::WossCreatorContainer< Data * >::insert(),
woss::WossCreatorContainer< CustomTransducer >::replace(), and woss::WossCreatorContainer< Data * >::replace().

13.85.5.4 debug template<typename Data >

bool woss::WossCreatorContainer< Data >::debug [protected]

Debug flag

Referenced by woss::WossCreatorContainer< Data >::find(), woss::WossCreatorContainer< Data >::get(),
woss::WossCreatorContainer< Data * >::get(), woss::WossCreatorContainer< Data >::isUsingDebug(), and
woss::WossCreatorContainer< Data >::setDebug().

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

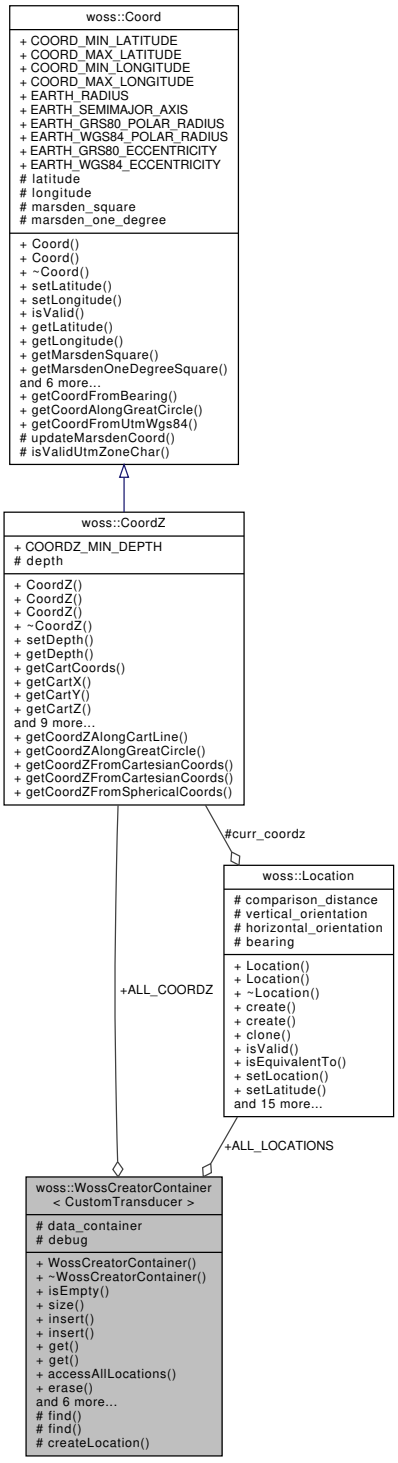
- woss/woss-creator-container.h
- woss/woss-creator-container.cpp

13.86 woss::WossCreatorContainer< CustomTransducer > Class Reference

Full specialization for woss::CustomTransducer.

`#include <woss-creator-container.h>`

Collaboration diagram for woss::WossCreatorContainer< CustomTransducer >:



Public Member Functions

- bool **isEmpty** () const
- int **size** () const
- bool insert (const CustomTransducer &data, Location *const tx, Location *const rx)
- bool insert (const CustomTransducer &data, const CoordZ &tx, const CoordZ &rx)
- CustomTransducer get (Location *const tx, Location *const rx) const
- CustomTransducer get (const CoordZ &tx, const CoordZ &rx) const
- CustomTransducer & accessAllLocations ()
- void **erase** (Location *const tx, Location *const rx)
- void **erase** (const CoordZ &tx, const CoordZ &rx)
- void replace (const CustomTransducer &data, Location *const tx, Location *const rx)
- void replace (const CustomTransducer &data, const CoordZ &tx, const CoordZ &rx)
- void **clear** ()
- void **setDebug** (bool flag)
- bool **isUsingDebug** () const

Static Public Attributes

- static Location *const **ALL_LOCATIONS**
- static const CoordZ **ALL_COORDZ**

Protected Types

- typedef ::std::map< Location *, CustomTransducer > **InnerContainer**
- typedef InnerContainer::iterator **ICIter**
- typedef InnerContainer::reverse_iterator **ICRIter**
- typedef InnerContainer::const_iterator **ICCIter**
- typedef InnerContainer::const_reverse_iterator **ICCRIter**
- typedef ::std::map< Location *, InnerContainer > **DataContainer**
- typedef DataContainer::iterator **DCIter**
- typedef DataContainer::const_iterator **DCCIter**
- typedef DataContainer::reverse_iterator **DCRIter**
- typedef DataContainer::const_reverse_iterator **DCRCIter**

Protected Member Functions

- DCIter **find** (const CoordZ &coordinates)
- ICIter **find** (const CoordZ &coordinates, const DCIter &iter)
- Location * **createLocation** (const CoordZ &coordinates)

Protected Attributes

- DataContainer **data_container**
- bool **debug**

13.86.1 Detailed Description

Full specialization for woss::CustomTransducer.

Full specialization for woss::CustomTransducer

13.86.2 Member Function Documentation

13.86.2.1 accessAllLocations() `CustomTransducer & woss::WossCreatorContainer< CustomTransducer >::accessAllLocations ( ) [inline]`

Returns a reference to the CustomTransducer associated to transmitter and receiver Location equal to ALL_↩ LOCATIONS. If no keys were present, the CustomTransducer is defaulted constructed and a reference to it is returned. The user has to properly delete the pointer prior changing it

Returns

a reference to a Data pointer

References woss::WossCreatorContainer< Data >::ALL_LOCATIONS, and woss::WossCreatorContainer< Data >::data_container.

Referenced by woss::BellhopCreator::BellhopCreator().

13.86.2.2 get() [1/2]

```
CustomTransducer woss::WossCreatorContainer< CustomTransducer >::get (
    const CoordZ & tx,
    const CoordZ & rx ) const
```

Returns a CustomTransducer pointer for given keys. If the keys are not found a NULL pointer is returned

Parameters

tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

a pointer to a rotated CustomTransducer object

13.86.2.3 get() [2/2]

```
CustomTransducer woss::WossCreatorContainer< CustomTransducer >::get (
    Location *const tx,
    Location *const rx ) const
```

Returns a CustomTransducer for given keys. If the keys are not found a NULL pointer is returned

Parameters

tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

a pointer to a rotated CustomTransducer object

Referenced by woss::BellhopCreator::getCustomTransducer(), and woss::BellhopCreator::initializeBhWoss().

13.86.2.4 insert() [1/2]

```
bool woss::WossCreatorContainer< CustomTransducer >::insert (
    const CustomTransducer & data,
    const CoordZ & tx,
    const CoordZ & rx ) [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted and it is deleted

Parameters

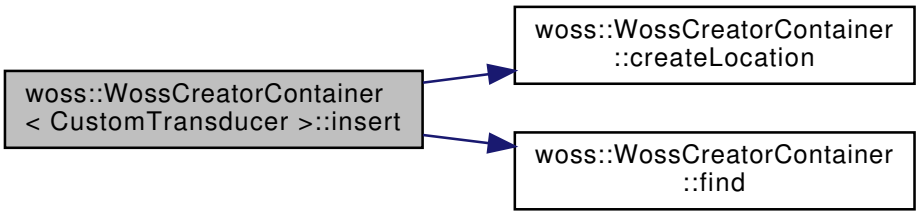
data	const pointer to a CustomTransducer object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::createLocation(), woss::WossCreatorContainer< Data >::data_container, and woss::WossCreatorContainer< Data >::find().

Here is the call graph for this function:



13.86.2.5 insert() [2/2]

```
bool woss::WossCreatorContainer< CustomTransducer >::insert (
    const CustomTransducer & data,
    Location *const tx,
    Location *const rx ) [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted and it is deleted

Parameters

data	const reference to a CustomTransducer object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::data_container.

13.86.2.6 replace() [1/2]

```
void woss::WossCreatorContainer< CustomTransducer >::replace (
    const CustomTransducer & data,
    const CoordZ & tx,
    const CoordZ & rx )  [inline]
```

Replaces a pointer into the container. If the keys are already present, the previous object is deleted

Parameters

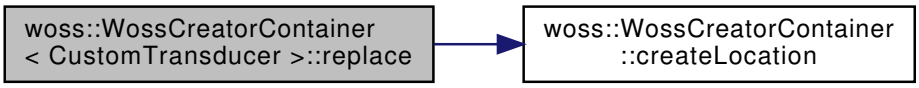
data	const pointer to a CustomTransducer object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::createLocation(), and woss::WossCreatorContainer< Data >::data_container.

Here is the call graph for this function:



13.86.2.7 replace() [2/2]

```
void woss::WossCreatorContainer< CustomTransducer >::replace (
    const CustomTransducer & data,
    Location *const tx,
    Location *const rx )  [inline]
```

Replaces a pointer into the container. If the keys are already present, the previous object is deleted

Parameters

data	const reference to a CustomTransducer object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::data_container.

Referenced by woss::BellhopCreator::setCustomTransducer().

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

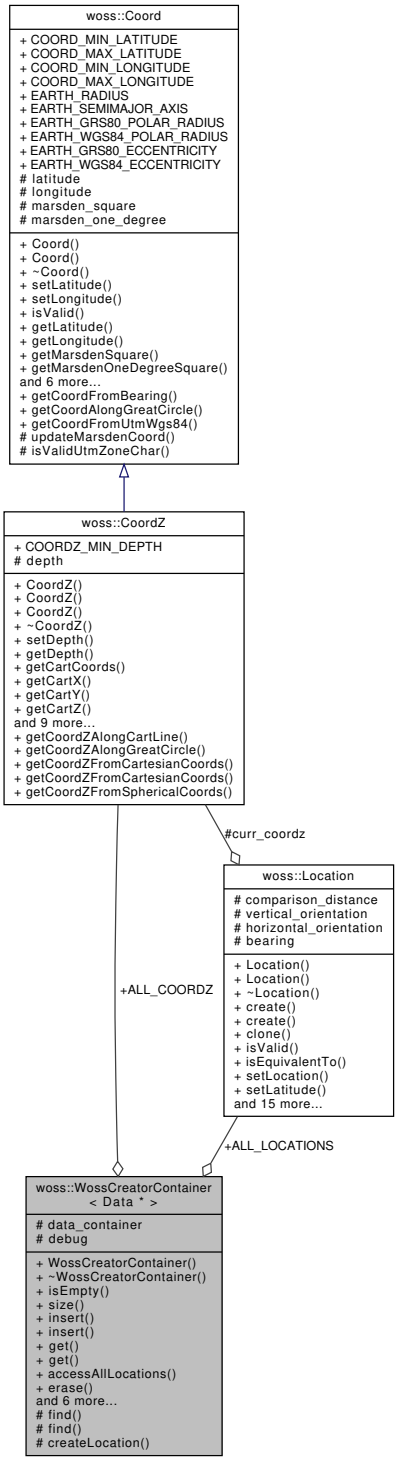
- woss/woss-creator-container.h

13.87 woss::WossCreatorContainer< Data * > Class Template Reference

Partial specialization for pointers.

`#include <woss-creator-container.h>`

Collaboration diagram for woss::WossCreatorContainer< Data * >:



Public Member Functions

- bool **isEmpty** () const
- int **size** () const
- bool insert (Data *data, Location *const tx, Location *const rx)
- bool insert (Data *data, const CoordZ &tx, const CoordZ &rx)
- Data * get (Location *const tx, Location *const rx) const

- Data * get (const CoordZ &tx, const CoordZ &rx) const
- Data *& accessAllLocations ()
- void **erase** (Location *const tx, Location *const rx)
- void **erase** (const CoordZ &tx, const CoordZ &rx)
- void replace (Data *const data, Location *const tx, Location *const rx)
- void replace (Data *const data, const CoordZ &tx, const CoordZ &rx)
- void **clear** ()
- void **setDebug** (bool flag)
- bool **isUsingDebug** () const

Static Public Attributes

- static Location *const **ALL_LOCATIONS** = NULL
- static const CoordZ **ALL_COORDZ** = CoordZ()

Protected Types

- typedef ::std::map< Location *, Data * > **InnerContainer**
- typedef InnerContainer::iterator **ICIter**
- typedef InnerContainer::reverse_iterator **ICRIter**
- typedef InnerContainer::const_iterator **ICCIter**
- typedef InnerContainer::const_reverse_iterator **ICCRIter**
- typedef ::std::map< Location *, InnerContainer > **DataContainer**
- typedef DataContainer::iterator **DCIter**
- typedef DataContainer::const_iterator **DCCIter**
- typedef DataContainer::reverse_iterator **DCRIter**
- typedef DataContainer::const_reverse_iterator **DCRCIter**

Protected Member Functions

- DCIter **find** (const CoordZ &coordinates)
- ICIter **find** (const CoordZ &coordinates, const DCIter &iter)
- Location * **createLocation** (const CoordZ &coordinates)

Protected Attributes

- DataContainer **data_container**
- bool **debug**

13.87.1 Detailed Description

template<typename Data>
class woss::WossCreatorContainer< Data * >

Partial specialization for pointers.

Partial specialiation for heap-allocated pointers. PLEASE NOTE that object class type must implement clone() method

13.87.2 Member Function Documentation

13.87.2.1 accessAllLocations() template<typename Data >

```
Data *& woss::WossCreatorContainer< Data * >::accessAllLocations  [inline]
```

Returns a reference to the Data pointer associated to transmitter and receiver Location equal to ALL_LOCATIONS. If no keys were present, the Data pointer is defaulted to NULL and a reference to it is returned. The user has to properly delete the pointer prior changing it

Returns

a reference to a Data pointer

References woss::WossCreatorContainer< Data >::ALL_LOCATIONS, and woss::WossCreatorContainer< Data >::data_container.

13.87.2.2 get() [1/2] template<typename Data >

```
Data * woss::WossCreatorContainer< Data * >::get (
            const CoordZ & tx,
            const CoordZ & rx ) const  [inline]
```

Returns a Data pointer for given keys. If the keys are not found a NULL pointer is returned

Parameters

tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

a pointer to a cloned Data object

References woss::WossCreatorContainer< Data >::ALL_COORDZ, woss::WossCreatorContainer< Data >::data_container, and woss::WossCreatorContainer< Data >::debug.

13.87.2.3 get() [2/2] template<typename Data >

```
Data * woss::WossCreatorContainer< Data * >::get (
            Location *const tx,
            Location *const rx ) const  [inline]
```

Returns a Data pointer for given keys. If the keys are not found a NULL pointer is returned

Parameters

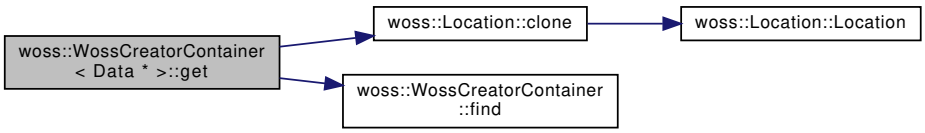
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

a pointer to a cloned Data object

References woss::WossCreatorContainer< Data >::ALL_LOCATIONS, woss::Location::clone(), woss::WossCreatorContainer< Data
woss::WossCreatorContainer< Data >::debug, and woss::WossCreatorContainer< Data >::find().

Here is the call graph for this function:



13.87.2.4 insert() [1/2] template<typename Data >

```
bool woss::WossCreatorContainer< Data * >::insert (
            Data * data,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted and it is deleted

Parameters

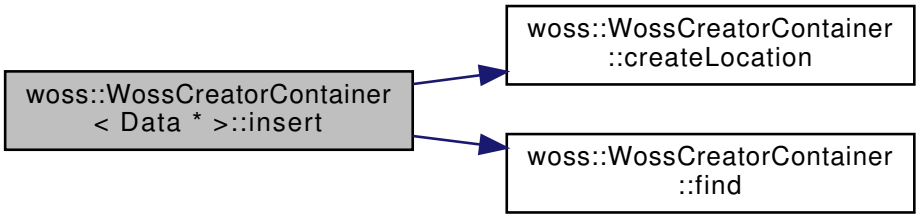
data	const pointer to a Data object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::createLocation(), woss::WossCreatorContainer< Data >::data_container,
and woss::WossCreatorContainer< Data >::find().

Here is the call graph for this function:



13.87.2.5 insert() [2/2] template<typename Data >

```
bool woss::WossCreatorContainer< Data * >::insert (
            Data * data,
            Location *const tx,
            Location *const rx )  [inline]
```

Inserts an object into the container. If the keys are already present, the object is NOT inserted and it is deleted

Parameters

data	const pointer to a Data object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::data_container.

13.87.2.6 replace() [1/2] template<typename Data >

```
void woss::WossCreatorContainer< Data * >::replace (
            Data *const data,
            const CoordZ & tx,
            const CoordZ & rx )  [inline]
```

Replaces a pointer into the container. If the keys are already present, the previous object is deleted

Parameters

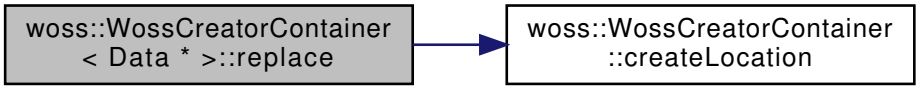
data	const pointer to a Data object to be inserted
tx	const reference to a valid transmitter CoordZ
rx	const reference to a valid receiver CoordZ

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::createLocation(), and woss::WossCreatorContainer< Data >::data_container.

Here is the call graph for this function:



13.87.2.7 replace() [2/2] template<typename Data >

```
void woss::WossCreatorContainer< Data * >::replace (
            Data *const data,
            Location *const tx,
            Location *const rx )  [inline]
```

Replaces a pointer into the container. If the keys are already present, the previous object is deleted

Parameters

data	const pointer to a Data object to be inserted
tx	const pointer to a valid transmitter Location
rx	const pointer to a valid receiver Location

Returns

true if it's empty, *false* otherwise

References woss::WossCreatorContainer< Data >::data_container.

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

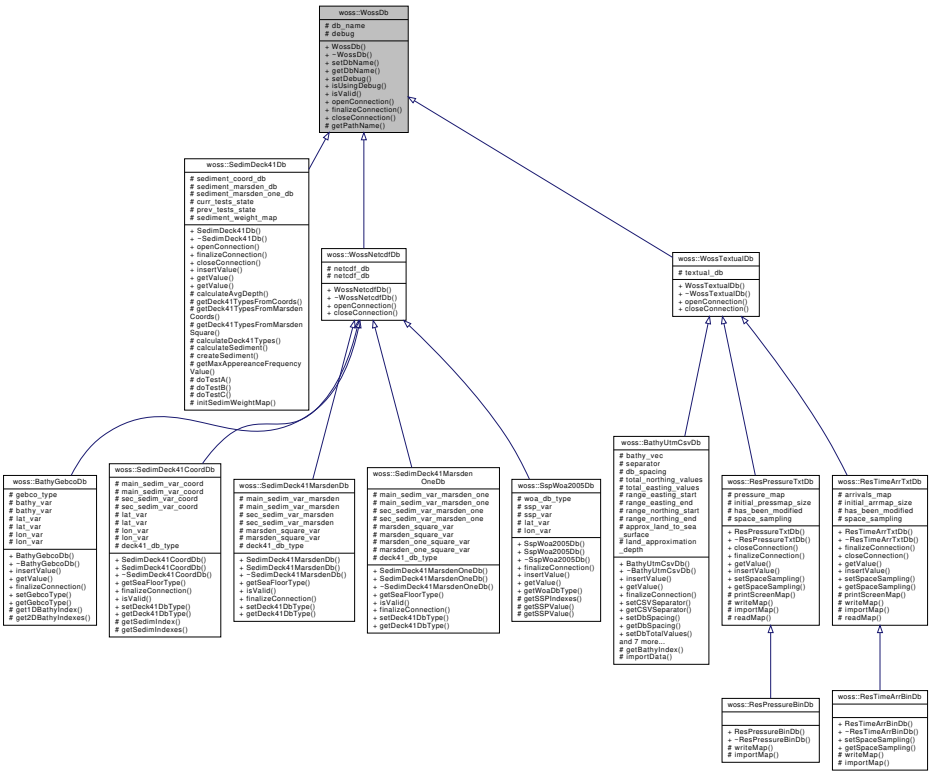
- woss/woss-creator-container.h

13.88 woss::WossDb Class Reference

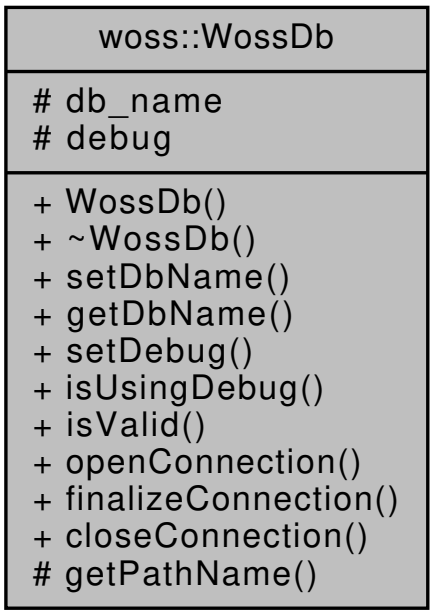
Abstract class that provides the interface of databases.

`#include <woss-db.h>`

Inheritance diagram for woss::WossDb:



Collaboration diagram for woss::WossDb:



Public Member Functions

- WossDb (const ::std::string &name)
- void setDbName (const ::std::string &pathname)
- ::std::string getDbName () const
- void setDebug (double flag=true)
- bool isUsingDebug () const
- virtual bool isValid ()
- virtual bool openConnection ()=0
- virtual bool finalizeConnection ()=0
- virtual bool closeConnection ()=0

Protected Member Functions

- PathName getPathName (const ::std::string &complete_path)

Protected Attributes

- ::std::string db_name
- bool debug

13.88.1 Detailed Description

Abstract class that provides the interface of databases.

WossDb is the foundation of any database included in WOSS. It has the tasks of opening, setting up and closing a connection do a database. Technology dependant issues should be putted in a class that inherits from WossDb.No specific data behaviour is defined here, this will be provided in other abstract classes: this has been specifically done to divide db technology from data. A custom database would have to inherit from a technology dependant class and from a data behaviour class

See also

WossBathymetryDb, WossSedimentDb, WossSSPDb, WossResArrDb, WossResPressDb

13.88.2 Constructor & Destructor Documentation

13.88.2.1 WossDb() `WossDb::WossDb ( const ::std::string & name )`

WossDb constructor

Parameters

name	pathname of database

13.88.3 Member Function Documentation

13.88.3.1 closeConnection() `virtual bool woss::WossDb::closeConnection ( ) [pure virtual]`

Closes the connection to the open database

Returns

true if method was successful, *false* otherwise

Implemented in woss::ResPressureTxtDb, woss::ResTimeArrTxtDb, woss::SedimDeck41Db, woss::WossNetcdfDb, and woss::WossTextualDb.

13.88.3.2 finalizeConnection() `virtual bool woss::WossDb::finalizeConnection ( ) [pure virtual]`

Post openConnection() actions. E.g. creating variables, initializing special pointers etc...

Returns

true if method was successful, *false* otherwise

Implemented in woss::BathyGebcoDb, woss::BathyUtmCsvDb, woss::ResPressureTxtDb, woss::ResTimeArrTxtDb, woss::SedimDeck41CoordDb, woss::SedimDeck41Db, woss::SedimDeck41MarsdenDb, woss::SedimDeck41MarsdenOneDb, and woss::SspWoa2005Db.

Referenced by woss::WossDbCreator::initializeDb().

13.88.3.3 getDbName() ::std::string woss::WossDb::getDbName ( ) const [inline]

Gets the pathname (or custom network address) of database file

Returns

pathname database pathname

References db_name.

13.88.3.4 getPathName() PathName WossDb::getPathName (const ::std::string & *complete_path*) [protected]

Gets path and database name separated

Returns

PathName value

See also

PathName

13.88.3.5 isUsingDebug() bool woss::WossDb::isUsingDebug ( ) const [inline]

Gets debug flag

Returns

true if using debug, *false* otherwise

References debug.

13.88.3.6 isValid() virtual bool woss::WossDb::isValid ( ) [inline], [virtual]

Checks the validity of WossDb

Returns

true if pathname is valid, *false* otherwise

Reimplemented in woss::SedimDeck41CoordDb, woss::SedimDeck41MarsdenDb, and woss::SedimDeck41MarsdenOneDb.

References db_name.

Referenced by woss::SedimDeck41CoordDb::isValid(), woss::SedimDeck41MarsdenDb::isValid(), woss::SedimDeck41MarsdenOneD and woss::WossNetcdfDb::openConnection().

13.88.3.7 openConnection() `virtual bool woss::WossDb::openConnection ( ) [pure virtual]`

Opens the connection to the pathname provided

Returns

true if method was successful, *false* otherwise

Implemented in woss::SedimDeck41Db, woss::WossNetcdfDb, and woss::WossTextualDb.

Referenced by woss::WossDbCreator::initializeDb().

13.88.3.8 setDbName() `void woss::WossDb::setDbName ( const ::std::string & pathname ) [inline]`

Sets the pathname (or custom network address) of database file

Parameters

name	pathname database pathname

References db_name.

Referenced by woss::SedimDeck41DbCreator::initializeSedimDb().

13.88.3.9 setDebug() `void woss::WossDb::setDebug ( double flag = true ) [inline]`

Sets debug flag

Parameters

flag	debug flag

References debug.

Referenced by woss::WossDbCreator::initializeDb().

13.88.4 Member Data Documentation

13.88.4.1 db_name ::std::string woss::WossDb::db_name [protected]

Pathname to database. It could be a filesystem pathname, or a network identifier etc...

Referenced by getDbName(), woss::BathyUtmCsvDb::importData(), woss::ResPressureBinDb::importMap(), woss::ResPressureTxtDb::importMap(), woss::ResTimeArrBinDb::importMap(), woss::ResTimeArrTxtDb::importMap(), woss::SedimDeck41CoordDb::isValid(), woss::SedimDeck41MarsdenDb::isValid(), woss::SedimDeck41MarsdenOneDb::isValid(), isValid(), woss::WossNetcdfDb::openConnection(), woss::WossTextualDb::openConnection(), setDbName(), woss::ResPressureBinDb::writeMap(), woss::ResPressureTxtDb::writeMap(), woss::ResTimeArrBinDb::writeMap(), and woss::ResTimeArrTxtDb::writeMap().

13.88.4.2 debug bool woss::WossDb::debug [protected]

Debug flag

Referenced by woss::SedimDeck41Db::calculateDeck41Types(), woss::BathyGebcoDb::finalizeConnection(), woss::BathyUtmCsvDb::getBathyIndex(), woss::SedimDeck41Db::getDeck41TypesFromCoords(), woss::SedimDeck41Db::getDeck4
woss::SedimDeck41Db::getDeck41TypesFromMarsdenSquare(), woss::SedimDeck41CoordDb::getSeaFloorType(), woss::SedimDeck41MarsdenDb::getSeaFloorType(), woss::SedimDeck41MarsdenOneDb::getSeaFloorType(), woss::BathyGebcoDb::getValue(), woss::BathyUtmCsvDb::getValue(), woss::BathyUtmCsvDb::importData(), woss::ResPressureBinDb::importMap(), woss::ResTimeArrBinDb::importMap(), woss::ResPressureTxtDb::insertValue(), woss::ResTimeArrTxtDb::insertValue(), isUsingDebug(), woss::ResTimeArrTxtDb::readMap(), and setDebug().

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

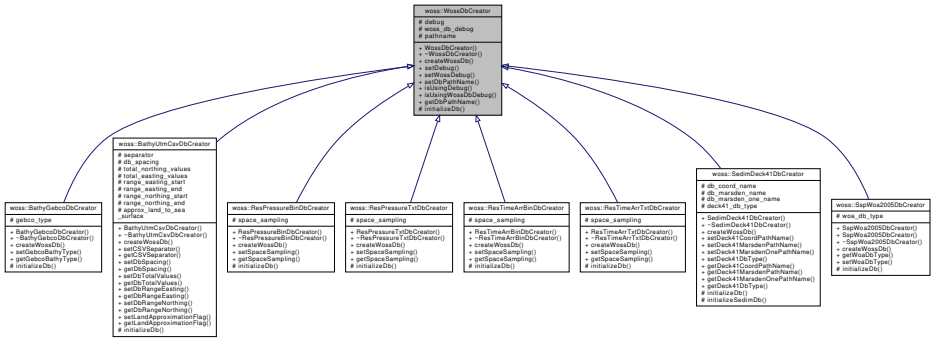
- woss/woss_db/woss-db.h
- woss/woss_db/woss-db.cpp

13.89 woss::WossDbCreator Class Reference

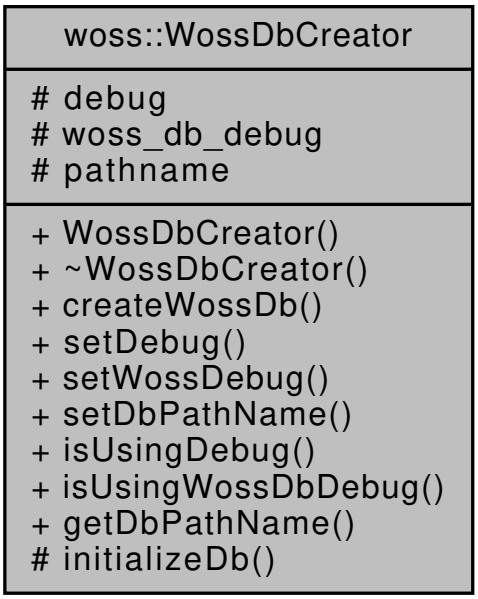
Abstract class that provides the interface of database creator (Factory object)

`#include <woss-db-creator.h>`

Inheritance diagram for woss::WossDbCreator:



Collaboration diagram for woss::WossDbCreator:



Public Member Functions

- WossDbCreator ()
- virtual WossDb ∗const createWossDb ()=0
- WossDbCreator & **setDebug** (bool flag)
- WossDbCreator & **setWossDebug** (bool flag)
- WossDbCreator & **setDbPathName** (const ::std::string &name)
- bool **isUsingDebug** () const
- bool **isUsingWossDbDebug** () const
- ::std::string **getDbPathName** () const

Protected Member Functions

- virtual bool initializeDb (WossDb ∗const woss_db)=0

Protected Attributes

- bool debug
- bool woss_db_debug
- ::std::string pathname

13.89.1 Detailed Description

Abstract class that provides the interface of database creator (Factory object)

WossDbCreator is the prototype of any database creator included in WOSS. It has the tasks of creating and initializing a specific database object, thus abstracting the caller from specific implementation parameters, and the database itself from system specifics (c++ framework, NS-2 framework, NS-3 framework and so on...)

It also provides a Tcl interpreter for NS-2 implementation.

See also

BathyGebcoDbCreator, SedimDeck41DbCreator, SspWoa2005DbCreator, ResTimeArrTxtDbCreator, ResPressureTxtDbCreator

13.89.2 Constructor & Destructor Documentation

13.89.2.1 WossDbCreator() `WossDbCreator::WossDbCreator ( )`

Default WossDbCreator creator

13.89.3 Member Function Documentation

13.89.3.1 createWossDb() `virtual WossDb *const woss::WossDbCreator::createWossDb ( ) [pure virtual]`

Abstract method. It is called to create and initialize a WossDb. The caller will be the **owner** of created object, therefore object destruction is his responsibility

Returns

a pointer to a properly initialized WossDb object

Implemented in woss::BathyGebcoDbCreator, woss::BathyUtmCsvDbCreator, woss::ResPressureBinDbCreator, woss::ResPressureTxtDbCreator, woss::ResTimeArrBinDbCreator, woss::ResTimeArrTxtDbCreator, woss::SedimDeck41DbCreator, and woss::SspWoa2005DbCreator.

References debug.

Referenced by woss::WossController::initialize().

13.89.3.2 initializeDb() `bool WossDbCreator::initializeDb ( WossDb *const woss_db ) [protected], [pure virtual]`

Initializes the pointed object

Parameters

woss_db	pointer to a recently created WossDb

Returns

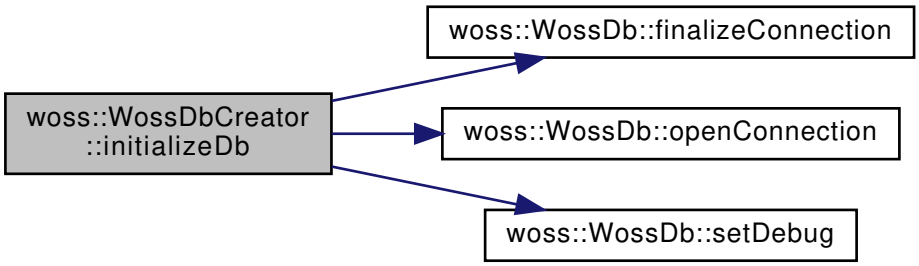
true if the method succeed, *false* otherwise

Implemented in woss::BathyGebcoDbCreator, woss::BathyUtmCsvDbCreator, woss::ResPressureBinDbCreator, woss::ResPressureTxtDbCreator, woss::ResTimeArrBinDbCreator, woss::ResTimeArrTxtDbCreator, woss::SspWoa2005DbCreator, and woss::SedimDeck41DbCreator.

References woss::WossDb::finalizeConnection(), woss::WossDb::openConnection(), woss::WossDb::setDebug(), and woss_db_debug.

Referenced by woss::BathyGebcoDbCreator::initializeDb(), woss::BathyUtmCsvDbCreator::initializeDb(), woss::ResPressureBinDbCr woss::ResPressureTxtDbCreator::initializeDb(), woss::ResTimeArrBinDbCreator::initializeDb(), woss::ResTimeArrTxtDbCreator::initia woss::SspWoa2005DbCreator::initializeDb(), and woss::SedimDeck41DbCreator::initializeDb().

Here is the call graph for this function:



13.89.4 Member Data Documentation

13.89.4.1 debug `bool woss::WossDbCreator::debug [protected]`

WossDbCreator debug flag binded in Tcl

Referenced by woss::ResTimeArrBinDbCreator::createWossDb(), woss::ResTimeArrTxtDbCreator::createWossDb(), and createWossDb().

13.89.4.2 pathname `::std::string woss::WossDbCreator::pathname [protected]`

Pathname or unique identifier

Referenced by woss::BathyGebcoDbCreator::createWossDb(), woss::BathyUtmCsvDbCreator::createWossDb(), woss::ResPressureBinDbCreator::createWossDb(), woss::ResPressureTxtDbCreator::createWossDb(), woss::ResTimeArrBinDbCrea woss::ResTimeArrTxtDbCreator::createWossDb(), and woss::SspWoa2005DbCreator::createWossDb().

13.89.4.3 woss_db_debug `bool woss::WossDbCreator::woss_db_debug [protected]`

WossDb debug flag binded in Tcl. All objects instantiated will have this flag

Referenced by initializeDb().

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

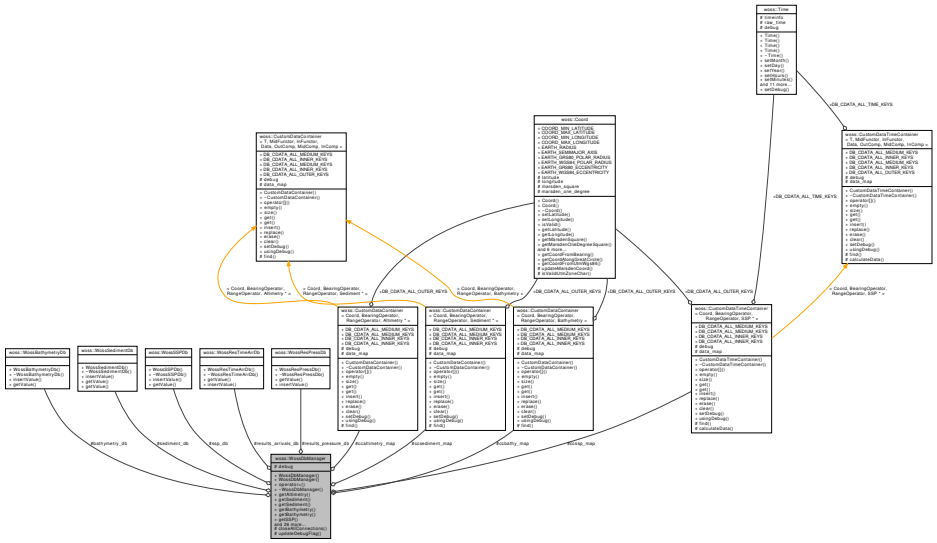
- woss/woss_db/woss-db-creator.h
- woss/woss_db/woss-db-creator.cpp

13.90 woss::WossDbManager Class Reference

Abstraction layer for database and data manipulation.

`#include <woss-db-manager.h>`

Collaboration diagram for woss::WossDbManager:



Classes

- class BearingOperator

  Bearing operator function object.
- class RangeOperator

  Range operator function object.

Public Types

- typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Bathymetry > **CCBathymetry**
- typedef CustomDataTimeContainer< Coord, BearingOperator, RangeOperator, SSP * > **CCSSP**
- typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Sediment * > **CCSediment**
- typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Altimetry * > **CCAltimetry**

Public Member Functions

- WossDbManager ()
- WossDbManager (WossDbManager &instance)
- WossDbManager & operator= (WossDbManager &instance)
- virtual ~WossDbManager ()
- virtual Altimetry * getAltimetry (const CoordZ &tx, const CoordZ &rx) const
- virtual Sediment * getSediment (const CoordZ &tx, const CoordZ &rx) const
- virtual Sediment * getSediment (const CoordZ &tx, const CoordZVector &rx_coordz_vector) const
- virtual Bathymetry getBathymetry (const Coord &tx, const Coord &rx) const
- virtual void getBathymetry (const Coord &tx, CoordZVector &rx_coordz_vector) const

- virtual SSP * getSSP (const Coord &tx, const Coord &rx, const Time &time, long double ssp_depth_precision=SSP_CUSTOM_DEPTH_PRECISION) const
- virtual SSP * getAverageSSP (const Coord &tx, const Coord &rx, const Time &time_start, const Time &time_end, int max_time_values, long double ssp_depth_precision=SSP_CUSTOM_DEPTH_PRECISION) const
- virtual TimeArr * getTimeArr (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const
- virtual void insertTimeArr (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const TimeArr &channel) const
- virtual Pressure * getPressure (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const
- virtual void insertPressure (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const Pressure &pressure) const
- WossDbManager & setBathymetryDb (WossBathymetryDb *ptr)
- WossDbManager & setSedimentDb (WossSedimentDb *ptr)
- WossDbManager & setSSPDb (WossSSPDb *ptr)
- WossDbManager & setResTimeArrDb (WossResTimeArrDb *ptr)
- WossDbManager & setResPressureDb (WossResPressDb *ptr)
- bool setCustomAltimetry (Altimetry *const altimetry, const Coord &tx_coord=CCAltimetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCAltimetry::DB_CDATA_ALL_INNER_KEYS)
- Altimetry * getCustomAltimetry (const Coord &tx_coord=CCAltimetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCAltimetry::DB_CDATA_ALL_INNER_KEYS)
- WossDbManager & eraseCustomAltimetry (const Coord &tx_coord=CCAltimetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCAltimetry::DB_CDATA_ALL_INNER_KEYS)
- bool setCustomSediment (Sediment *const sediment, const Coord &tx_coord=CCSediment::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSediment::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSediment::DB_CDATA_ALL_INNER_KEYS)
- Sediment * getCustomSediment (const Coord &tx_coord=CCSediment::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSediment::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSediment::DB_CDATA_ALL_INNER_KEYS)
- WossDbManager & eraseCustomSediment (const Coord &tx_coord=CCSediment::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSediment::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSediment::DB_CDATA_ALL_INNER_KEYS)
- bool setCustomSSP (SSP *const ssp, const Coord &tx_coord=CCSSP::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSSP::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSSP::DB_CDATA_ALL_INNER_KEYS, const Time &time_value=CCSSP::DB_CDATA_ALL_TIME_KEYS)
- virtual bool importCustomSSP (const ::std::string &filename, const Time &=CCSSP::DB_CDATA_ALL_TIME_KEYS, const Coord &tx_coord=CCSSP::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSSP::DB_CDATA_ALL_MEDIUM_KEYS)
- SSP * getCustomSSP (const Coord &tx_coord=CCSSP::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSSP::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSSP::DB_CDATA_ALL_INNER_KEYS, const Time &time_value=CCSSP::DB_CDATA_ALL_TIME_KEYS)
- WossDbManager & eraseCustomSSP (const Coord &tx_coord=CCSSP::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCSSP::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCSSP::DB_CDATA_ALL_INNER_KEYS, const Time &time_value=CCSSP::DB_CDATA_ALL_TIME_KEYS)
- bool setCustomBathymetry (Bathymetry *const bathymetry, const Coord &tx_coord=CCBathymetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCBathymetry::DB_CDATA_ALL_INNER_KEYS)
- virtual bool importCustomBathymetry (const ::std::string &filename, const Coord &tx_coord=CCBathymetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS)
- Bathymetry * getCustomBathymetry (const Coord &tx_coord=CCBathymetry::DB_CDATA_ALL_OUTER_KEYS, double bearing=CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCBathymetry::DB_CDATA_ALL_INNER_KEYS)

- WossDbManager & eraseCustomBathymetry (const Coord &tx_coord=CCBathymetry::DB_CDATA_ALL_↩ OUTER_KEYS, double bearing=CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS, double range=CCBathymetry↩ ::DB_CDATA_ALL_INNER_KEYS)
- WossDbManager & **setDebug** (bool flag)
- bool **getDebug** ()

Protected Member Functions

- virtual bool closeAllConnections ()
- virtual void **updateDebugFlag** ()

Protected Attributes

- WossBathymetryDb ∗ bathymetry_db
- WossSedimentDb ∗ sediment_db
- WossSSPDb ∗ ssp_db
- WossResTimeArrDb ∗ results_arrivals_db
- WossResPressDb ∗ results_pressure_db
- bool debug
- CCBathymetry ccbathy_map
- CCSediment ccsediment_map
- CCSSP ccssp_map
- CCAltimetry ccaltimetry_map

13.90.1 Detailed Description

Abstraction layer for database and data manipulation.

WossDbManager has the exclusive handling of **all** databases involved. No other object can access directly to a WossDb entity. Thanks to the abstraction provided, WossDbManager can perform arithmetic and other data manipulation. It should be used with woss::Singleton for safety reasons. (e.g. woss::Singleton<woss::WossDbManager>) WossDbManager also provides a way to generate environmental data on the fly. Providing a generator coordinate and a set of bearing and ranges, the user can create all sort of polygon where the given environmental data is valid.

See also

setCustom∗ methods, command

13.90.2 Constructor & Destructor Documentation

13.90.2.1 WossDbManager() [1/2] `WossDbManager::WossDbManager ( )`

Default constructor

13.90.2.2 WossDbManager() [2/2] WossDbManager::WossDbManager (WossDbManager & *instance*)

Copy constructor (no const here, we have to modify the copy)

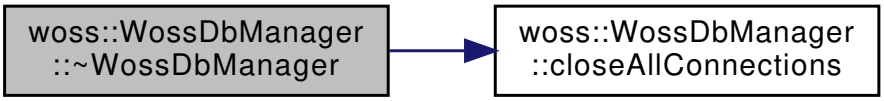
References bathymetry_db, results_arrivals_db, results_pressure_db, sediment_db, and ssp_db.

13.90.2.3 ~WossDbManager() WossDbManager::~WossDbManager ( ) [virtual]

Destructor. It deletes all pointers involved. Derived classes **don't have to do it**

References bathymetry_db, closeAllConnections(), results_arrivals_db, results_pressure_db, sediment_db, and ssp_db.

Here is the call graph for this function:



13.90.3 Member Function Documentation

13.90.3.1 closeAllConnections() bool WossDbManager::closeAllConnections ( ) [protected], [virtual]

Closes all connections of owned databases

References bathymetry_db, results_arrivals_db, results_pressure_db, sediment_db, and ssp_db.

Referenced by ~WossDbManager().

13.90.3.2 eraseCustomAltimetry() WossDbManager & woss::WossDbManager::eraseCustomAltimetry (const Coord & *tx_coord* = *CCAltimetry::DB_CDATA_ALL_OUTER_KEYS*, double *bearing* = *CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS*, double *range* = *CCAltimetry::DB_CDATA_ALL_INNER_KEYS*) [inline]

Erases the custom Altimetry for given parameters

Parameters

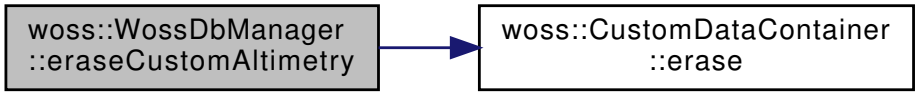
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

reference to ∗**this**

References ccaltimetry_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::er

Here is the call graph for this function:



13.90.3.3 eraseCustomBathymetry()

```
WossDbManager & woss::WossDbManager::eraseCustomBathymetry (
    const Coord & tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Erases the custom Bathymetry for given parameters

Parameters

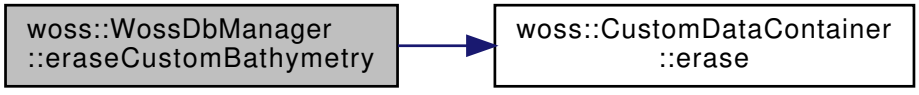
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

reference to ∗**this**

References ccbathy_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::erase

Here is the call graph for this function:



13.90.3.4 eraseCustomSediment()

```
WossDbManager & woss::WossDbManager::eraseCustomSediment (
    const Coord & tx_coord = CCSediment::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSediment::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Erases the custom Sediment for given parameters

Parameters

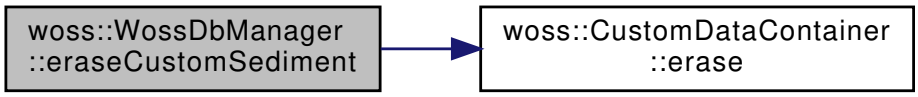
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

reference to ∗**this**

References ccsediment_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::e

Here is the call graph for this function:



13.90.3.5 eraseCustomSSP()

```
WossDbManager & woss::WossDbManager::eraseCustomSSP (
    const Coord & tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
    const Time & time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS )  [inline]
```

Erases the custom SSP for given parameters

Parameters

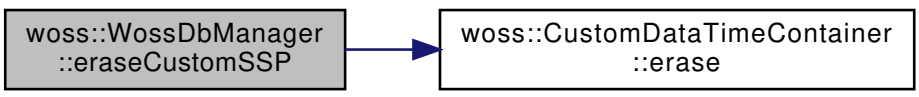
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]
time_value	const reference to a valid Time object

Returns

reference to ∗**this**

References ccssp_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::er

Here is the call graph for this function:



13.90.3.6 getAltimetry()

```
Altimetry * WossDbManager::getAltimetry (
    const CoordZ & tx,
    const CoordZ & rx ) const  [virtual]
```

Returns a pointer to a heap-created Altimetry value for given coordinates and depth, if present in the Altimetry database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid CoordZ object

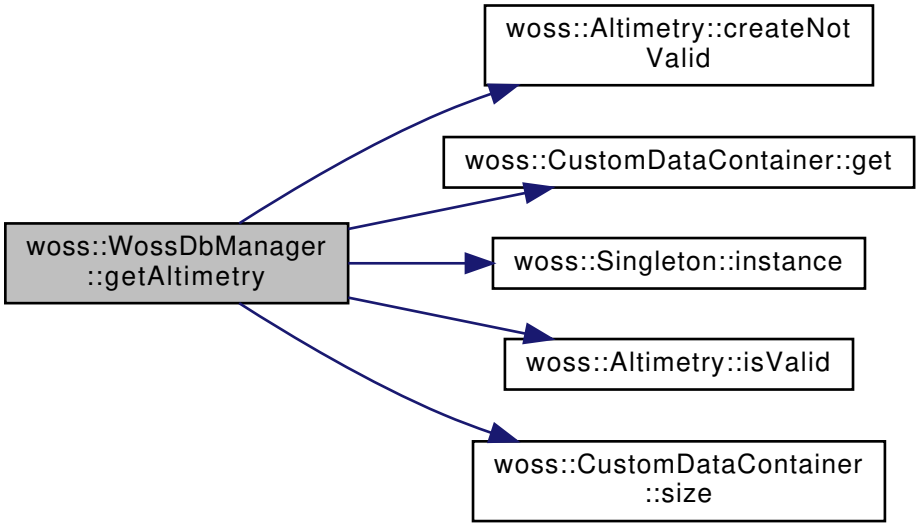
Returns

valid Altimetry if coordinates are found, *not valid* otherwise

References ccaltimetry_map, woss::Altimetry::createNotValid(), debug, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, woss::Singleton< T >::instance(), woss::Altimetry::isValid(), and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutC

Referenced by woss::ACToolboxWoss::initAltimetry().

Here is the call graph for this function:



13.90.3.7 getAverageSSP()

```
SSP * WossDbManager::getAverageSSP (
    const Coord & tx,
    const Coord & rx,
    const Time & time_start,
    const Time & time_end,
    int max_time_values,
    long double ssp_depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const  [virtual]
```

Returns a pointer a heap-created average SSP for given coordinates, start and end time date if they are present in the database. **User is responsible of pointer's ownership**

Parameters

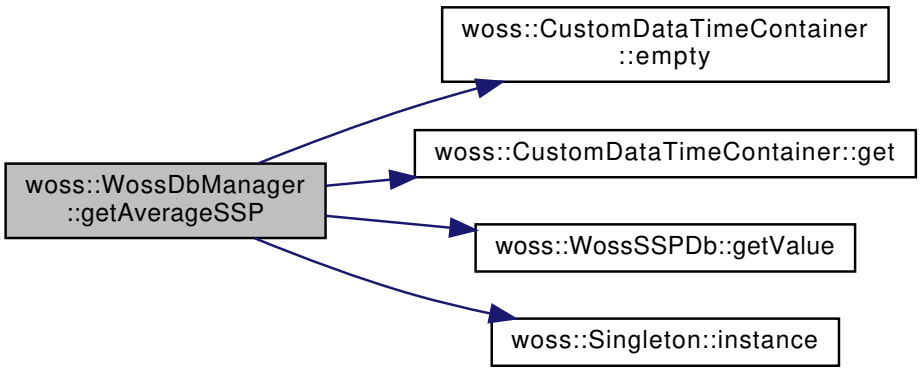
coords	const reference to a valid Coord object
time_start	const reference to a valid Time object
time_end	const reference to a valid Time object
max_time_values	total number of Time to take between time_start and time_end
ssp_depth_precision	returned SSP's depth precision

Returns

valid SSP if coordinates and both time date are found, *not valid* otherwise

References ccssp_map, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get(), woss::WossSSPDb::getV, woss::Singleton< T >::instance(), and ssp_db.

Here is the call graph for this function:



13.90.3.8 getBathymetry() [1/2]

```
double WossDbManager::getBathymetry (
    const Coord & tx,
    const Coord & rx ) const  [virtual]
```

Returns the positive depth value (bathymetry) of given coordinates, if present in the database

Parameters

coords	const reference to a valid Coord object

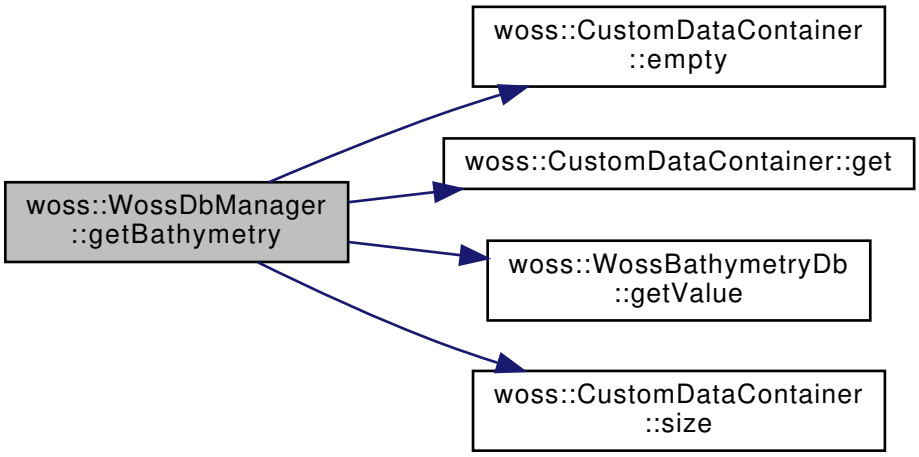
Returns

positive depth value [m] if coordinates are found, *HUGE_VAL* otherwise

References bathymetry_db, ccbathy_map, debug, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidCom, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get(), woss::WossBathymetryDb::ge, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::size().

Referenced by getBathymetry(), and woss::ACToolboxWoss::initCoordZVector().

Here is the call graph for this function:



13.90.3.9 getBathymetry() [2/2]

```
void WossDbManager::getBathymetry (
    const Coord & tx,
    CoordZVector & rx_coordz_vector ) const  [virtual]
```

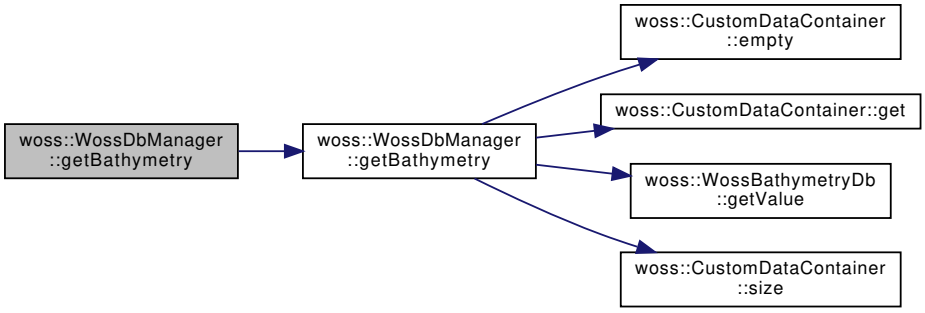
Sets the positive depth for each CoordZ present in the vector, HUGE_VAL is set if coordinates are not present in the database

Parameters

coords	reference to a CoordZVector

References getBathymetry().

Here is the call graph for this function:



13.90.3.10 getCustomAltimetry()

```
Altimetry * woss::WossDbManager::getCustomAltimetry (
    const Coord & tx_coord = CCAltimetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCAltimetry::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Gets the custom Altimetry for given generator Coord

Parameters

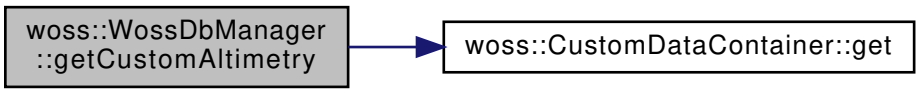
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

NULL pointer if parameters are not found.

References ccaltimetry_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::ge

Here is the call graph for this function:



13.90.3.11 getCustomBathymetry()

```
Bathymetry * woss::WossDbManager::getCustomBathymetry (
    const Coord & tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Gets the custom Bathymetry for given generator Coord

Parameters

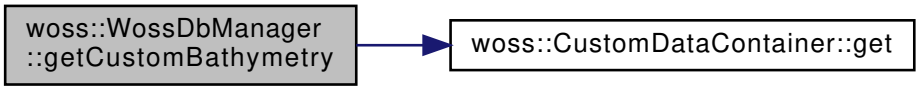
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

NULL pointer if parameters are not found.

References ccbathy_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get().

Here is the call graph for this function:



13.90.3.12 getCustomSediment()

```
Sediment * woss::WossDbManager::getCustomSediment (
    const Coord & tx_coord = CCSediment::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSediment::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Gets the custom Sediment for given generator Coord

Parameters

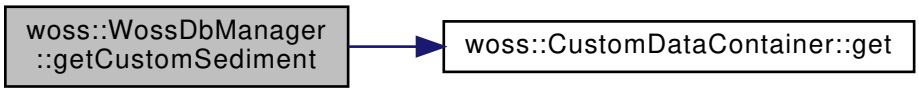
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

NULL pointer if parameters are not found.

References ccsediment_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::g

Here is the call graph for this function:



13.90.3.13 getCustomSSP()

```
SSP * woss::WossDbManager::getCustomSSP (
    const Coord & tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
    const Time & time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS )  [inline]
```

Gets the custom SSP for given generator Coord

Parameters

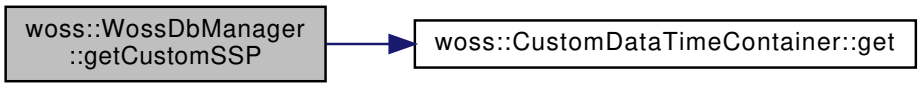
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]
time_value	const reference to a valid Time object

Returns

NULL pointer if parameters are not found.

References ccssp_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::ge

Here is the call graph for this function:



13.90.3.14 getPressure()

```
Pressure * WossDbManager::getPressure (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [virtual]
```

Returns a pointer to a heap-created Pressure value of given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object

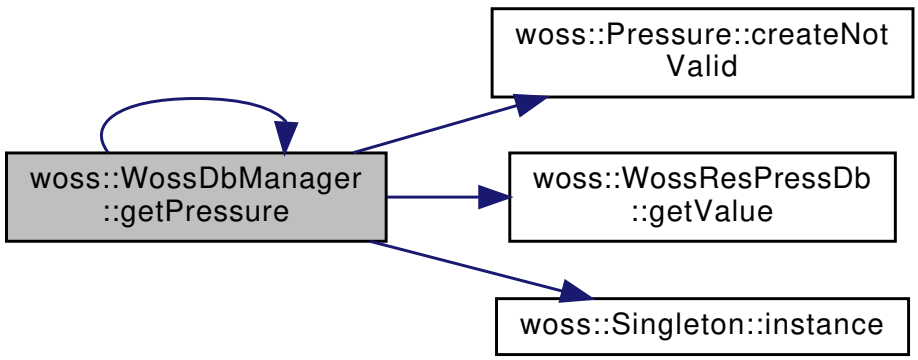
Returns

valid TimeArr if parameters are found, *not valid* otherwise

References woss::Pressure::createNotValid(), getPressure(), woss::WossResPressDb::getValue(), woss::Singleton< T >::instance(), and results_pressure_db.

Referenced by woss::WossManagerResDb::dbGetPressure(), and getPressure().

Here is the call graph for this function:



13.90.3.15 getSediment() [1/2]

```
Sediment * WossDbManager::getSediment (
    const CoordZ & tx,
    const CoordZ & rx ) const  [virtual]
```

Returns a pointer to a heap-created Sediment value for given coordinates and depth, if present in the Sediment database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid CoordZ object

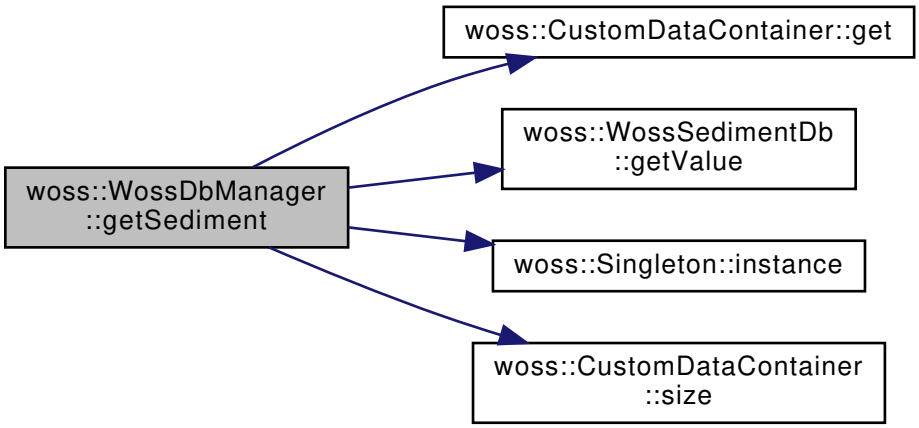
Returns

valid Sediment if coordinates are found, *not valid* otherwise

References ccsediment_map, debug, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >
woss::WossSedimentDb::getValue(), woss::Singleton< T >::instance(), sediment_db, and woss::CustomDataContainer< T, MidFunc

Referenced by getSediment(), and woss::ACToolboxWoss::initSediment().

Here is the call graph for this function:



13.90.3.16 getSediment() [2/2]

```
Sediment * WossDbManager::getSediment (
    const CoordZ & tx,
    const CoordZVector & rx_coordz_vector ) const  [virtual]
```

Returns the representative sediment value of given coordinates and depth vector, if at least one set of coordinates is present in the Sediment database. **User is responsible of pointer's ownership**

Parameters

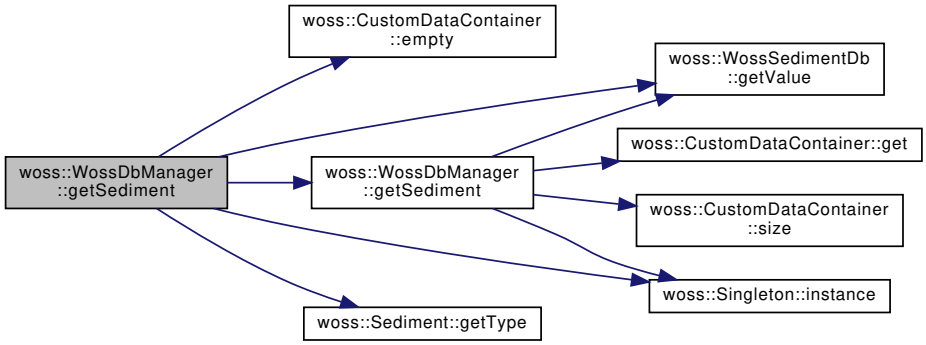
coordz_vector	const reference to a valid CoordZ vector

Returns

valid Sediment if at least one set of coordinates is found, *not valid* otherwise

References ccsediment_map, debug, woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp > getSediment(), woss::Sediment::getType(), woss::WossSedimentDb::getValue(), woss::Singleton< T >::instance(), and sediment_db.

Here is the call graph for this function:



13.90.3.17 getSSP()

```
SSP * WossDbManager::getSSP (
        const Coord & tx,
        const Coord & rx,
        const Time & time,
        long double ssp_depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const  [virtual]
```

Returns the SSP value of given coordinates and date time if both present in the database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid Coord object
time	const reference to a valid Time object
ssp_depth_precision	ssp depth precision [m]

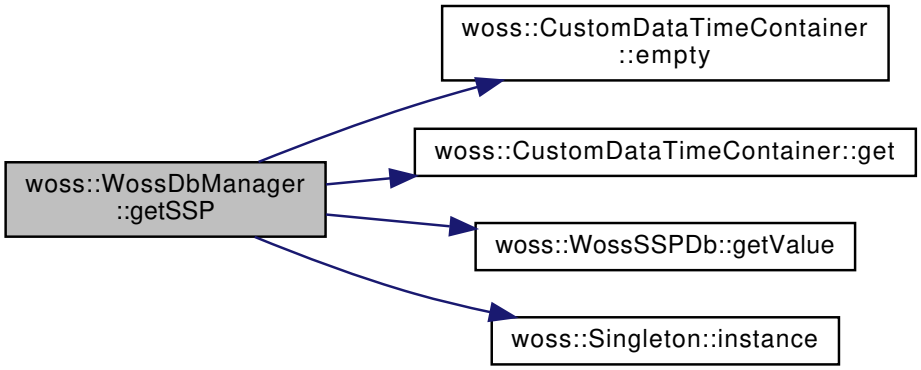
Returns

valid SSP if coordinates are found, *not valid* otherwise

References ccssp_map, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::empty woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get(), woss::WossSSPDb::getV woss::Singleton< T >::instance(), and ssp_db.

Referenced by woss::ACToolboxWoss::initSSPVector().

Here is the call graph for this function:



13.90.3.18 getTimeArr()

```
TimeArr * WossDbManager::getTimeArr (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [virtual]
```

Returns a pointer to a heap-created TimeArr value of given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object

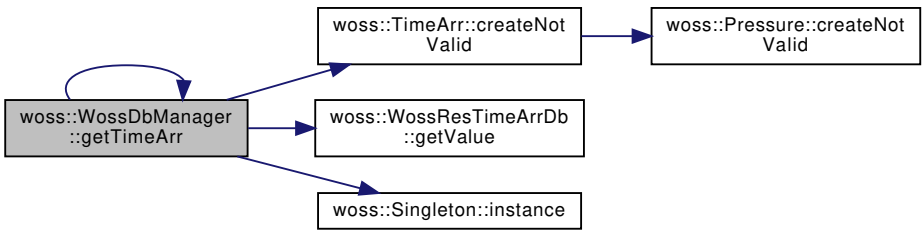
Returns

valid TimeArr if parameters are found, *not valid* otherwise

References woss::TimeArr::createNotValid(), getTimeArr(), woss::WossResTimeArrDb::getValue(), woss::Singleton< T >::instance(), and results_arrivals_db.

Referenced by woss::WossManagerResDb::dbGetTimeArr(), and getTimeArr().

Here is the call graph for this function:



13.90.3.19 importCustomBathymetry()

```
bool WossDbManager::importCustomBathymetry (
    const ::std::string & filename,
    const Coord & tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS )  [virtual]
```

Imports a CustomBathymetry from file. The file has to be a two column format: Range [m] Depth [m]

Parameters

filename	const reference to a string
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]

Returns

true if import was successful, false otherwise

References ccbathy_map, and debug.

13.90.3.20 importCustomSSP()

```
bool WossDbManager::importCustomSSP (
    const ::std::string & filename,
    const Time & time = CCSSP::DB_CDATA_ALL_TIME_KEYS,
    const Coord & tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS )  [virtual]
```

Imports a CustomSSP from file.
File format:

- first line: type name.
- second line: latitude for depth and pressure correction purposes
- third line: longitude for depth and pressure correction purposes

Following format depends on type name:

- **"SSP"** : three columns. range [m] | depth [m] | sound speed [m/s]
- **"FULL"** : six columns. range [m] | depth [m] | temperature [C°] | salinity [ppu] | pressure [bar] | sound speed [m/s]
- **"TEMPERATURE_SALINITY_PRESSURE"** : four columns. range [m] | temperature [C°] | salinity [ppu] | pressure [bar]
- **"DEPTH_TEMPERATURE_SALINITY"** : four columns. range [m] | depth [m] | temperature [C°] | salinity [ppu]

Parameters

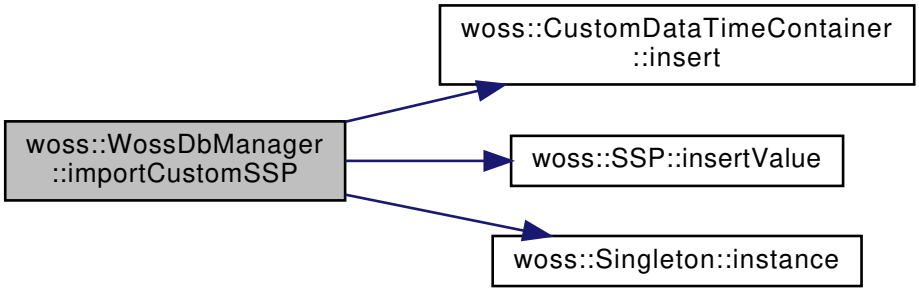
filename	const reference to a string
time	const reference to a valid Time object
tx_coord	const reference to a valid Coord object (originator coordinates)
bearing	bearing value [radians]

Returns

true is import was completed succesfully, false otherwise

References ccssp_map, debug, woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp > woss::SSP::insertValue(), and woss::Singleton< T >::instance().

Here is the call graph for this function:



13.90.3.21 insertPressure()

```
void WossDbManager::insertPressure (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value,
    const Pressure & pressure ) const  [virtual]
```

Inserts the given Pressure value in the database at given frequency, transmitter and receiver coordinates

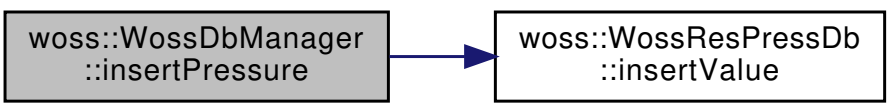
Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object
channel	computed Pressure

References woss::WossResPressDb::insertValue(), and results_pressure_db.

Referenced by woss::WossManagerResDb::dbInsertPressure().

Here is the call graph for this function:



13.90.3.22 insertTimeArr()

```
void WossDbManager::insertTimeArr (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value,
    const TimeArr & channel ) const  [virtual]
```

Inserts a given TimeArr value in the database at given frequency, transmitter and receiver coordinates

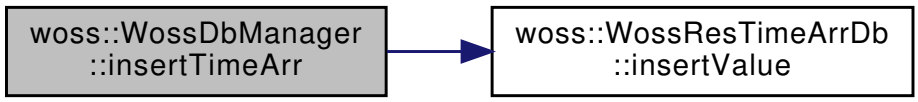
Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
time_value	const reference to a valid Time object
channel	computed TimeArr

References woss::WossResTimeArrDb::insertValue(), and results_arrivals_db.

Referenced by woss::WossManagerResDb::dbInsertTimeArr().

Here is the call graph for this function:



13.90.3.23 operator=()

```
WossDbManager & WossDbManager::operator= (
    WossDbManager & instance )
```

Assignment operator (no const here, we have to modify the copy)

References bathymetry_db, ccaltimetry_map, ccbathy_map, ccsediment_map, ccssp_map, debug, results_arrivals_db, results_pressure_db, sediment_db, and ssp_db.

13.90.3.24 setBathymetryDb()

```
WossDbManager & woss::WossDbManager::setBathymetryDb (
    WossBathymetryDb * ptr )  [inline]
```

Initializes the bathymetry database pointer

Parameters

ptr	valid WossBathymetryDb*

Returns

reference to ∗**this**

References bathymetry_db.

Referenced by woss::WossController::initialize().

13.90.3.25 setCustomAltimetry()

```
bool woss::WossDbManager::setCustomAltimetry (
    Altimetry *const altimetry,
    const Coord & tx_coord = CCAltimetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCAltimetry::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Sets the custom Altimetry for given generator Coord

Parameters

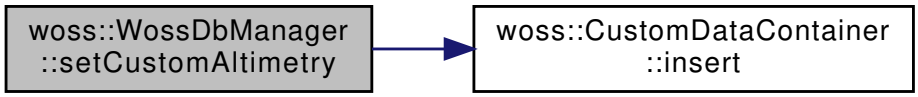
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

true if method succed, *false* otherwise

References ccaltimetry_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::ins

Here is the call graph for this function:



13.90.3.26 setCustomBathymetry()

```
bool woss::WossDbManager::setCustomBathymetry (
    Bathymetry *const bathymetry,
    const Coord & tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Sets the custom Bathymetry for given generator Coord

Parameters

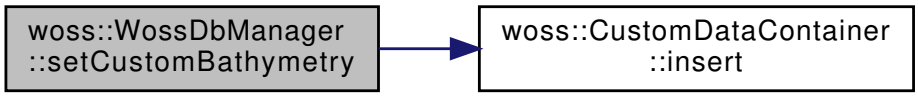
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

if insertion succeed, the function returns a NULL pointer, that needs to be initialized. if an object was already instantiated, the function returns the pointer to it;

References ccbathy_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::inser

Here is the call graph for this function:



13.90.3.27 setCustomSediment()

```
bool woss::WossDbManager::setCustomSediment (
    Sediment *const sediment,
    const Coord & tx_coord = CCSediment::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSediment::DB_CDATA_ALL_INNER_KEYS )  [inline]
```

Sets the custom Sediment for given generator Coord

Parameters

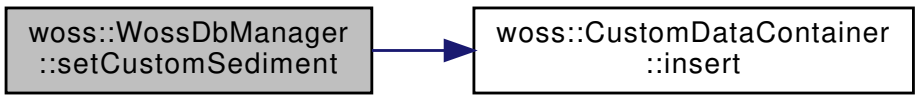
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]

Returns

if insertion succeed, the function returns a NULL pointer, that needs to be initialized. if an object was already instantiated, the function returns the pointer to it;

References ccsediment_map, and woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::in

Here is the call graph for this function:



13.90.3.28 setCustomSSP()

```
bool woss::WossDbManager::setCustomSSP (
    SSP *const ssp,
    const Coord & tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
    double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
    double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
    const Time & time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS )  [inline]
```

Sets the custom SSP for given generator Coord

Parameters

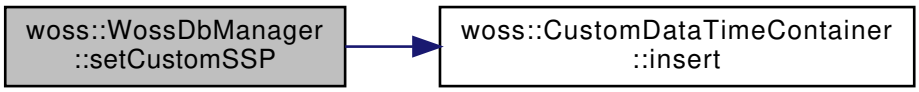
tx_coord	const reference to a valid Coord (generator coordinates)
bearing	bearing value [radians]
range	range value [m]
time_value	const reference to a valid Time object

Returns

if insertion succeed, the function returns a NULL pointer, that needs to be initialized. if an object was already instantiated, the function returns the pointer to it;

References ccssp_map, and woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::ins

Here is the call graph for this function:



13.90.3.29 setResPressureDb()

`WossDbManager & woss::WossDbManager::setResPressureDb ( WossResPressDb * ptr ) [inline]`

Initializes the Pressure database pointer

Parameters

ptr	valid WossResPressDb*

Returns

reference to ***this**

References results_pressure_db.

Referenced by woss::WossController::initialize().

13.90.3.30 setResTimeArrDb()

`WossDbManager & woss::WossDbManager::setResTimeArrDb ( WossResTimeArrDb * ptr ) [inline]`

Initializes the TimeArr database pointer

Parameters

ptr	valid WossResTimeArrDb*

Returns

reference to ∗**this**

References results_arrivals_db.

Referenced by woss::WossController::initialize().

13.90.3.31 setSedimentDb() WossDbManager & woss::WossDbManager::setSedimentDb (WossSedimentDb * *ptr*) [inline]

Initializes the Sediment database pointer

Parameters

ptr	valid WossSedimentDb∗

Returns

reference to ∗**this**

References sediment_db.

Referenced by woss::WossController::initialize().

13.90.3.32 setSSPDb() WossDbManager & woss::WossDbManager::setSSPDb (WossSSPDb * *ptr*) [inline]

Initializes the SSP database pointer

Parameters

ptr	valid WossSSPDb∗

Returns

reference to ∗**this**

References ssp_db.

Referenced by woss::WossController::initialize().

13.90.4 Member Data Documentation

13.90.4.1 bathymetry_db `WossBathymetryDb* woss::WossDbManager::bathymetry_db [protected]`

Bathymetry database pointer

Referenced by closeAllConnections(), getBathymetry(), operator=(), setBathymetryDb(), WossDbManager(), and ~WossDbManager().

13.90.4.2 ccaltimetry_map `CCAltimetry woss::WossDbManager::ccaltimetry_map [protected]`

CustomAltimetry containter for user-given generator CoordZ

Referenced by eraseCustomAltimetry(), getAltimetry(), getCustomAltimetry(), operator=(), and setCustomAltimetry().

13.90.4.3 ccbathy_map `CCBathymetry woss::WossDbManager::ccbathy_map [protected]`

CustomBathymetry containter for user-given generator CoordZ

Referenced by eraseCustomBathymetry(), getBathymetry(), getCustomBathymetry(), importCustomBathymetry(), operator=(), and setCustomBathymetry().

13.90.4.4 ccsediment_map `CCSediment woss::WossDbManager::ccsediment_map [protected]`

custom Sediment containter for user-given generator CoordZ

Referenced by eraseCustomSediment(), getCustomSediment(), getSediment(), operator=(), and setCustomSediment().

13.90.4.5 ccssp_map `CCSSP woss::WossDbManager::ccssp_map [protected]`

CustomSSP containter for user-given generator CoordZ

Referenced by eraseCustomSSP(), getAverageSSP(), getCustomSSP(), getSSP(), importCustomSSP(), operator=(), and setCustomSSP().

13.90.4.6 debug `bool woss::WossDbManager::debug [protected]`

Debug flag

Referenced by getAltimetry(), getBathymetry(), getSediment(), importCustomBathymetry(), importCustomSSP(), and operator=().

13.90.4.7 results_arrivals_db

`WossResTimeArrDb* woss::WossDbManager::results_arrivals_db [protected]`

TimeArr database pointer

Referenced by closeAllConnections(), getTimeArr(), insertTimeArr(), operator=(), setResTimeArrDb(), WossDbManager(), and ~WossDbManager().

13.90.4.8 results_pressure_db

`WossResPressDb* woss::WossDbManager::results_pressure_db [protected]`

Pressure database pointer

Referenced by closeAllConnections(), getPressure(), insertPressure(), operator=(), setResPressureDb(), WossDbManager(), and ~WossDbManager().

13.90.4.9 sediment_db

`WossSedimentDb* woss::WossDbManager::sediment_db [protected]`

Sediment database pointer

Referenced by closeAllConnections(), getSediment(), operator=(), setSedimentDb(), WossDbManager(), and ~WossDbManager().

13.90.4.10 ssp_db

`WossSSPDb* woss::WossDbManager::ssp_db [protected]`

SSP database pointer

Referenced by closeAllConnections(), getAverageSSP(), getSSP(), operator=(), setSSPDb(), WossDbManager(), and ~WossDbManager().

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

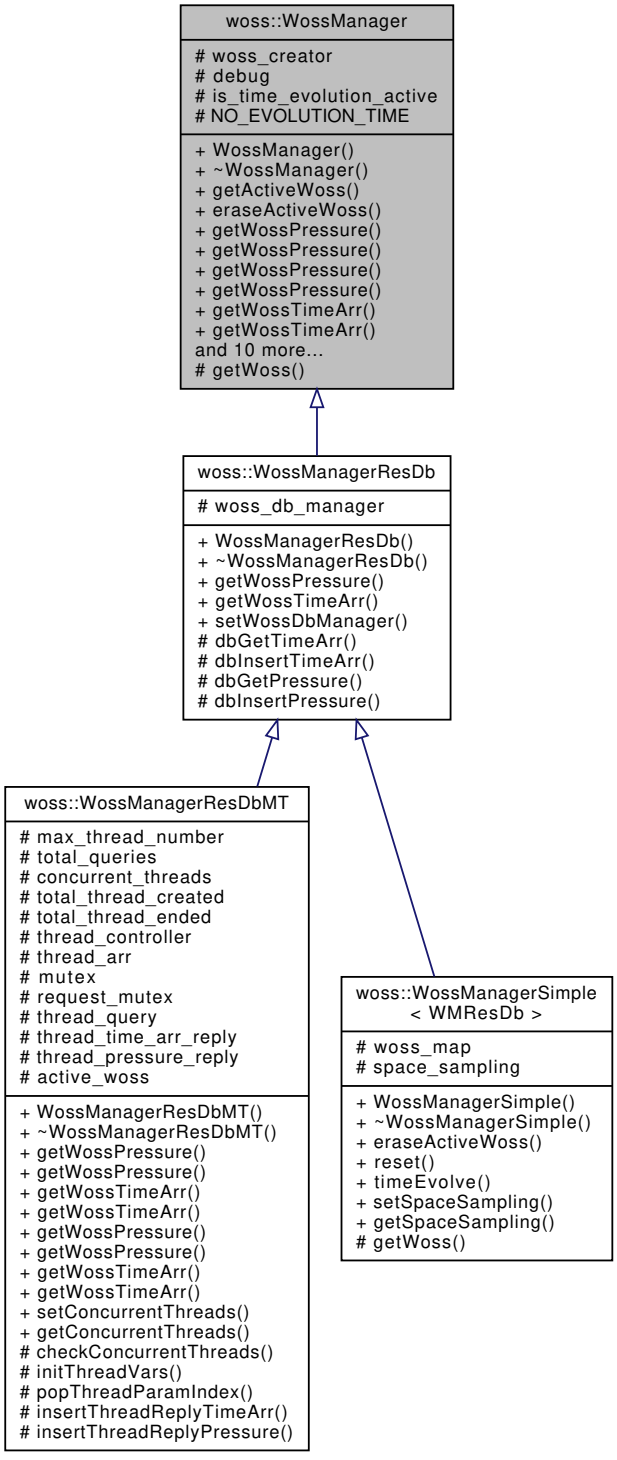
- woss/woss_db/woss-db-manager.h
- woss/woss_db/woss-db-manager.cpp

13.91 woss::WossManager Class Reference

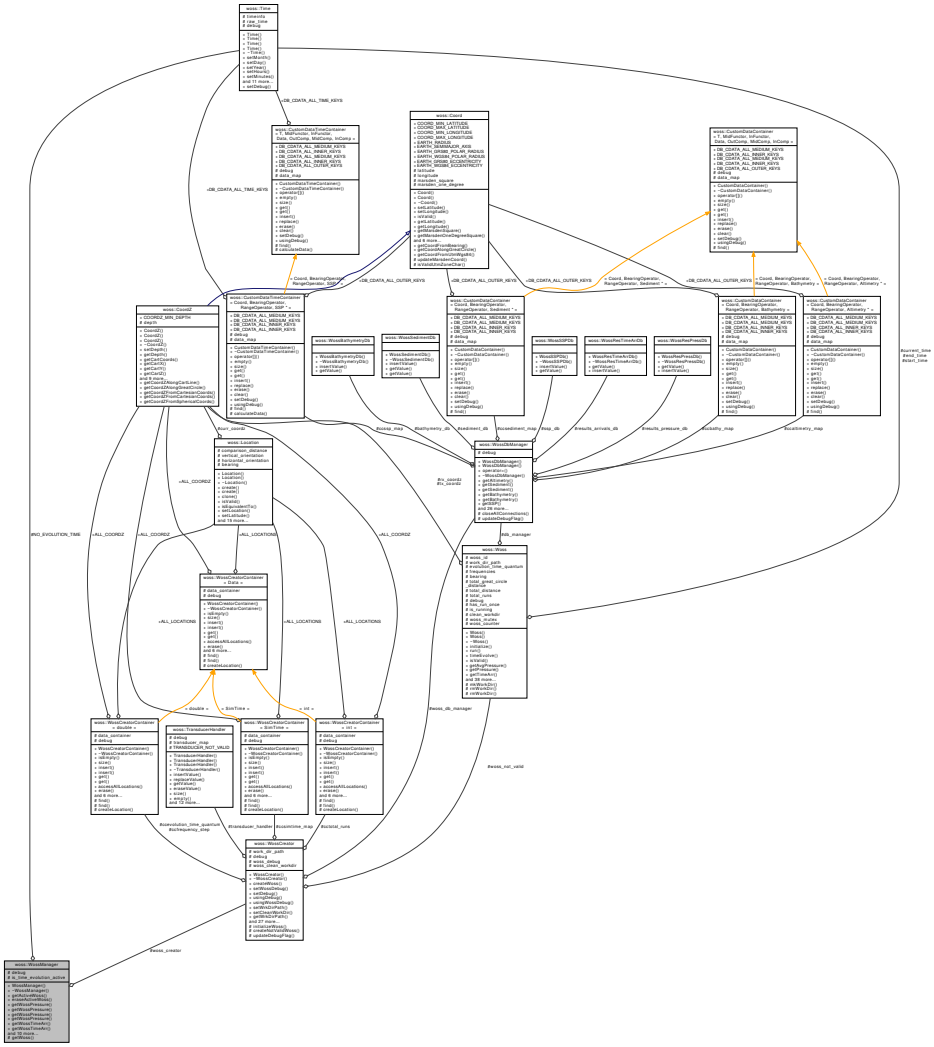
Abstract class that interfaces Pressure or TimeArr requests from user layer.

`#include <woss-manager.h>`

Inheritance diagram for woss::WossManager:



Collaboration diagram for woss::WossManager:



Public Member Functions

- WossManager ()
- virtual const Woss & getActiveWoss (const CoordZ &tx, const CoordZ &rx, double start_frequency, double end_frequency) const
- virtual WossManager & eraseActiveWoss (const CoordZ &tx, const CoordZ &rx, double start_frequency, double end_frequency)=0
- virtual Pressure * getWossPressure (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)=0
- virtual Pressure * getWossPressure (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, double time_value=0.0)
- virtual PressureVector getWossPressure (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, const Time &time_value)
- virtual PressureVector getWossPressure (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, double time_value=0.0)
- virtual TimeArr * getWossTimeArr (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)=0
- virtual TimeArr * getWossTimeArr (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, double time_value=0.0)

- virtual TimeArrVector getWossTimeArr (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, const Time &time_value)
- virtual TimeArrVector getWossTimeArr (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, double time_value=0.0)
- virtual bool reset ()=0
- virtual bool timeEvolve (const Time &time_value)=0
- WossManager & setWossCreator (const WossCreator *const ptr)
- void **setTimeEvolutionActiveFlag** (bool flag)
- void **setDebugFlag** (bool flag)
- const WossCreator *const **getWossCreator** ()
- bool **getTimeEvolutionActiveFlag** ()
- bool **getDebugFlag** ()

Protected Member Functions

- virtual Woss *const getWoss (const CoordZ &tx, const CoordZ &rx, double start_frequency, double end_↩ frequency)=0

Protected Attributes

- const WossCreator * woss_creator
- bool debug
- bool **is_time_evolution_active**

Static Protected Attributes

- static const Time **NO_EVOLUTION_TIME** = Time(1,1,1901,1,1,1)

13.91.1 Detailed Description

Abstract class that interfaces Pressure or TimeArr requests from user layer.

WossManager interfaces the user with the whole library. Every request for Pressure or TimeArr should be done to this class. Logical inteligence for planning CPU load, multi-frequency and multi-run channel simulations should be placed in this inheritance chain

13.91.2 Constructor & Destructor Documentation

13.91.2.1 WossManager() `WossManager::WossManager ( )`

WossManager default constructor

13.91.3 Member Function Documentation

13.91.3.1 eraseActiveWoss()

```
virtual WossManager & woss::WossManager::eraseActiveWoss (
        const CoordZ & tx,
        const CoordZ & rx,
        double start_frequency,
        double end_frequency )  [pure virtual]
```

Deletes a woss::Woss object for given params

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

reference to ∗this

Implemented in woss::WossManagerSimple< WMResDb >.

13.91.3.2 getActiveWoss()

```
const Woss & WossManager::getActiveWoss (
    const CoordZ & tx,
    const CoordZ & rx,
    double start_frequency,
    double end_frequency ) const  [virtual]
```

Returns a const reference to a valid and properly initialized woss::Woss object.

Parameters

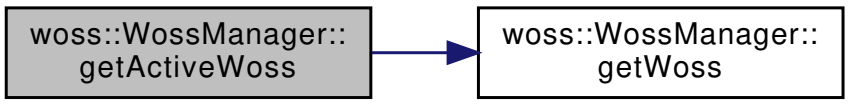
tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

const reference to a valid woss::Woss object

References getWoss().

Here is the call graph for this function:



13.91.3.3 getWoss()

```
virtual Woss *const woss::WossManager::getWoss (
    const CoordZ & tx,
    const CoordZ & rx,
    double start_frequency,
    double end_frequency )  [protected], [pure virtual]
```

Returns a pointer to a properly initialized Woss, for storage purposes.

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

pointer to a valid Woss object

Implemented in woss::WossManagerSimple< WMResDb >.

Referenced by getActiveWoss(), woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), and getWossTimeArr().

13.91.3.4 getWossPressure() [1/4]

```
Pressure * WossManager::getWossPressure (
        const CoordZ & tx_coordz,
        const CoordZ & rx_coordz,
        double start_frequency,
        double end_frequency,
        const Time & time_value )  [pure virtual]
```

Returns a valid Pressure* for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

Returns

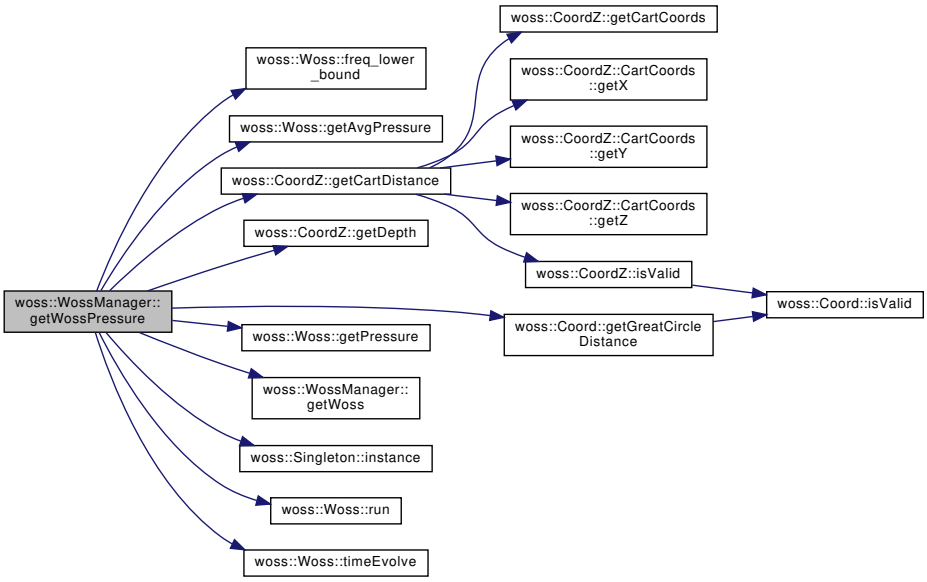
valid Pressure* pointer

Implemented in woss::WossManagerResDb, and woss::WossManagerResDbMT.

References debug, woss::Woss::freq_lower_bound(), woss::Woss::getAvgPressure(), woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::Coord::getGreatCircleDistance(), woss::Woss::getPressure(), getWoss(), woss::Singleton< T >::instance(), woss::Woss::run(), and woss::Woss::timeEvolve().

Referenced by WossMPropagation::computeGain(), getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), and woss::WMSMTcreateThreadPressure().

Here is the call graph for this function:



13.91.3.5 getWossPressure() [2/4]

```
Pressure * WossManager::getWossPressure (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    double time_value = 0.0 )  [virtual]
```

Returns a valid Pressure* for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

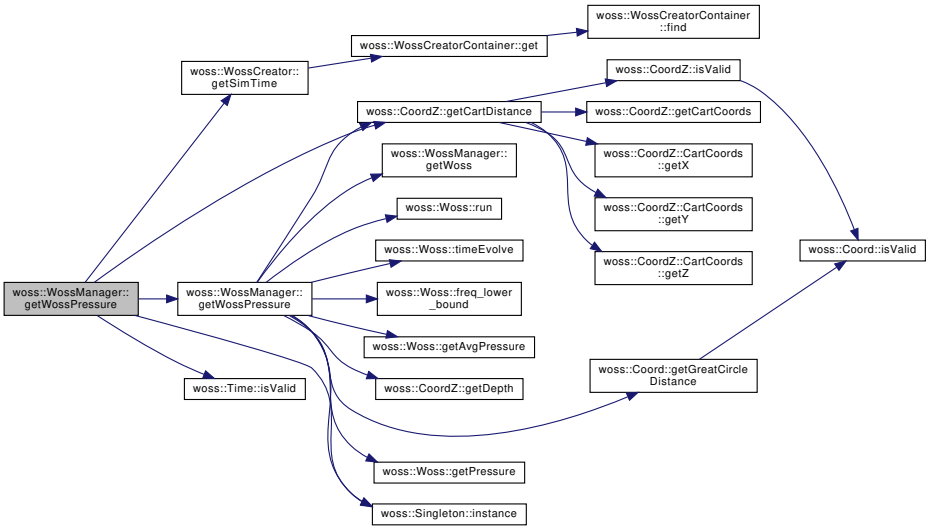
Returns

valid Pressure* pointer

Reimplemented in woss::WossManagerResDbMT.

References debug, woss::CoordZ::getCartDistance(), woss::WossCreator::getSimTime(), getWossPressure(), woss::Singleton< T >::instance(), woss::Time::isValid(), and woss_creator.

Here is the call graph for this function:



13.91.3.6 getWossPressure() [3/4]

```
PressureVector WossManager::getWossPressure (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [virtual]
```

Returns a valid vector of Pressure∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

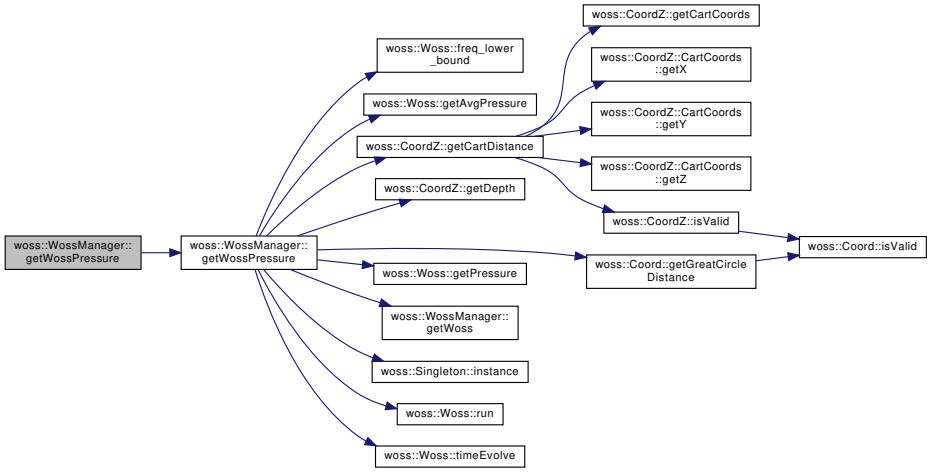
Returns

valid PressureVector

Reimplemented in woss::WossManagerResDbMT.

References getWossPressure().

Here is the call graph for this function:



13.91.3.7 getWossPressure() [4/4]

```
PressureVector WossManager::getWossPressure (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    double time_value = 0.0 )  [virtual]
```

Returns a valid vector of Pressure∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	seconds after start time

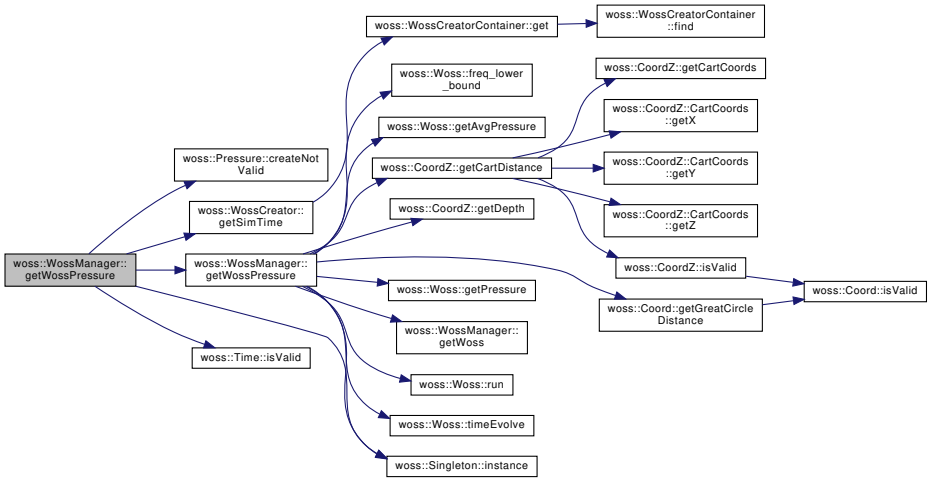
Returns

valid PressureVector

Reimplemented in woss::WossManagerResDbMT.

References woss::Pressure::createNotValid(), debug, woss::WossCreator::getSimTime(), getWossPressure(), woss::Singleton< T >::instance(), woss::Time::isValid(), and woss_creator.

Here is the call graph for this function:



13.91.3.8 getWossTimeArr() [1/4]

```
TimeArr * WossManager::getWossTimeArr (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [pure virtual]
```

Returns a valid TimeArr* for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

Returns

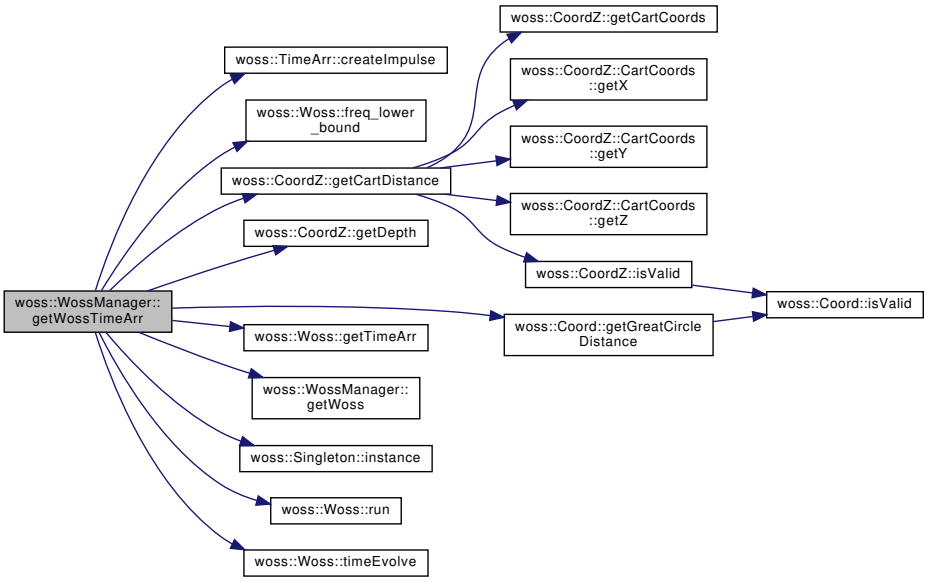
valid TimeArr* pointer

Implemented in woss::WossManagerResDb, and woss::WossManagerResDbMT.

References woss::TimeArr::createImpulse(), debug, woss::Woss::freq_lower_bound(), woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::Coord::getGreatCircleDistance(), woss::Woss::getTimeArr(), getWoss(), woss::Singleton< T >::instance(), woss::Woss::run(), and woss::Woss::timeEvolve().

Referenced by getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), and woss::WMSMTcreateThreadTimeArr().

Here is the call graph for this function:



13.91.3.9 getWossTimeArr() [2/4]

```
TimeArr * WossManager::getWossTimeArr (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    double time_value = 0.0 )  [virtual]
```

Returns a valid TimeArr∗ for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

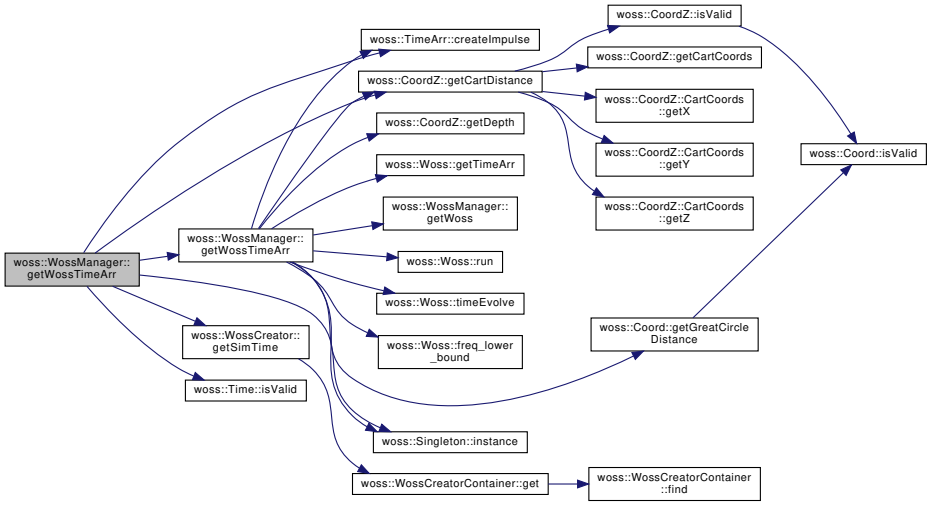
Returns

valid TimeArr∗ pointer

Reimplemented in woss::WossManagerResDbMT.

References woss::TimeArr::createImpulse(), debug, woss::CoordZ::getCartDistance(), woss::WossCreator::getSimTime(), getWossTimeArr(), woss::Singleton< T >::instance(), woss::Time::isValid(), and woss_creator.

Here is the call graph for this function:



13.91.3.10 getWossTimeArr() [3/4]

```
TimeArrVector WossManager::getWossTimeArr (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [virtual]
```

Returns a valid vector of TimeArr∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a Time object

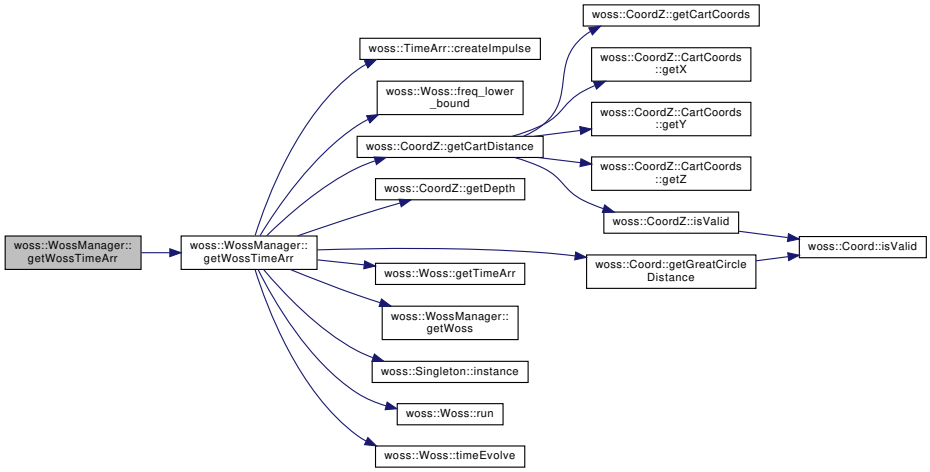
Returns

valid TimeArrVector

Reimplemented in woss::WossManagerResDbMT.

References getWossTimeArr().

Here is the call graph for this function:



13.91.3.11 getWossTimeArr() [4/4]

```
TimeArrVector WossManager::getWossTimeArr (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    double time_value = 0.0 )  [virtual]
```

Returns a valid vector of TimeArr∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

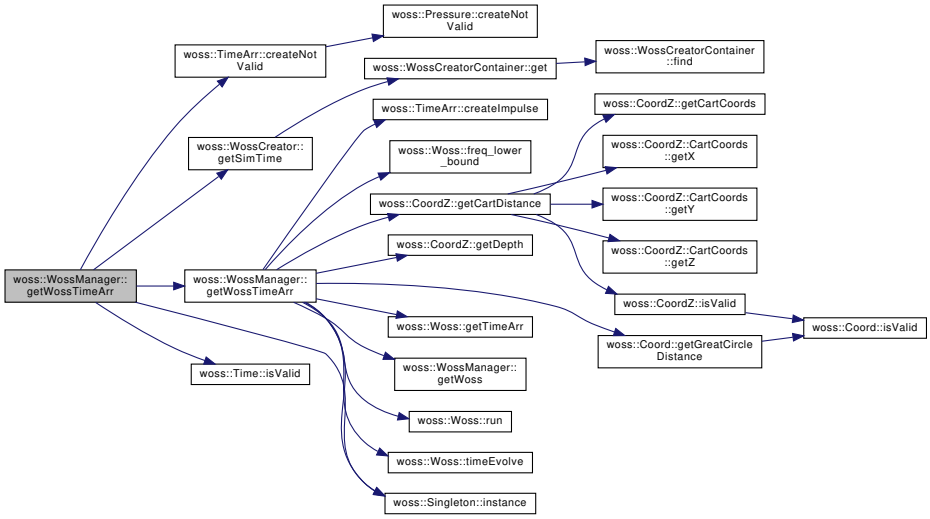
Returns

valid TimeArrVector

Reimplemented in woss::WossManagerResDbMT.

References woss::TimeArr::createNotValid(), debug, woss::WossCreator::getSimTime(), getWossTimeArr(), woss::Singleton< T >::instance(), woss::Time::isValid(), and woss_creator.

Here is the call graph for this function:



13.91.3.12 reset() `virtual bool woss::WossManager::reset ( )  [pure virtual]`

Deletes all created Woss instances

Returns

true if method was successful, *false* otherwise

Implemented in woss::WossManagerSimple< WMResDb >.

13.91.3.13 setWossCreator() `WossManager & woss::WossManager::setWossCreator ( const WossCreator *const ptr )  [inline]`

Sets a pointer to a WossCreator instance, for Woss creation purposes

Parameters

ptr	const pointer to a const WossCreator instance

Returns

reference to ∗**this**

References woss_creator.

Referenced by woss::WossController::initialize().

13.91.3.14 timeEvolve() `virtual bool woss::WossManager::timeEvolve ( const Time & time_value ) [pure virtual]`

Performs a time evoulion of all time-dependant parameters of all created Woss instances

Parameters

time_value	const reference to a valid Time object

Returns

true if method was successful, *false* otherwise

Implemented in woss::WossManagerSimple< WMResDb >.

13.91.4 Member Data Documentation

13.91.4.1 debug `bool woss::WossManager::debug [protected]`

Debug flag

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), getWossTimeArr(), woss::WossManagerResDbMT::initThreadVars(), woss::WossManagerResDbMT::popThreadParamIndex(), woss::WMSMTcreateThreadPressure(), and woss::WMSMTcreateThreadTimeArr().

13.91.4.2 woss_creator `const WossCreator* woss::WossManager::woss_creator [protected]`

Const pointer to a WossCreator instance, for Woss creation purposes

Referenced by woss::WossManagerResDb::getWossPressure(), woss::WossManagerResDbMT::getWossPressure(), getWossPressure(), woss::WossManagerResDb::getWossTimeArr(), woss::WossManagerResDbMT::getWossTimeArr(), getWossTimeArr(), and setWossCreator().

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

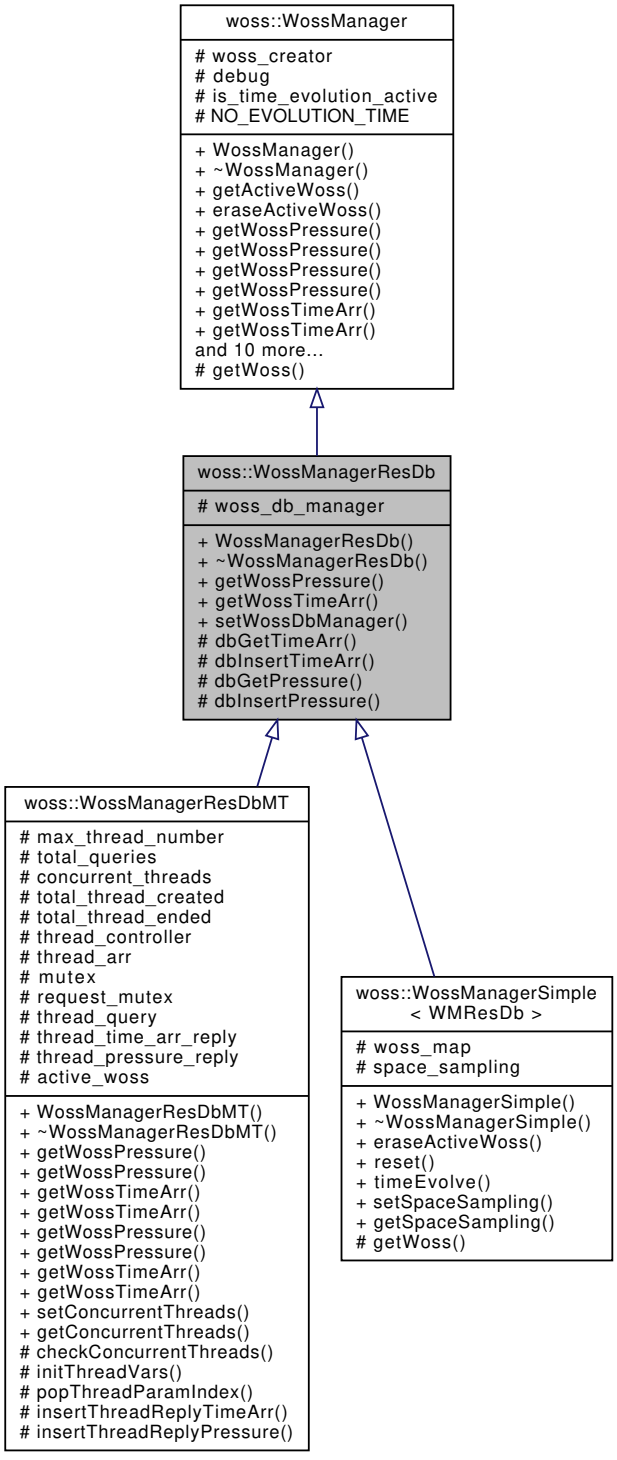
- woss/woss-manager.h
- woss/woss-manager.cpp

13.92 woss::WossManagerResDb Class Reference

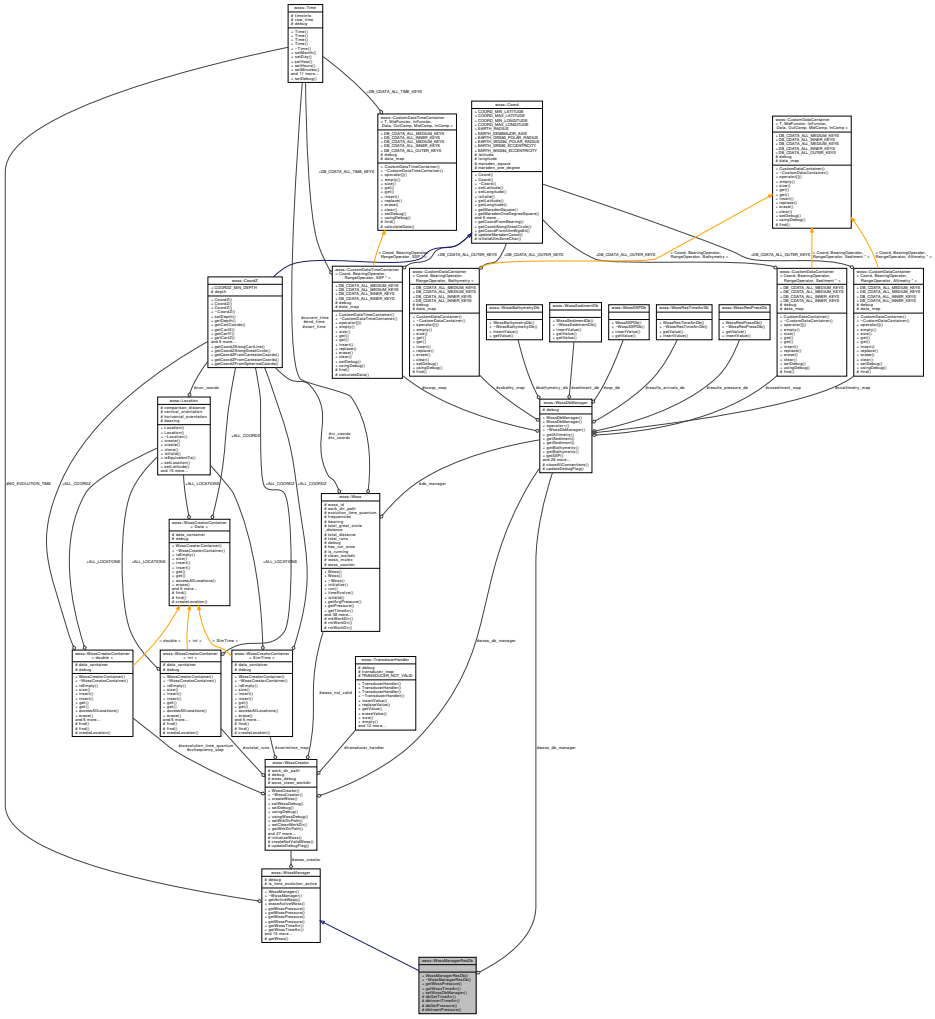
Abstract class that implements WossManager. It adds computed results dbs control.

`#include <woss-manager.h>`

Inheritance diagram for woss::WossManagerResDb:



Collaboration diagram for woss::WossManagerResDb:



Public Member Functions

- virtual Pressure * getWossPressure (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)
- virtual TimeArr * getWossTimeArr (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)
- WossManagerResDb & setWossDbManager (const WossDbManager *const ptr)

Protected Member Functions

- TimeArr * dbGetTimeArr (const CoordZ &tx, const CoordZ &rx, double frequency, const Time &time_value) const
- void dbInsertTimeArr (const CoordZ &tx, const CoordZ &rx, double frequency, const Time &time_value, const TimeArr &channel) const
- Pressure * dbGetPressure (const CoordZ &tx, const CoordZ &rx, double frequency, const Time &time_value) const
- void dbInsertPressure (const CoordZ &tx, const CoordZ &rx, double frequency, const Time &time_value, const Pressure &press) const

Protected Attributes

- const WossDbManager ∗ woss_db_manager

Additional Inherited Members

13.92.1 Detailed Description

Abstract class that implements WossManager. It adds computed results dbs control.

WossManagerResDb adds control over optional computed dbs control. If dbs are present and valid requested TimeArr or Pressure is returned, no channel simulator is run

13.92.2 Member Function Documentation

13.92.2.1 dbGetPressure()

```
Pressure * woss::WossManagerResDb::dbGetPressure (
    const CoordZ & tx,
    const CoordZ & rx,
    double frequency,
    const Time & time_value ) const  [inline], [protected]
```

Returns a Pressure∗ from a WossPressureDb for given parameters. **User is responsible of pointer's ownership**

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
frequency	frequency [Hz]
time_value	const reference to a valid Time oject

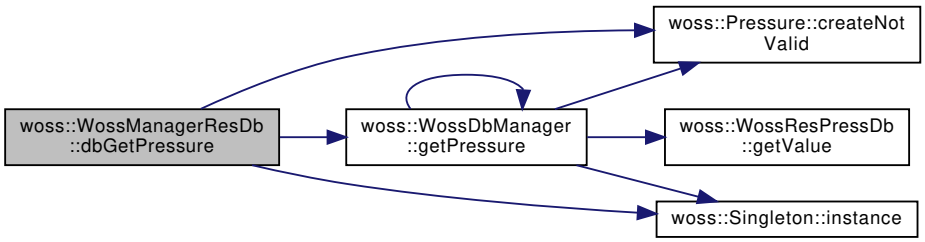
Returns

Pressure value

References woss::Pressure::createNotValid(), woss::WossDbManager::getPressure(), woss::Singleton< T >::instance(), and woss_db_manager.

Referenced by getWossPressure(), and woss::WossManagerResDbMT::getWossPressure().

Here is the call graph for this function:



13.92.2.2 dbGetTimeArr()

```
TimeArr * woss::WossManagerResDb::dbGetTimeArr (
    const CoordZ & tx,
    const CoordZ & rx,
    double frequency,
    const Time & time_value ) const  [inline], [protected]
```

Returns a TimeArr∗ from a WossResTimeArrDb for given parameters. **User is responsible of pointer's ownership**

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
frequency	frequency [Hz]
time_value	const reference to a valid Time oject

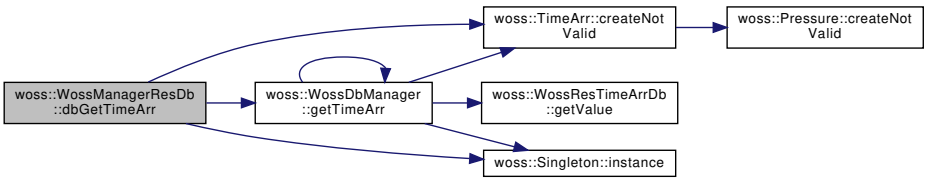
Returns

TimeArr value

References woss::TimeArr::createNotValid(), woss::WossDbManager::getTimeArr(), woss::Singleton< T >::instance(), and woss_db_manager.

Referenced by getWossTimeArr(), and woss::WossManagerResDbMT::getWossTimeArr().

Here is the call graph for this function:



13.92.2.3 dbInsertPressure()

```
void woss::WossManagerResDb::dbInsertPressure (
    const CoordZ & tx,
    const CoordZ & rx,
    double frequency,
    const Time & time_value,
    const Pressure & press ) const  [inline], [protected]
```

Inserts a Pressure in a WossResPressureDb

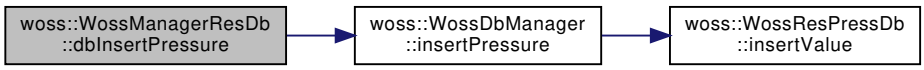
Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
frequency	frequency [Hz]
time_value	const reference to a valid Time oject
press	const reference to a valid Pressure to be inserted

References woss::WossDbManager::insertPressure(), and woss_db_manager.

Referenced by getWossPressure(), and woss::WossManagerResDbMT::getWossPressure().

Here is the call graph for this function:



13.92.2.4 dbInsertTimeArr()

```
void woss::WossManagerResDb::dbInsertTimeArr (
    const CoordZ & tx,
    const CoordZ & rx,
    double frequency,
    const Time & time_value,
    const TimeArr & channel ) const  [inline], [protected]
```

Inserts a TimeArr in a WossResTimeArrDb

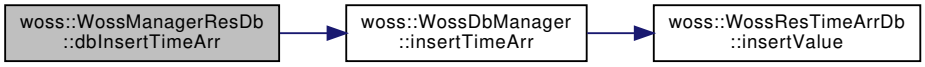
Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
frequency	frequency [Hz]
time_value	const reference to a valid Time oject
channel	const reference to a valid TimeArr to be inserted

References woss::WossDbManager::insertTimeArr(), and woss_db_manager.

Referenced by getWossTimeArr(), and woss::WossManagerResDbMT::getWossTimeArr().

Here is the call graph for this function:



13.92.2.5 getWossPressure()

```
Pressure * WossManagerResDb::getWossPressure (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [virtual]
```

Returns a valid Pressure for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

Returns

valid Pressure value

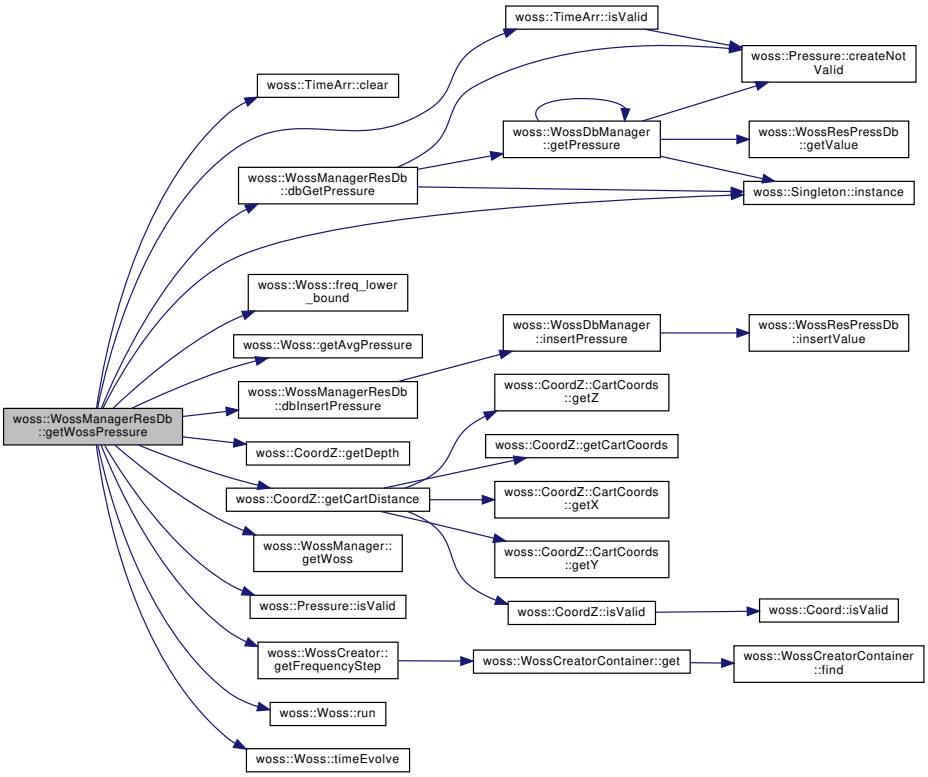
Implements woss::WossManager.

Reimplemented in woss::WossManagerResDbMT.

References woss::TimeArr::clear(), dbGetPressure(), dbInsertPressure(), woss::WossManager::debug, woss::Woss::freq_lower_bour
woss::Woss::getAvgPressure(), woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::WossCreator::getFrequencyStep
woss::WossManager::getWoss(), woss::Singleton< T >::instance(), woss::Pressure::isValid(), woss::TimeArr::isValid(),
woss::Woss::run(), woss::Woss::timeEvolve(), and woss::WossManager::woss_creator.

Referenced by woss::WossManagerResDbMT::getWossPressure().

Here is the call graph for this function:



13.92.2.6 getWossTimeArr()

```
TimeArr * WossManagerResDb::getWossTimeArr (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [virtual]
```

Returns a valid Pressure for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

Returns

valid Pressure value Returns a valid TimeArr for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time& object

Returns

valid TimeArr value

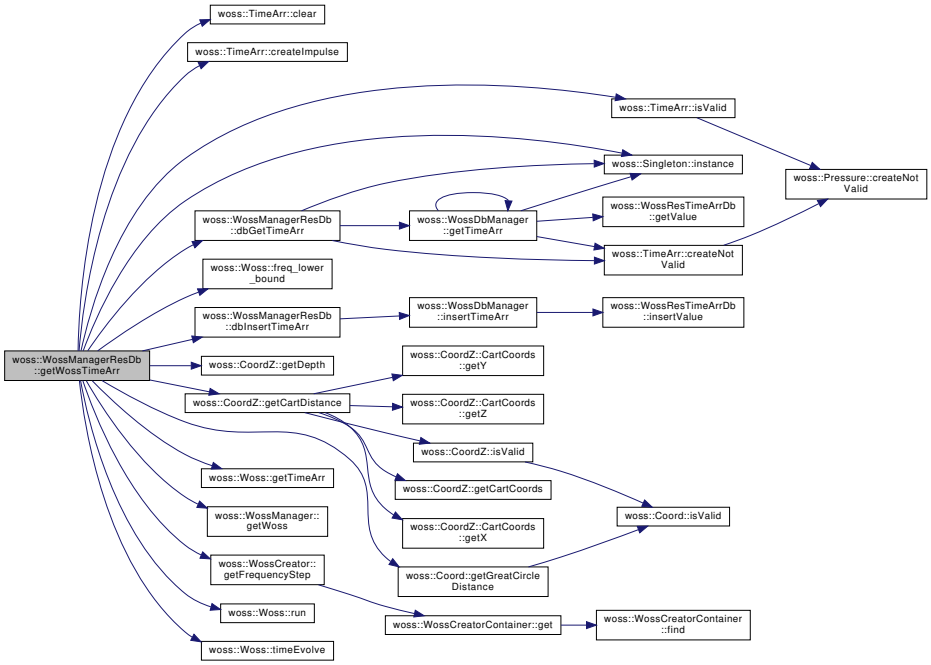
Implements woss::WossManager.

Reimplemented in woss::WossManagerResDbMT.

References woss::TimeArr::clear(), woss::TimeArr::createImpulse(), dbGetTimeArr(), dbInsertTimeArr(), woss::WossManager::debug, woss::Woss::freq_lower_bound(), woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::WossCreator::getFrequencyStep(), woss::Coord::getGreatCircleDistance(), woss::Woss::getTimeArr(), woss::WossManager::getWoss(), woss::Singleton< T >::instance(), woss::TimeArr::isValid(), woss::Woss::run(), woss::Woss::timeEvolve(), and woss::WossManager::woss_creator.

Referenced by woss::WossManagerResDbMT::getWossTimeArr().

Here is the call graph for this function:



13.92.2.7 setWossDbManager() `WossManagerResDb & woss::WossManagerResDb::setWossDbManager ( const WossDbManager *const ptr ) [inline]`

Returns a valid TimeArr for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

Returns

valid TimeArr value Sets a pointer to a WossDbManager instance, for db query purposes

Parameters

ptr	const pointer to a const WossDbManager instance

Returns

reference to ∗**this**

References woss_db_manager.

Referenced by woss::WossController::initialize().

13.92.3 Member Data Documentation

13.92.3.1 woss_db_manager `const WossDbManager* woss::WossManagerResDb::woss_db_manager [protected]`

Const pointer to a WossDbManager

Referenced by dbGetPressure(), dbGetTimeArr(), dbInsertPressure(), dbInsertTimeArr(), and setWossDbManager().

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

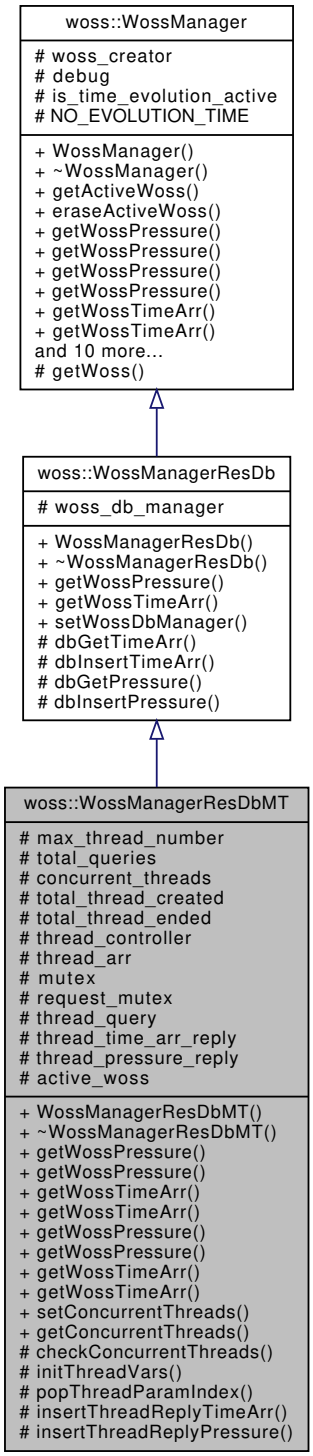
- woss/woss-manager.h
- woss/woss-manager.cpp

13.93 woss::WossManagerResDbMT Class Reference

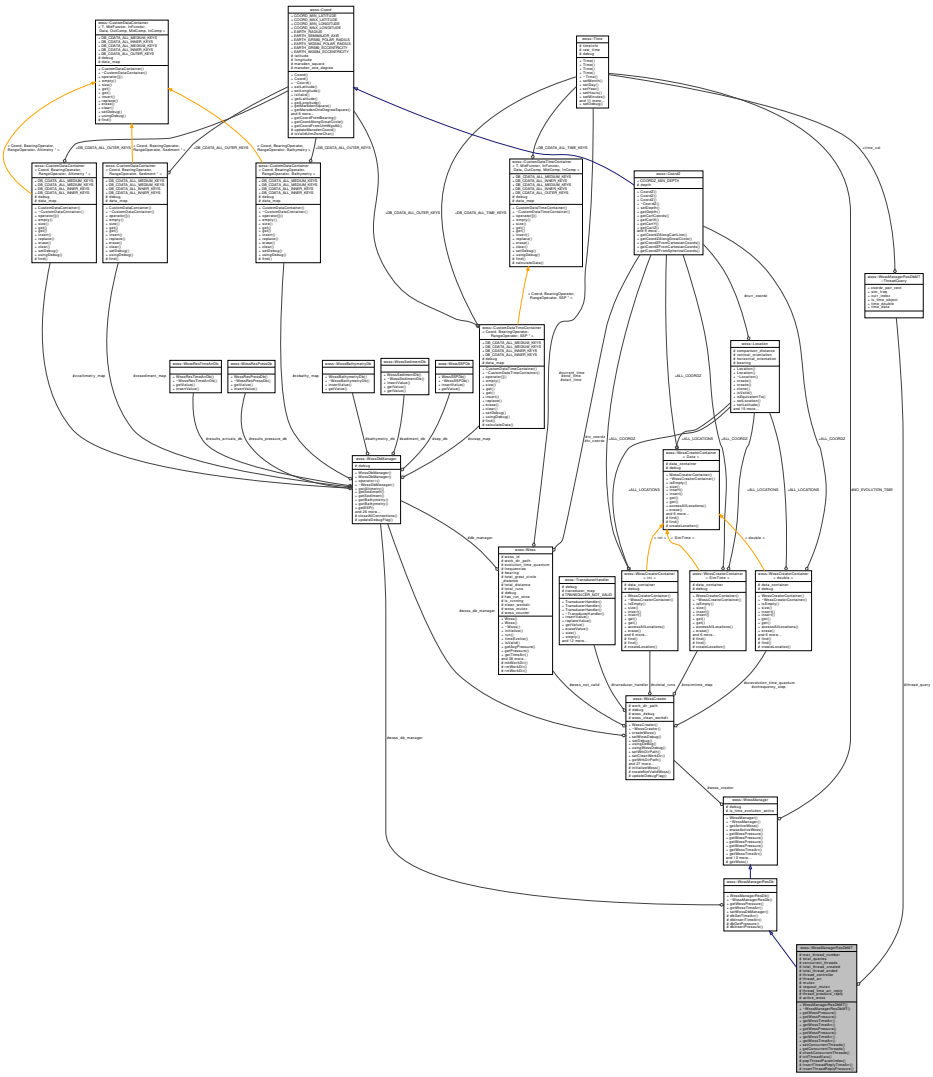
Multi-threaded extension of WossManagerResDb.

`#include <woss-manager.h>`

Inheritance diagram for woss::WossManagerResDbMT:



Collaboration diagram for woss::WossManagerResDbMT:



Classes

- struct ThreadCondSignal
- struct ThreadParam
- struct ThreadQuery

Public Member Functions

- WossManagerResDbMT ()
- virtual Pressure * getWossPressure (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)
- virtual Pressure * getWossPressure (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, double time_value)
- virtual TimeArr * getWossTimeArr (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, const Time &time_value)
- virtual TimeArr * getWossTimeArr (const CoordZ &tx_coordz, const CoordZ &rx_coordz, double start_↩ frequency, double end_frequency, double time_value)

- virtual PressureVector getWossPressure (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, const Time &time_value)
- virtual PressureVector getWossPressure (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, double time_value)
- virtual TimeArrVector getWossTimeArr (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, const Time &time_value)
- virtual TimeArrVector getWossTimeArr (const CoordZPairVect &coordinates, double start_frequency, double end_frequency, double time_value)
- void setConcurrentThreads (int number)
- int getConcurrentThreads ()

Protected Types

- typedef ::std::pair< int, ThreadParam > ThreadParamIndex
- typedef ::std::map< Woss *, ThreadCondSignal * > **ActiveWoss**
- typedef ActiveWoss::iterator **AWIter**
- typedef ActiveWoss::reverse_iterator **AWRIter**
- typedef ActiveWoss::const_iterator **AWCIter**
- typedef ActiveWoss::const_reverse_iterator **AWCRIter**

Protected Member Functions

- void checkConcurrentThreads ()
- void initThreadVars ()
- ThreadParamIndex popThreadParamIndex ()
- void insertThreadReplyTimeArr (int index, woss::TimeArr *time_arr)
- void insertThreadReplyPressure (int index, woss::Pressure *pressure)

Protected Attributes

- int max_thread_number
- int total_queries
- int concurrent_threads
- volatile int total_thread_created
- volatile int total_thread_ended
- pthread_t thread_controller
- pthread_t thread_arr [MAX_TOTAL_PTHREAD]
- pthread_spinlock_t mutex
- pthread_spinlock_t request_mutex
- ThreadQuery thread_query
- TimeArrVector thread_time_arr_reply
- PressureVector thread_pressure_reply
- ActiveWoss active_woss

Friends

- void * WMSMTcreateThreadTimeArr (void *ptr)
- void * WMSMTcreateThreadPressure (void *ptr)

Additional Inherited Members

13.93.1 Detailed Description

Multi-threaded extension of WossManagerResDb.

WossManagerResDbMT is a multi-threaded extension of WossManagerResDb. It uses the pthread library. This class is optimized for multi-processor cpu.**Don't use it if a multi-processor cpu is not installed**. *Please notice that simulation will suffer an heavy time penalty if a result db is used and Woss objects are not run*. This is due to the thread creation and synchronization overhead. Therefore no multi-thread should be used when reading already computed channel simulator data.

13.93.2 Member Typedef Documentation

13.93.2.1 ThreadParamIndex `typedef ::std::pair< int, ThreadParam > woss::WossManagerResDbMT::ThreadParamInde` `[protected]`

Type used by an active query thread

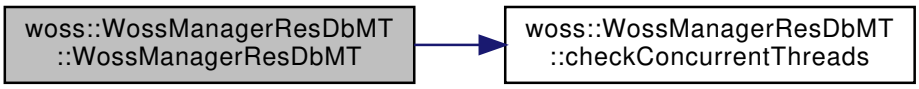
13.93.3 Constructor & Destructor Documentation

13.93.3.1 WossManagerResDbMT() `WossManagerResDbMT::WossManagerResDbMT ( )`

WossManagerResDbMT default constructor

References checkConcurrentThreads(), max_thread_number, mutex, and request_mutex.

Here is the call graph for this function:



13.93.4 Member Function Documentation

13.93.4.1 checkConcurrentThreads() `void WossManagerResDbMT::checkConcurrentThreads ( )` `[protected]`

Sets concurrent_threads valid range

References concurrent_threads, and max_thread_number.

Referenced by setConcurrentThreads(), and WossManagerResDbMT().

13.93.4.2 getConcurrentThreads() `int woss::WossManagerResDbMT::getConcurrentThreads ( ) [inline]`

Gets the number of concurrent threads

Returns

number of concurrent threads

References concurrent_threads.

13.93.4.3 getWossPressure() [1/4]

```
Pressure * WossManagerResDbMT::getWossPressure (
        const CoordZ & tx_coordz,
        const CoordZ & rx_coordz,
        double start_frequency,
        double end_frequency,
        const Time & time_value ) [virtual]
```

Returns a valid Pressure for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

Returns

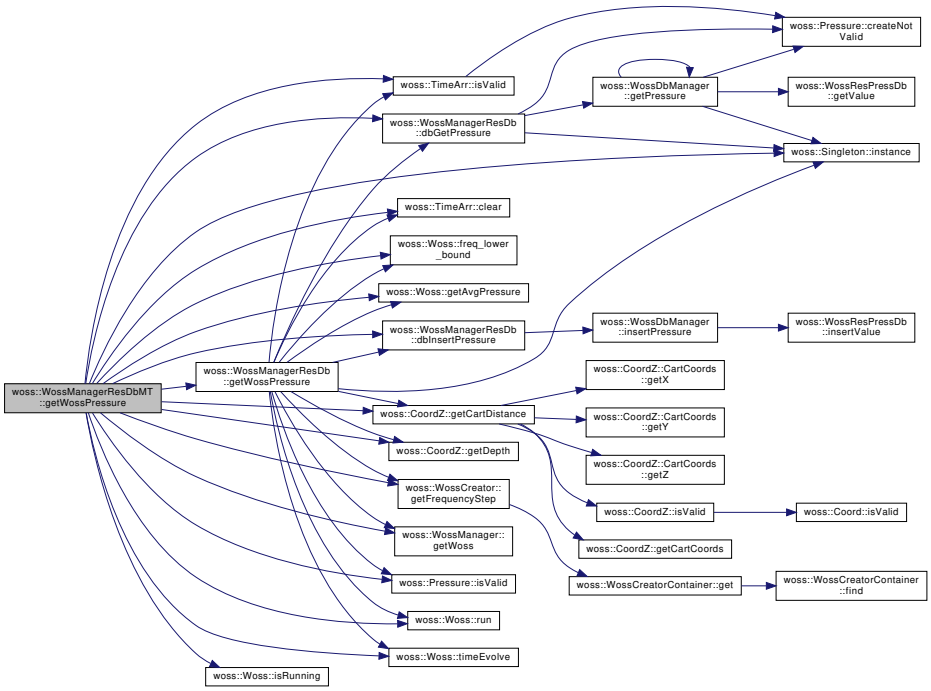
valid Pressure value

Reimplemented from woss::WossManagerResDb.

References active_woss, woss::TimeArr::clear(), concurrent_threads, woss::WossManagerResDb::dbGetPressure(), woss::WossManagerResDb::dbInsertPressure(), woss::WossManager::debug, woss::Woss::freq_lower_bound(), woss::Woss::getAvgPressure(), woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::WossCreator::getFrequencyStep, woss::WossManager::getWoss(), woss::WossManagerResDb::getWossPressure(), woss::Singleton< T >::instance(), woss::Woss::isRunning(), woss::Pressure::isValid(), woss::TimeArr::isValid(), request_mutex, woss::Woss::run(), woss::Woss::timeEvolve(), and woss::WossManager::woss_creator.

Referenced by getWossPressure().

Here is the call graph for this function:



13.93.4.4 getWossPressure() [2/4]

```
Pressure * WossManagerResDbMT::getWossPressure (
        const CoordZ & tx_coordz,
        const CoordZ & rx_coordz,
        double start_frequency,
        double end_frequency,
        double time_value )  [virtual]
```

Returns a valid Pressure for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

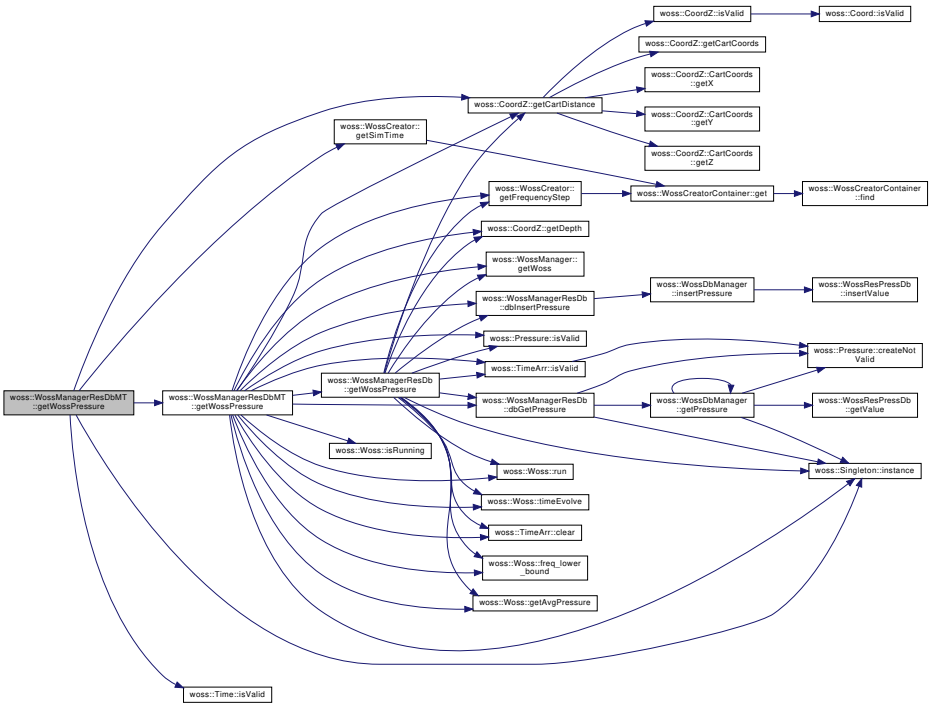
Returns

valid Pressure value

Reimplemented from woss::WossManager.

References woss::WossManager::debug, woss::CoordZ::getCartDistance(), woss::WossCreator::getSimTime(), getWossPressure(), woss::Singleton< T >::instance(), woss::Time::isValid(), request_mutex, and woss::WossManager::woss_creato

Here is the call graph for this function:



13.93.4.5 getWossPressure() [3/4]

```
PressureVector WossManagerResDbMT::getWossPressure (
        const CoordZPairVect & coordinates,
        double start_frequency,
        double end_frequency,
        const Time & time_value )  [virtual]
```

Returns a valid vector of Pressure∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

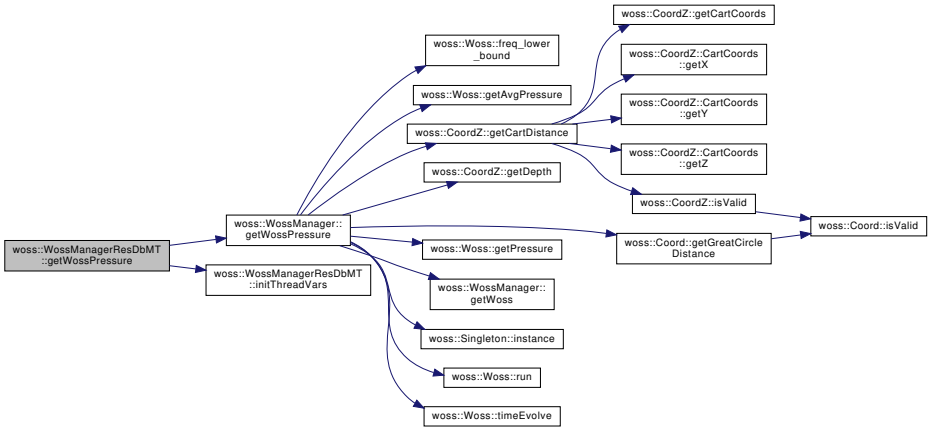
Returns

valid PressureVector

Reimplemented from woss::WossManager.

References concurrent_threads, woss::WossManager::getWossPressure(), initThreadVars(), thread_arr, thread_pressure_reply, thread_query, total_queries, total_thread_created, total_thread_ended, and WMSMTcreateThreadPressure.

Here is the call graph for this function:



13.93.4.6 getWossPressure() [4/4]

```
PressureVector WossManagerResDbMT::getWossPressure (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    double time_value )  [virtual]
```

Returns a valid vector of Pressure* for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

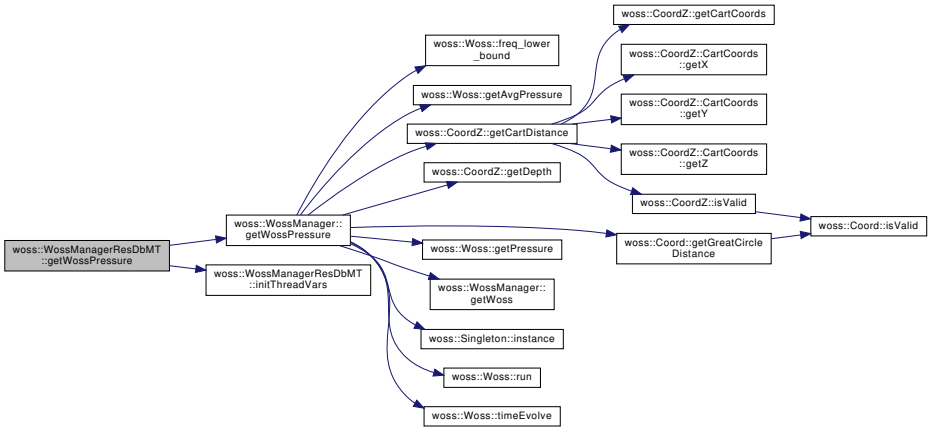
Returns

valid PressureVector

Reimplemented from woss::WossManager.

References concurrent_threads, woss::WossManager::getWossPressure(), initThreadVars(), thread_arr, thread_pressure_reply, thread_query, total_queries, total_thread_created, total_thread_ended, and WMSMTcreateThreadPressure.

Here is the call graph for this function:



13.93.4.7 getWossTimeArr() [1/4]

```
TimeArr * WossManagerResDbMT::getWossTimeArr (
    const CoordZ & tx_coordz,
    const CoordZ & rx_coordz,
    double start_frequency,
    double end_frequency,
    const Time & time_value )  [virtual]
```

Returns a valid TimeArr for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

Returns

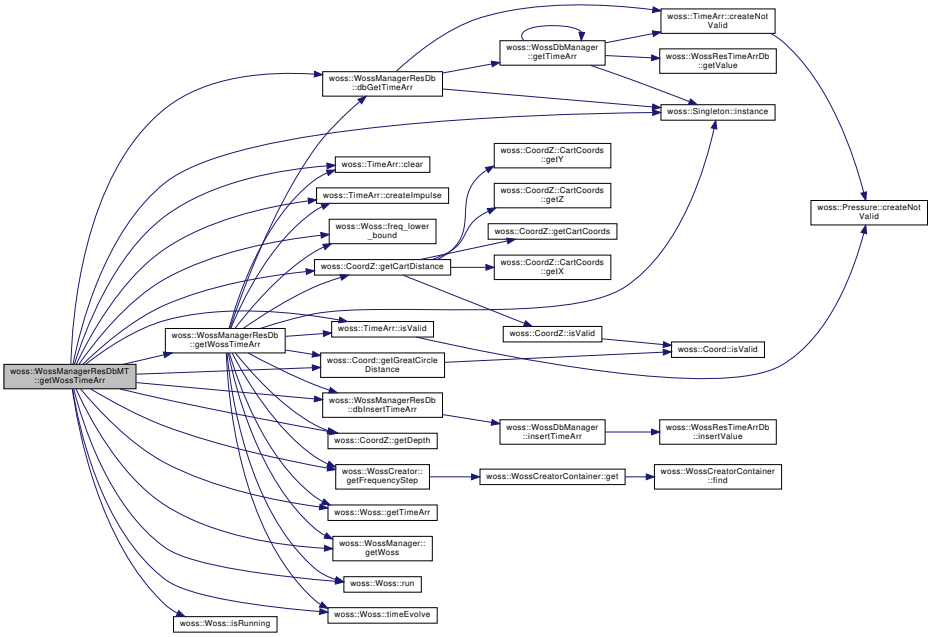
valid TimeArr value

Reimplemented from woss::WossManagerResDb.

References active_woss, woss::TimeArr::clear(), concurrent_threads, woss::TimeArr::createImpulse(), woss::WossManagerResDb::d
woss::WossManagerResDb::dbInsertTimeArr(), woss::WossManager::debug, woss::Woss::freq_lower_bound(),
woss::CoordZ::getCartDistance(), woss::CoordZ::getDepth(), woss::WossCreator::getFrequencyStep(), woss::Coord::getGreatCircleD
woss::Woss::getTimeArr(), woss::WossManager::getWoss(), woss::WossManagerResDb::getWossTimeArr(),
woss::Singleton< T >::instance(), woss::Woss::isRunning(), woss::TimeArr::isValid(), request_mutex, woss::Woss::run(),
woss::Woss::timeEvolve(), and woss::WossManager::woss_creator.

Referenced by getWossTimeArr().

Here is the call graph for this function:



13.93.4.8 getWossTimeArr() [2/4]

```
TimeArr * WossManagerResDbMT::getWossTimeArr (
        const CoordZ & tx_coordz,
        const CoordZ & rx_coordz,
        double start_frequency,
        double end_frequency,
        double time_value )  [virtual]
```

Returns a valid TimeArr for given parameters

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

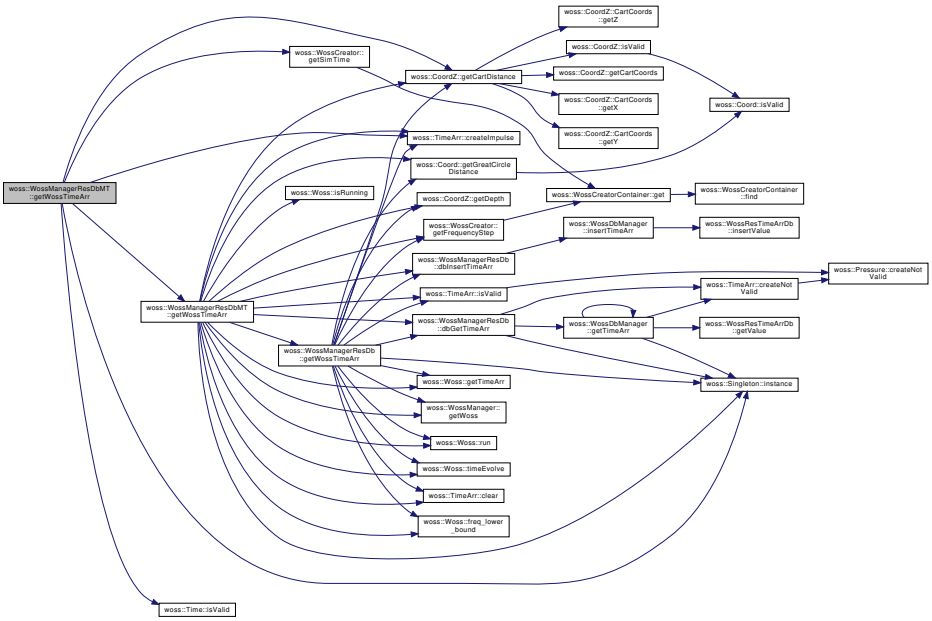
Returns

valid TimeArr value

Reimplemented from woss::WossManager.

References woss::TimeArr::createImpulse(), woss::WossManager::debug, woss::CoordZ::getCartDistance(), woss::WossCreator::getSimTime(), getWossTimeArr(), woss::Singleton< T >::instance(), woss::Time::isValid(), request_mutex, and woss::WossManager::woss_creator.

Here is the call graph for this function:



13.93.4.9 getWossTimeArr() [3/4]

```
TimeArrVector WossManagerResDbMT::getWossTimeArr (
        const CoordZPairVect & coordinates,
        double start_frequency,
        double end_frequency,
        const Time & time_value )  [virtual]
```

Returns a valid vector of TimeArr* for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	const reference to a valid Time object

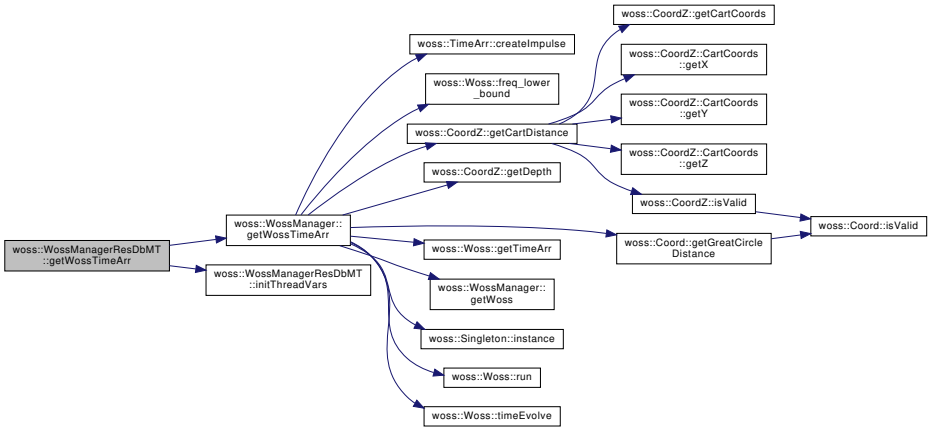
Returns

valid TimeArrVector

Reimplemented from woss::WossManager.

References concurrent_threads, woss::WossManager::getWossTimeArr(), initThreadVars(), thread_arr, thread_query, thread_time_arr_reply, total_queries, total_thread_created, total_thread_ended, and WMSMTcreateThreadTimeArr.

Here is the call graph for this function:



13.93.4.10 getWossTimeArr() [4/4]

```
TimeArrVector WossManagerResDbMT::getWossTimeArr (
    const CoordZPairVect & coordinates,
    double start_frequency,
    double end_frequency,
    double time_value )  [virtual]
```

Returns a valid vector of TimeArr∗ for given parameters

Parameters

coordinates	const reference to a valid CoordZPairVect
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
time_value	number of seconds after start time

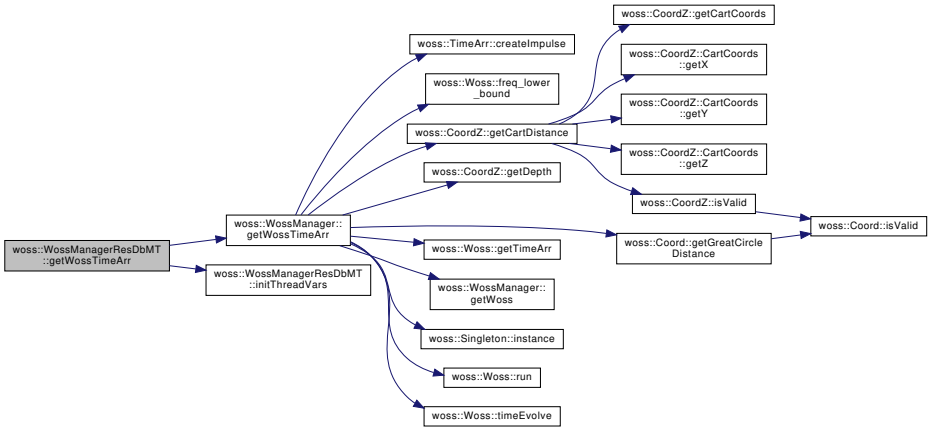
Returns

valid TimeArrVector

Reimplemented from woss::WossManager.

References concurrent_threads, woss::WossManager::getWossTimeArr(), initThreadVars(), thread_arr, thread_query, thread_time_arr_reply, total_queries, total_thread_created, total_thread_ended, and WMSMTcreateThreadTimeArr.

Here is the call graph for this function:



13.93.4.11 initThreadVars() `void WossManagerResDbMT::initThreadVars ( ) [protected]`

Initializes variable for current query round

References concurrent_threads, woss::WossManager::debug, thread_query, total_queries, total_thread_created, and total_thread_ended.

Referenced by getWossPressure(), and getWossTimeArr().

13.93.4.12 insertThreadReplyPressure() `void WossManagerResDbMT::insertThreadReplyPressure ( int index, woss::Pressure * pressure ) [protected]`

Insert the given Pressure pointer in the PressureVector reply at given index

Parameters

index	vector index
pressure	pointer to a heap-created Pressure

References mutex, and thread_pressure_reply.

Referenced by woss::WMSMTcreateThreadPressure().

13.93.4.13 insertThreadReplyTimeArr() `void WossManagerResDbMT::insertThreadReplyTimeArr ( int index, woss::TimeArr * time_arr ) [protected]`

Insert the given TimeArr pointer in the TimeArrVector reply at given index

Parameters

index	vector index
time_arr	pointer to a heap-created TimeArr

References mutex, and thread_time_arr_reply.

Referenced by woss::WMSMTcreateThreadTimeArr().

13.93.4.14 popThreadParamIndex() `WossManagerResDbMT::ThreadParamIndex WossManagerResDbMT::popThreadParamIndex ( ) [protected]`

Returns a valid ThreadParamIndex for a requesting thread

Returns

valid ThreadParamIndex

References woss::WossManager::debug, mutex, and thread_query.

Referenced by woss::WMSMTcreateThreadPressure(), and woss::WMSMTcreateThreadTimeArr().

13.93.4.15 setConcurrentThreads() `void woss::WossManagerResDbMT::setConcurrentThreads ( int number ) [inline]`

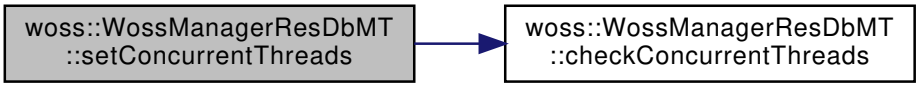
Sets the number of concurrent threads. If *number* $<$ 0 multi-threading is disabled. If *number* = 0 the thread number is automatically handled.

Parameters

number	number of concurrent threads

References checkConcurrentThreads(), and concurrent_threads.

Here is the call graph for this function:



13.93.5 Friends And Related Function Documentation

13.93.5.1 WMSMTcreateThreadPressure `void * WMSMTcreateThreadPressure ( void * ptr ) [friend]`

Function used for Pressure thread creation

Parameters

ptr	void pointer

Returns

void pointer

Referenced by getWossPressure().

13.93.5.2 WMSMTcreateThreadTimeArr `void * WMSMTcreateThreadTimeArr ( void * ptr ) [friend]`

Function used for TimeArr thread creation

Parameters

ptr	void pointer

Returns

void pointer

Referenced by getWossTimeArr().

13.93.6 Member Data Documentation

13.93.6.1 active_woss `ActiveWoss woss::WossManagerResDbMT::active_woss [protected]`

Set of current active Woss objects

Referenced by getWossPressure(), and getWossTimeArr().

13.93.6.2 concurrent_threads `int woss::WossManagerResDbMT::concurrent_threads [protected]`

Max number of concurrent threads

Referenced by checkConcurrentThreads(), getConcurrentThreads(), getWossPressure(), getWossTimeArr(), initThreadVars(), and setConcurrentThreads().

13.93.6.3 max_thread_number

`int woss::WossManagerResDbMT::max_thread_number [protected]`

Max number of created threads

Referenced by checkConcurrentThreads(), and WossManagerResDbMT().

13.93.6.4 mutex

`pthread_spinlock_t woss::WossManagerResDbMT::mutex [protected]`

Master spinlock

Referenced by insertThreadReplyPressure(), insertThreadReplyTimeArr(), popThreadParamIndex(), and WossManagerResDbMT().

13.93.6.5 request_mutex

`pthread_spinlock_t woss::WossManagerResDbMT::request_mutex [protected]`

Secondary spinlock

Referenced by getWossPressure(), getWossTimeArr(), and WossManagerResDbMT().

13.93.6.6 thread_arr

`pthread_t woss::WossManagerResDbMT::thread_arr[MAX_TOTAL_PTHREAD] [protected]`

Array of pthread ids associated to the created query threads

Referenced by getWossPressure(), and getWossTimeArr().

13.93.6.7 thread_controller

`pthread_t woss::WossManagerResDbMT::thread_controller [protected]`

pthread id associated to the creation croller thread

13.93.6.8 thread_pressure_reply

`PressureVector woss::WossManagerResDbMT::thread_pressure_reply [protected]`

The computation of current queries

Referenced by getWossPressure(), and insertThreadReplyPressure().

13.93.6.9 thread_query

`ThreadQuery woss::WossManagerResDbMT::thread_query [protected]`

Storage for current queries

Referenced by getWossPressure(), getWossTimeArr(), initThreadVars(), and popThreadParamIndex().

13.93.6.10 thread_time_arr_reply

`TimeArrVector woss::WossManagerResDbMT::thread_time_arr_reply [protected]`

The computation of current queries

Referenced by getWossTimeArr(), and insertThreadReplyTimeArr().

13.93.6.11 total_queries

`int woss::WossManagerResDbMT::total_queries [protected]`

Number of queries

Referenced by getWossPressure(), getWossTimeArr(), and initThreadVars().

13.93.6.12 total_thread_created

`volatile int woss::WossManagerResDbMT::total_thread_created [protected]`

Number of created threads

Referenced by getWossPressure(), getWossTimeArr(), and initThreadVars().

13.93.6.13 total_thread_ended

`volatile int woss::WossManagerResDbMT::total_thread_ended [protected]`

Total number of completed threads

Referenced by getWossPressure(), getWossTimeArr(), and initThreadVars().

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

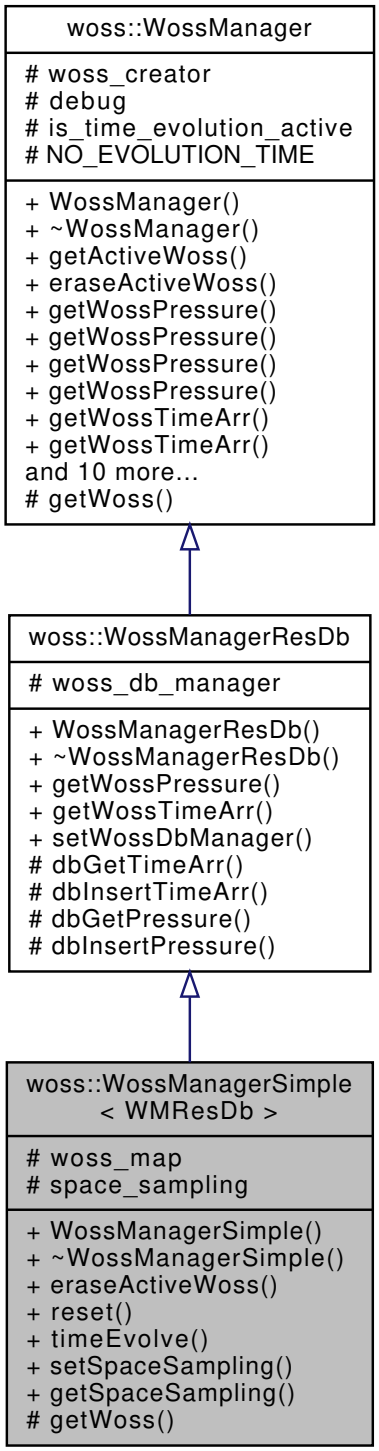
- woss/woss-manager.h
- woss/woss-manager.cpp

13.94 woss::WossManagerSimple< WMResDb > Class Template Reference

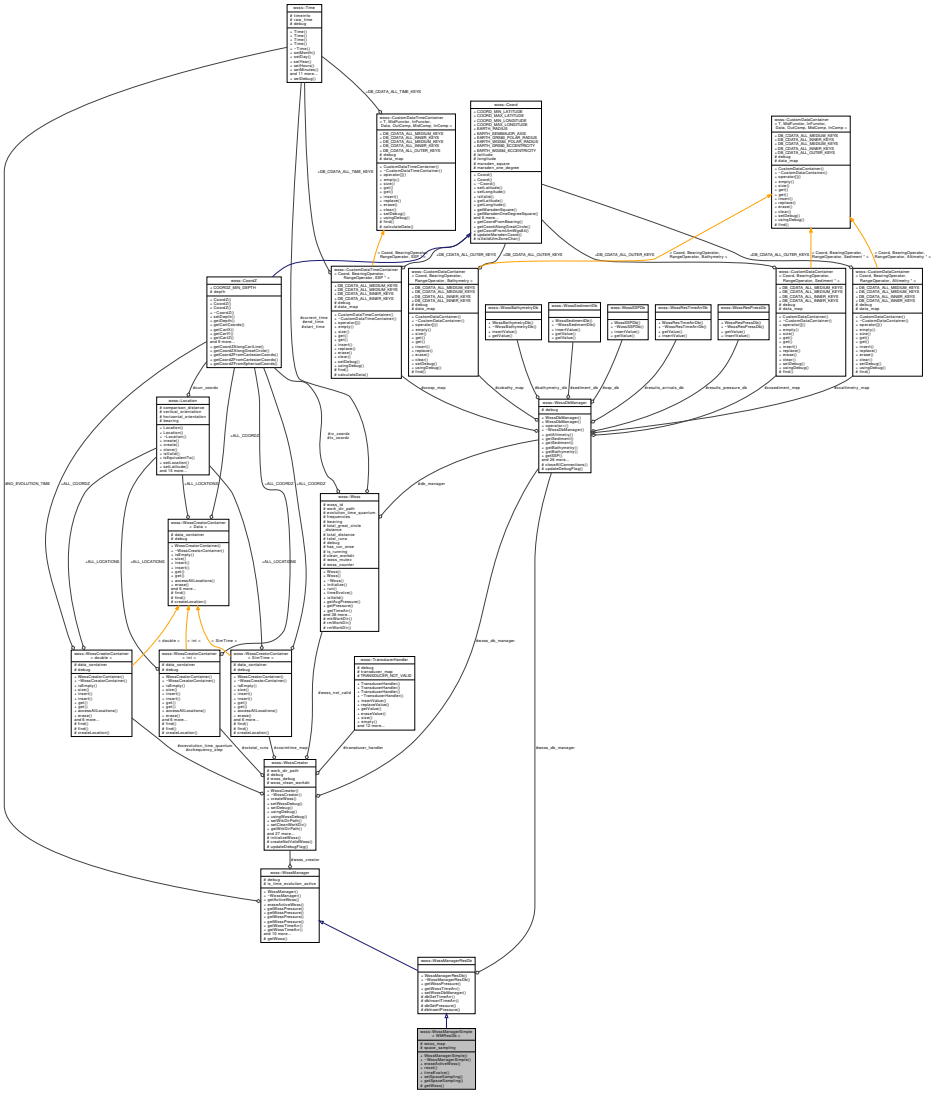
simple template extension of WossManagerResDb or WossManagerResDbMT

`#include <woss-manager-simple.h>`

Inheritance diagram for woss::WossManagerSimple< WMResDb >:



Collaboration diagram for woss::WossManagerSimple< WMResDb >:



Public Member Functions

- WossManagerSimple ()
- virtual WossManagerSimple & eraseActiveWoss (const CoordZ &tx, const CoordZ &rx, double start_↩ frequency, double end_frequency)
- virtual bool reset ()
- virtual bool timeEvolve (const Time &time_value)

Static Public Member Functions

- static void setSpaceSampling (double radius)
- static double getSpaceSampling ()

Protected Types

- typedef ::std::map< CoordZ, Woss *, CoordComparator< WossManagerSimple, CoordZ > > WossCoordZMap
- typedef WossCoordZMap::iterator **WCZIter**
- typedef WossCoordZMap::reverse_iterator **WCZRIter**
- typedef ::std::map< CoordZ, WossCoordZMap, CoordComparator< WossManagerSimple, CoordZ > > WossContainer
- typedef WossContainer::iterator **WCIter**
- typedef WossContainer::reverse_iterator **WCRIter**

Protected Member Functions

- virtual Woss *const getWoss (const CoordZ &tx, const CoordZ &rx, double start_frequency, double end_↩ frequency)

Protected Attributes

- WossContainer woss_map

Static Protected Attributes

- static double space_sampling = 0.0

13.94.1 Detailed Description

template<typename WMResDb = WossManagerResDb>
class woss::WossManagerSimple< WMResDb >

simple template extension of WossManagerResDb or WossManagerResDbMT

WossManagerSimple is a simple but functional template extension of WossManagerResDb or WossManagerResDbMT. It creates a Woss for every tx-rx pair. No memory management is done. In simulation with high mobility rate, a Woss for every receiver will be created everytime a transmitter will move, without removing old objects. **If a memory management is needed, the user should extend this class to suit his needs**.

13.94.2 Member Typedef Documentation

13.94.2.1 WossContainer `template<typename WMResDb = WossManagerResDb>`

`typedef ::std::map< CoordZ, WossCoordZMap, CoordComparator< WossManagerSimple, CoordZ > > woss::WossManagerSimple< WMResDb >::WossContainer [protected]`

Map that links a transmitter CoordZ to a WossCoordZMap

13.94.2.2 WossCoordZMap

```
template<typename WMResDb = WossManagerResDb>
typedef ::std::map< CoordZ, Woss*, CoordComparator< WossManagerSimple, CoordZ > > woss::WossManagerSimple<
WMResDb >::WossCoordZMap  [protected]
```

Map that links a receiver CoordZ to a pointer to a valid Woss object

13.94.3 Constructor & Destructor Documentation

13.94.3.1 WossManagerSimple()

```
template<typename WMResDb >
woss::WossManagerSimple< WMResDb >::WossManagerSimple
```

WossManagerSimple default constructor

13.94.4 Member Function Documentation

13.94.4.1 eraseActiveWoss()

```
template<typename WMResDb >
WossManagerSimple< WMResDb > & woss::WossManagerSimple< WMResDb >::eraseActiveWoss (
            const CoordZ & tx,
            const CoordZ & rx,
            double start_frequency,
            double end_frequency )  [virtual]
```

Deletes a woss::Woss object for given params

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

reference to ∗this

Implements woss::WossManager.

13.94.4.2 getSpaceSampling()

```
template<typename WMResDb = WossManagerResDb>
static double woss::WossManagerSimple< WMResDb >::getSpaceSampling ( )  [inline], [static]
```

Gets the radius in meters of a cartesian sphere, in which all coordinates are considered to be equivalent

Returns

radius in meters

References woss::WossManagerSimple< WMResDb >::space_sampling.

13.94.4.3 getWoss() template<typename WMResDb >

```
Woss *const woss::WossManagerSimple< WMResDb >::getWoss (
            const CoordZ & tx,
            const CoordZ & rx,
            double start_frequency,
            double end_frequency )  [protected], [virtual]
```

Returns a pointer to a properly initialized Woss, for storage purposes

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]

Returns

pointer to a valid Woss object

Implements woss::WossManager.

13.94.4.4 reset() template<typename WMResDb >

```
bool woss::WossManagerSimple< WMResDb >::reset  [virtual]
```

Deletes all created Woss instances

Returns

true if method was successful, *false* otherwise

Implements woss::WossManager.

13.94.4.5 setSpaceSampling() template<typename WMResDb = WossManagerResDb>

```
static void woss::WossManagerSimple< WMResDb >::setSpaceSampling (
            double radius )  [inline], [static]
```

Sets the radius in meters of a cartesian sphere, in which all coordinates are considered to be equivalent

Parameters

radius	radius >= 0.0 in meters

References woss::WossManagerSimple< WMResDb >::space_sampling.

13.94.4.6 timeEvolve() template<typename WMResDb >

```
bool woss::WossManagerSimple< WMResDb >::timeEvolve (
            const Time & time_value )  [virtual]
```

Performs a time evoulion of all time-dependant parameters of all created Woss instances

Parameters

time_value	const reference to a valid Time object

Returns

true if method was successful, *false* otherwise

Implements woss::WossManager.

13.94.5 Member Data Documentation

13.94.5.1 space_sampling template<typename WMResDb >

```
double woss::WossManagerSimple< WMResDb >::space_sampling = 0.0  [static], [protected]
```

The radius in meters (>= 0.0) of a cartesian sphere, in which all coordinates are considered to be equivalent

Referenced by woss::WossManagerSimple< WMResDb >::getSpaceSampling(), and woss::WossManagerSimple< WMResDb >::s

13.94.5.2 woss_map template<typename WMResDb = WossManagerResDb>

```
WossContainer woss::WossManagerSimple< WMResDb >::woss_map  [protected]
```

Map containing all created Woss objects

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

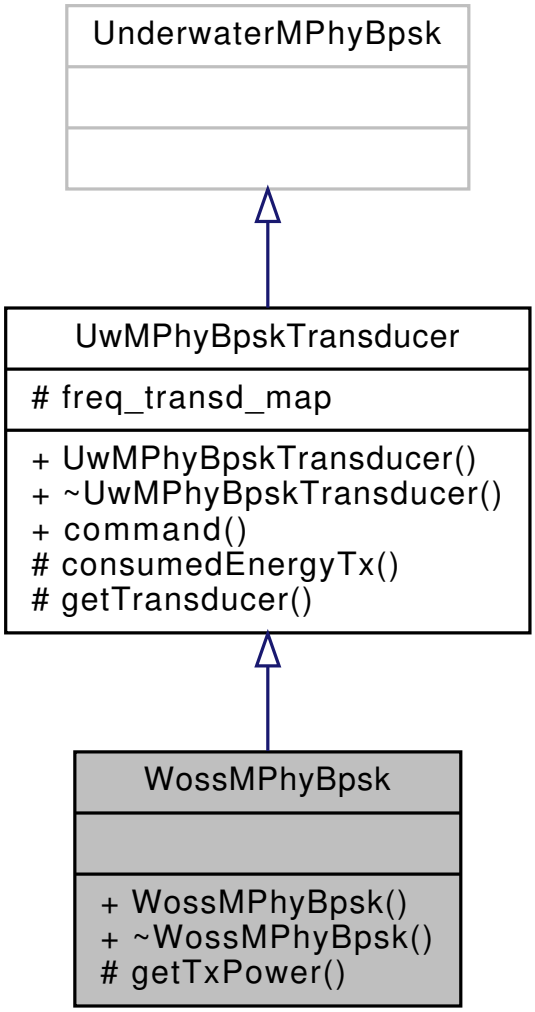
- woss/woss-manager-simple.h

13.95 WossMPhyBpsk Class Reference

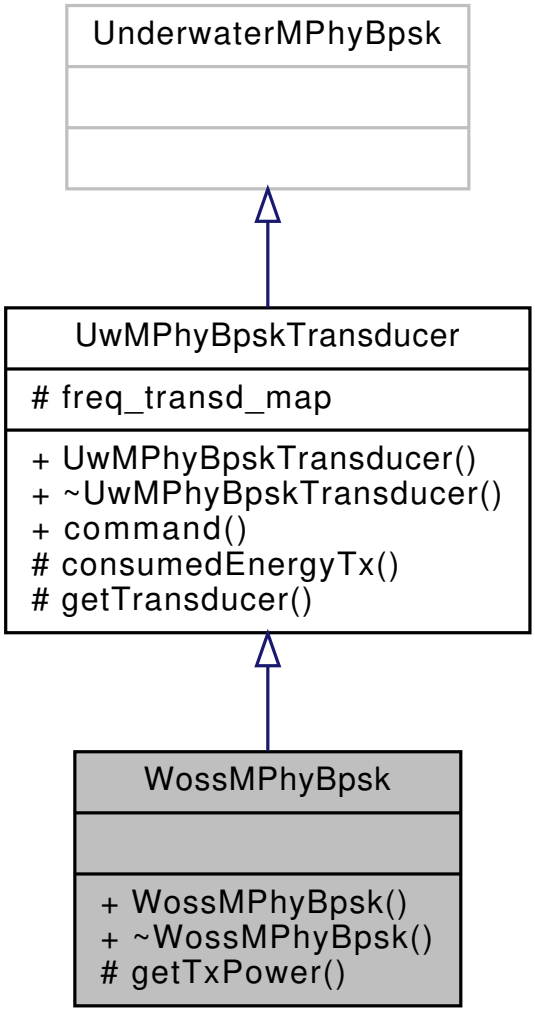
BPSK modulation class with woss::Transducer power control.

`#include <uw-woss-bpsk.h>`

Inheritance diagram for WossMPhyBpsk:



Collaboration diagram for WossMPhyBpsk:



Protected Member Functions

- virtual double getTxPower (Packet *p)

Additional Inherited Members

13.95.1 Detailed Description

BPSK modulation class with woss::Transducer power control.

WossMPhyBpsk extends UwMPhyBpskTransducer replacing old tx power control based on distance.

13.95.2 Member Function Documentation

13.95.2.1 getTxPower() `double WossMPhyBpsk::getTxPower ( Packet * p ) [protected], [virtual]`

It sends a synchronous cross-layer message asking for the current channel estimation. Upon valid answer it provides optimal power calculations.

Parameters

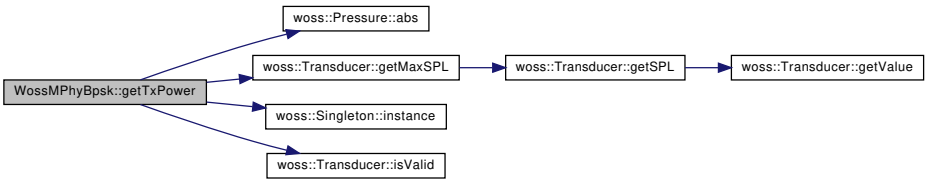
p	pointer to the current Packet being processed

Returns

Tx power in uPa

References woss::Pressure::abs(), woss::Transducer::getMaxSPL(), woss::Singleton< T >::instance(), and woss::Transducer::isValid().

Here is the call graph for this function:



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

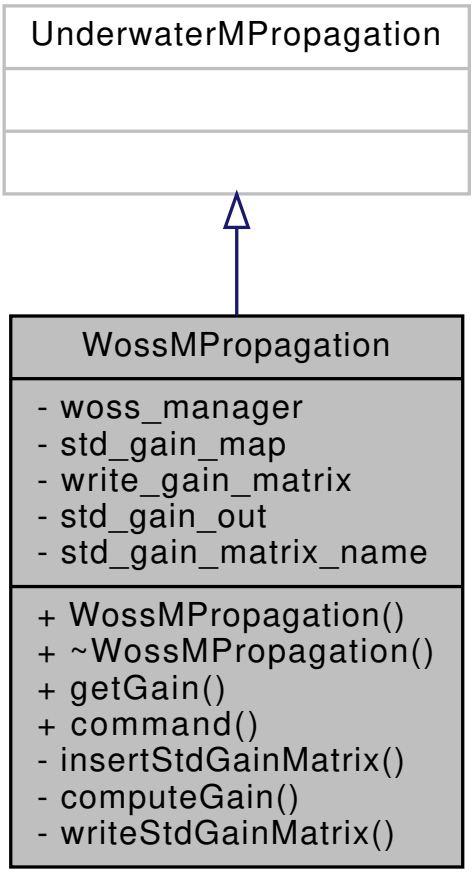
- woss_phy/uw-woss-bpsk.h
- woss_phy/uw-woss-bpsk.cpp

13.96 WossMPropagation Class Reference

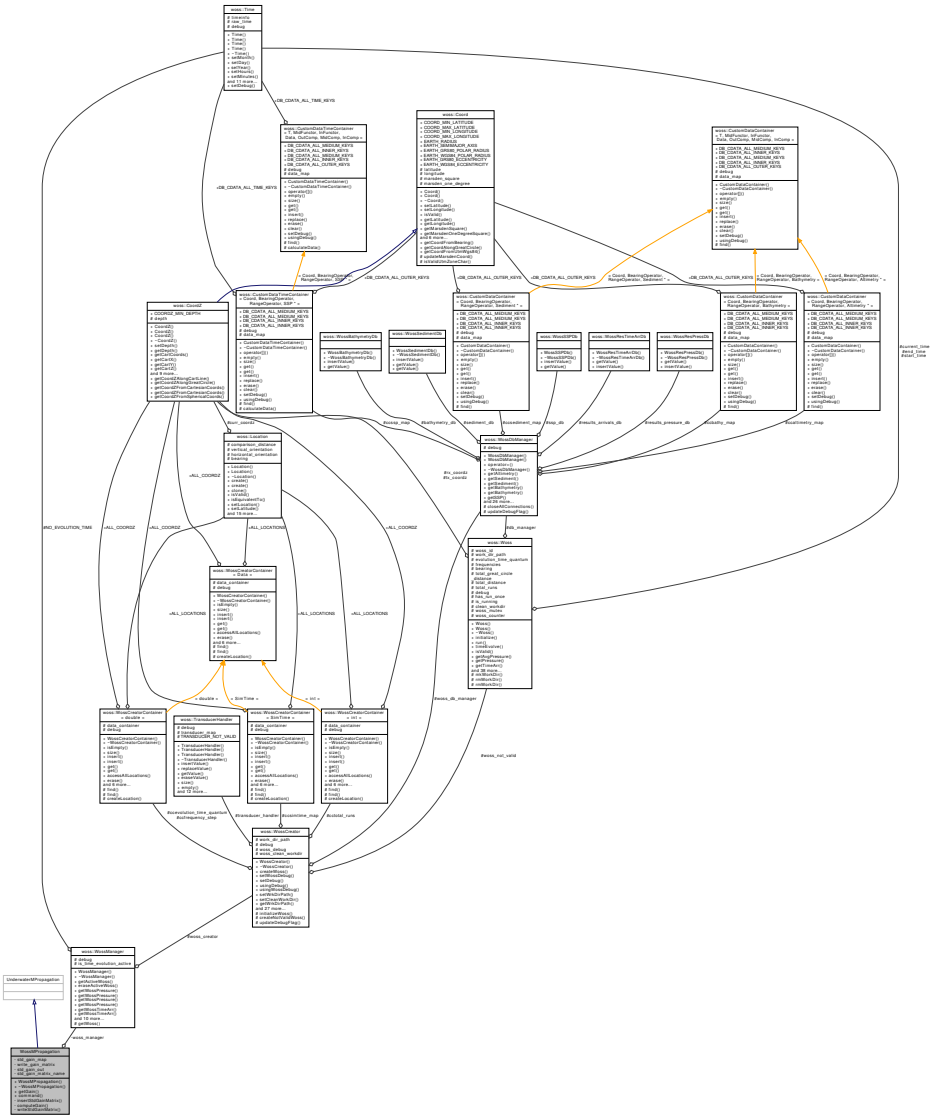
UnderwaterMPropagation class for channel calculations with WOSS.

`#include <uw-woss-mpropagation.h>`

Inheritance diagram for WossMPropagation:



Collaboration diagram for WossMPropagation:



Public Member Functions

- virtual double getGain (Packet ∗p)
- virtual int **command** (int argc, const char ∗const ∗argv)

Private Member Functions

- void **insertStdGainMatrix** (Position ∗sp, Position ∗rp, double gain)
- double computeGain (Packet ∗p)
- void **writeStdGainMatrix** ()

Private Attributes

- woss::WossManager ∗ **woss_manager**
- GainMatrix **std_gain_map**
- bool **write_gain_matrix**
- fstream **std_gain_out**
- string **std_gain_matrix_name**

13.96.1 Detailed Description

UnderwaterMPropagation class for channel calculations with WOSS.

WossMPropagation extends UnderwaterMPropagation for channel calculations with WOSS

13.96.2 Member Function Documentation

13.96.2.1 computeGain() `double WossMPropagation::computeGain ( Packet * p ) [private]`

Provides attenuation calculation with WOSS framework

Parameters

p	pointer to the current Packet being processed

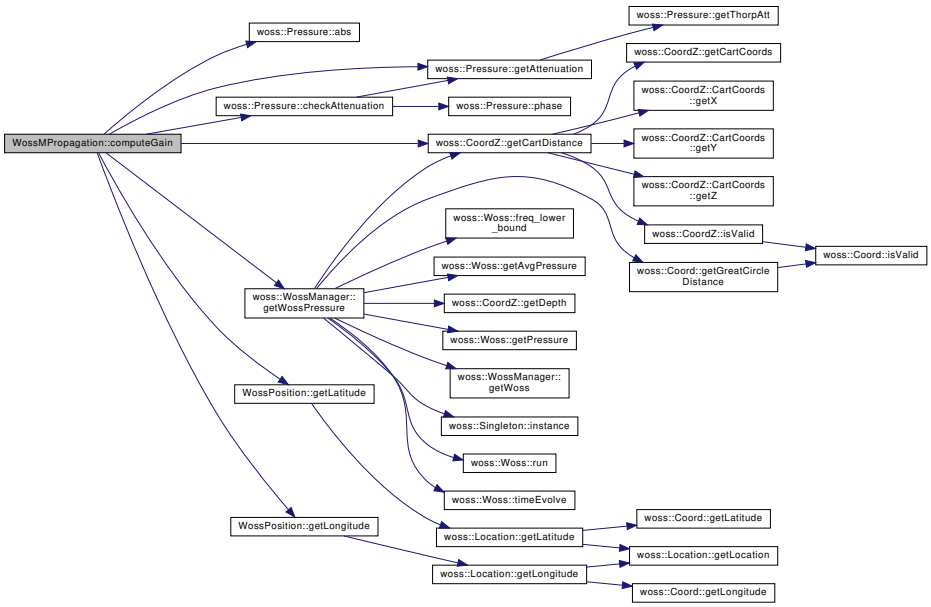
Returns

gain in uPa

References woss::Pressure::abs(), hdr_woss::already_processed, hdr_woss::attenuation, woss::Pressure::checkAttenuation(), hdr_woss::frequency, woss::Pressure::getAttenuation(), woss::CoordZ::getCartDistance(), WossPosition::getLatitude(), WossPosition::getLongitude(), and woss::WossManager::getWossPressure().

Referenced by getGain().

Here is the call graph for this function:



13.96.2.2 getGain() double WossMPropagation::getGain (Packet * p) [virtual]

Checks if a WossChannelModule has already processed current Packet; if not, it provides calculations

Parameters

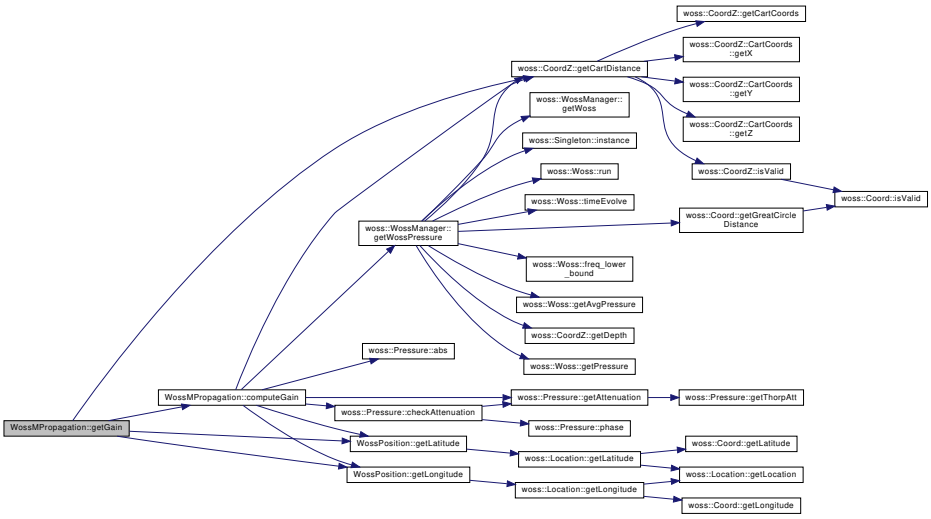
p	pointer to the current Packet being processed

Returns

gain in uPa

References hdr_woss::already_processed, hdr_woss::attenuation, computeGain(), woss::CoordZ::getCartDistance(), WossPosition::getLatitude(), and WossPosition::getLongitude().

Here is the call graph for this function:



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

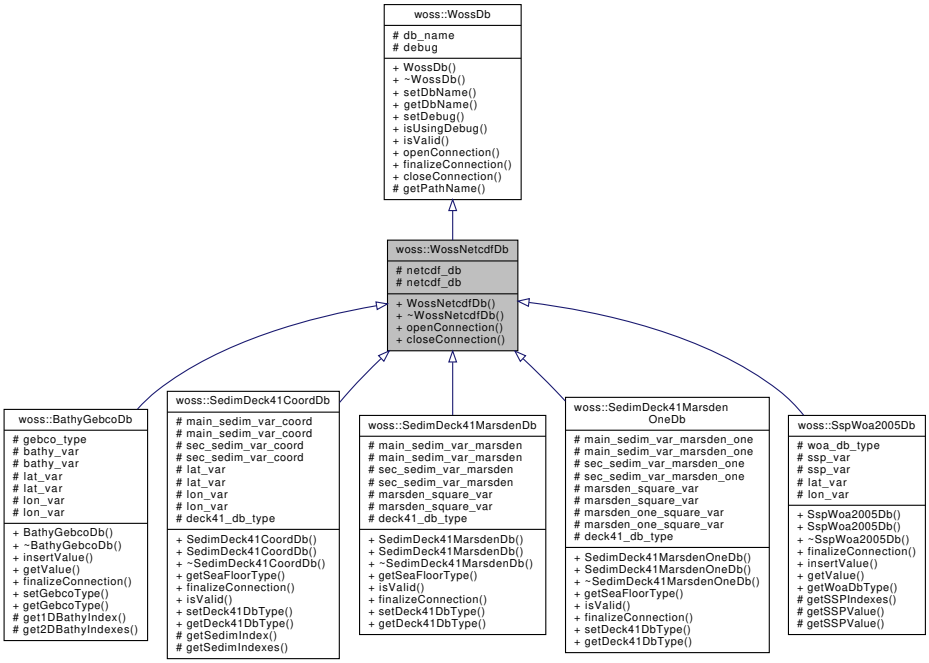
- woss_phy/uw-woss-mpropagation.h
- woss_phy/uw-woss-mpropagation.cpp

13.97 woss::WossNetcdfDb Class Reference

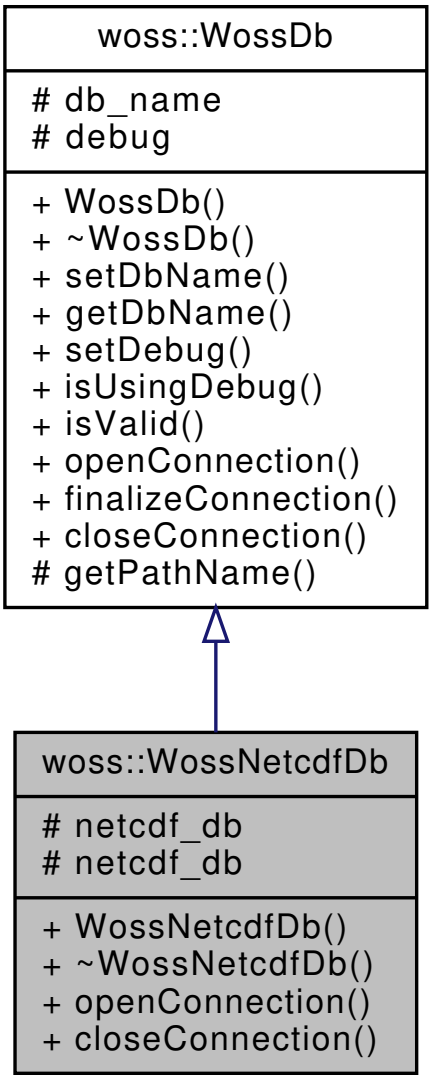
NetCDF implementation of WossDb.

`#include <woss-db.h>`

Inheritance diagram for woss::WossNetcdfDb:



Collaboration diagram for woss::WossNetcdfDb:



Public Member Functions

- WossNetcdfDb (const ::std::string &name)
- virtual bool openConnection ()
- virtual bool closeConnection ()

Protected Attributes

- netCDF::NcFile * netcdf_db
- NcFile * **netcdf_db**

Additional Inherited Members

13.97.1 Detailed Description

NetCDF implementation of WossDb.

WossNetcdfDb is the NetCDF specialization of WossDb class. It sets up connection to the file and properly initializes a NcFile* pointer. NetCDF variables however are not superimposed by this class. User has the task to correctly create and initialize them with the method finalizeConnection()

See also

BathyGebcoDb, SedimDeck41CoordDb, SedimDeck41MarsdenDb, SedimDeck41MarsdenOneDb, SspWoa2005Db

13.97.2 Constructor & Destructor Documentation

13.97.2.1 WossNetcdfDb() `WossNetcdfDb::WossNetcdfDb ( const ::std::string & name )`

WossNetcdfDb constructor

Parameters

name	pathname of database

13.97.3 Member Function Documentation

13.97.3.1 closeConnection() `bool WossNetcdfDb::closeConnection ( ) [virtual]`

Closes the connection to the NetCDF pathname provided

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References netcdf_db.

Referenced by woss::SedimDeck41Db::closeConnection().

13.97.3.2 openConnection() `bool WossNetcdfDb::openConnection ( ) [virtual]`

Opens the connection to the NetCDF pathname provided

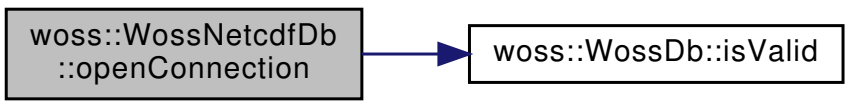
Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References woss::WossDb::db_name, woss::WossDb::isValid(), and netcdf_db.

Here is the call graph for this function:



13.97.4 Member Data Documentation

13.97.4.1 netcdf_db `netCDF::NcFile* woss::WossNetcdfDb::netcdf_db [protected]`

NcFile pointer to a NetCDF database descriptor. It will be properly initialized by openConnection()

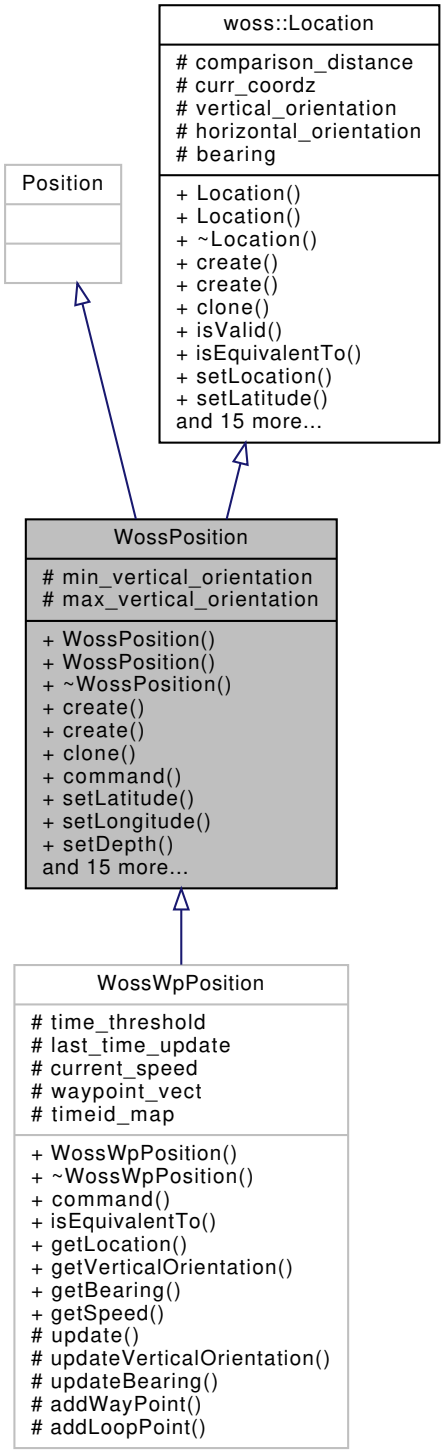
Referenced by closeConnection(), woss::BathyGebcoDb::finalizeConnection(), woss::SedimDeck41CoordDb::finalizeConnection(), woss::SedimDeck41MarsdenDb::finalizeConnection(), woss::SedimDeck41MarsdenOneDb::finalizeConnection(), woss::SspWoa2005Db::finalizeConnection(), and openConnection().

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

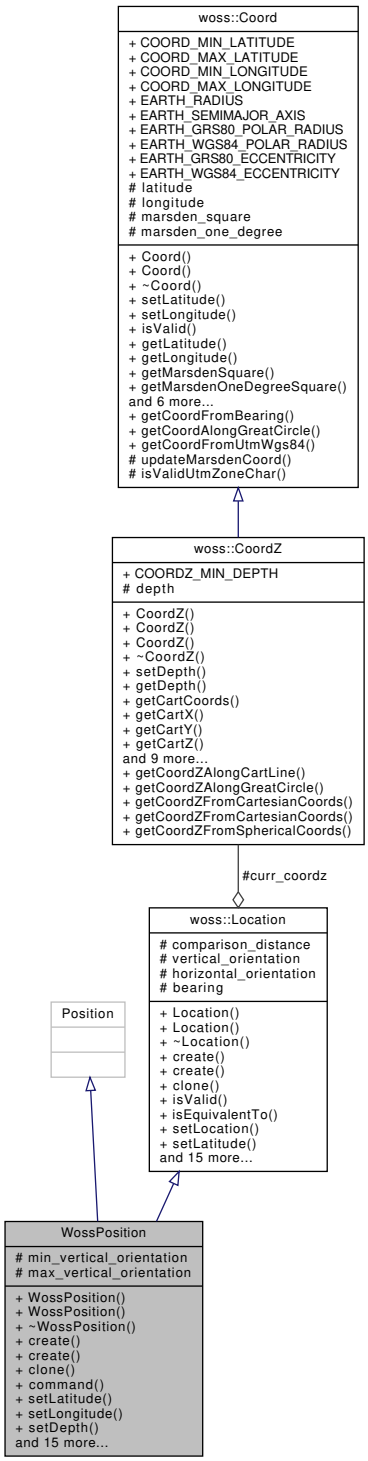
- woss/woss_db/woss-db.h
- woss/woss_db/woss-db.cpp

13.98 WossPosition Class Reference

Inheritance diagram for WossPosition:



Collaboration diagram for WossPosition:



Public Member Functions

- **WossPosition** (double latitude, double longitude, double depth=0, double dist=LOCATION_↩ COMPARISON_DISTANCE)
- **WossPosition** (const woss::CoordZ &coordz=woss::CoordZ(), double dist=LOCATION_COMPARISON_↩ DISTANCE)
- virtual WossPosition * create (double latitude, double longitude, double depth=0, double dist=LOCATION_↩ COMPARISON_DISTANCE) const

- virtual WossPosition * create (const woss::CoordZ &coordz=woss::CoordZ(), double dist=LOCATION_↩ COMPARISON_DISTANCE) const
- virtual WossPosition * clone () const
- virtual int **command** (int argc, const char *const *argv)
- virtual void setLatitude (double val)
- virtual void setLongitude (double val)
- virtual void setDepth (double val)
- virtual void **setAltitude** (double val)
- virtual void **setMinVerticalOrientation** (double val)
- virtual void **setMaxVerticalOrientation** (double val)
- virtual void **setX** (double val)
- virtual void **setY** (double val)
- virtual void **setZ** (double val)
- virtual double getX ()
- virtual double getY ()
- virtual double getZ ()
- virtual double getLatitude ()
- virtual double getLongitude ()
- virtual double getDepth ()
- virtual double **getAltitude** ()
- virtual double **getMinVerticalOrientation** ()
- virtual double **getMaxVerticalOrientation** ()

Protected Attributes

- double **min_vertical_orientation**
- double **max_vertical_orientation**

13.98.1 Member Function Documentation

13.98.1.1 clone() `virtual WossPosition * WossPosition::clone ( ) const [inline], [virtual]`

Location virtual factory method

Returns

a heap-allocated copy of **this** instance

Reimplemented from woss::Location.

13.98.1.2 create() `[1/2]`

```
virtual WossPosition * WossPosition::create (
    const woss::CoordZ & coordz = woss::CoordZ(),
    double dist = LOCATION_COMPARISON_DISTANCE ) const [inline], [virtual]
```

Location virtual factory method

Parameters

coordz	coordinates
dist	distance comparison precision [m]

Returns

a heap-allocated Location object

Reimplemented from woss::Location.

13.98.1.3 create() [2/2]

```
virtual WossPosition * WossPosition::create (
        double latitude,
        double longitude,
        double depth = 0,
        double dist = LOCATION_COMPARISON_DISTANCE ) const  [inline], [virtual]
```

Location virtual factory method

Parameters

latitude	latitude value [decimal degrees]
longitude	longitude value [decimal degrees]
depth	depth value [m]
dist	distance comparison precision [m]

Returns

a heap-allocated Location object

Reimplemented from woss::Location.

13.98.1.4 getDepth()

```
double WossPosition::getDepth ( )  [virtual]
```

Gets current depth

Returns

valid depth [m]

Reimplemented from woss::Location.

References woss::Location::getDepth().

Here is the call graph for this function:

13.98.1.5 getLatitude() `double WossPosition::getLatitude ( ) [virtual]`

Gets current latitude

Returns

valid latitude [decimal degrees]

Reimplemented from woss::Location.

References woss::Location::getLatitude().

Referenced by WossMPropagation::computeGain(), and WossMPropagation::getGain().

Here is the call graph for this function:

13.98.1.6 getLongitude() `double WossPosition::getLongitude ( ) [virtual]`

Gets current longitude

Returns

valid longitude [decimal degrees]

Reimplemented from woss::Location.

References woss::Location::getLongitude().

Referenced by WossMPropagation::computeGain(), and WossMPropagation::getGain().

Here is the call graph for this function:

13.98.1.7 getX() `double WossPosition::getX ( ) [virtual]`

Gets current cartesian x-axis value

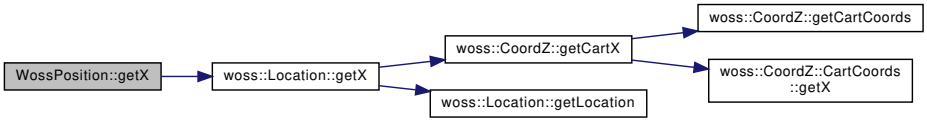
Returns

x value

Reimplemented from woss::Location.

References woss::Location::getX().

Here is the call graph for this function:



13.98.1.8 getY() `double WossPosition::getY ( ) [virtual]`

Gets current cartesian y-axis value

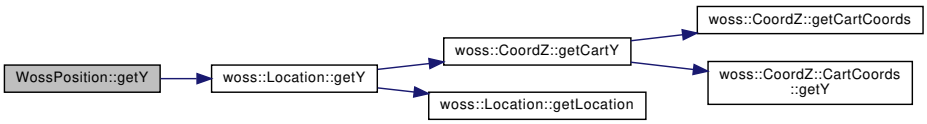
Returns

y value

Reimplemented from woss::Location.

References woss::Location::getY().

Here is the call graph for this function:



13.98.1.9 getZ() `double WossPosition::getZ ( ) [virtual]`

Gets current cartesian z-axis value

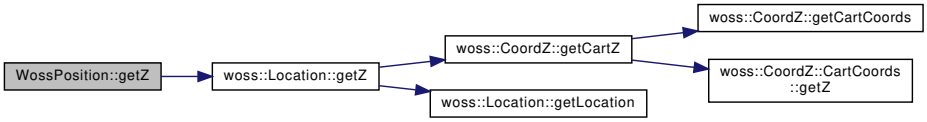
Returns

z value

Reimplemented from woss::Location.

References woss::Location::getZ().

Here is the call graph for this function:



13.98.1.10 setDepth() `void WossPosition::setDepth ( double depth ) [virtual]`

Sets initial depth

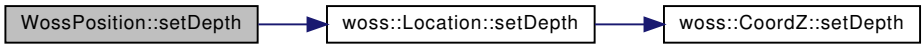
Parameters

lat	valid depth [m]

Reimplemented from woss::Location.

References woss::Location::setDepth().

Here is the call graph for this function:



13.98.1.11 setLatitude() `void WossPosition::setLatitude ( double lat ) [virtual]`

Sets initial latitude

Parameters

lat	valid latitude [decimal degrees]

Reimplemented from woss::Location.

References woss::Location::setLatitude().

Here is the call graph for this function:

13.98.1.12 setLongitude() `void WossPosition::setLongitude ( double lon ) [virtual]`

Sets initial longitude

Parameters

lat	valid longitude [decimal degrees]

Reimplemented from woss::Location.

References woss::Location::setLongitude().

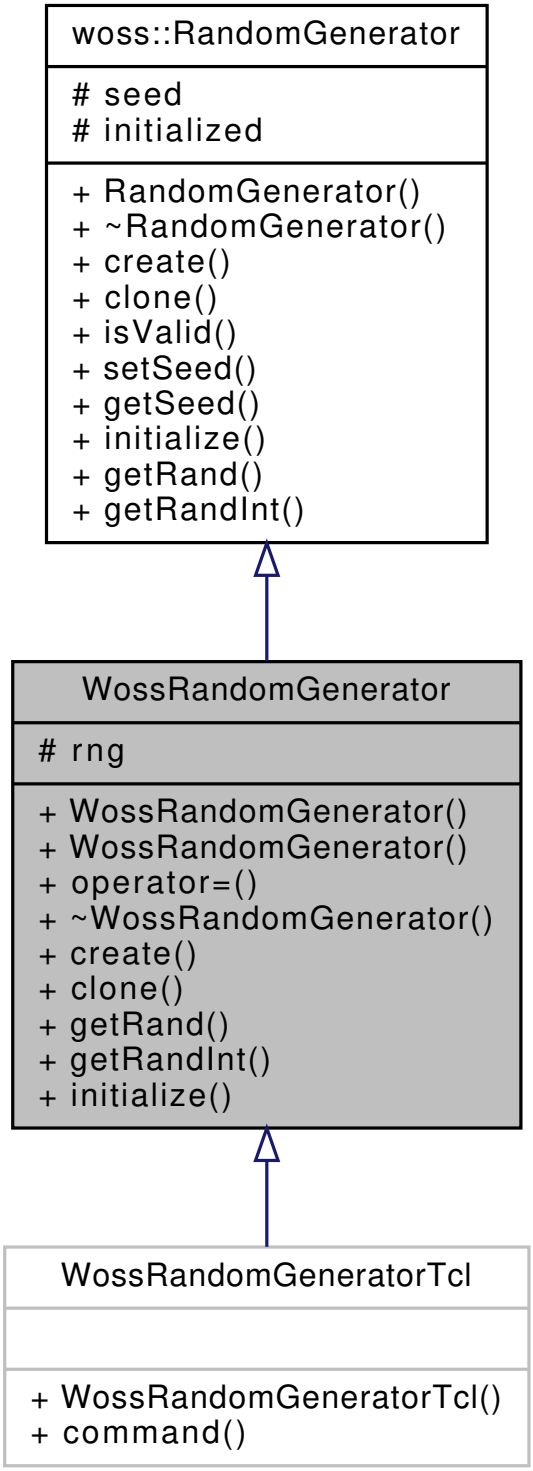
Here is the call graph for this function:

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

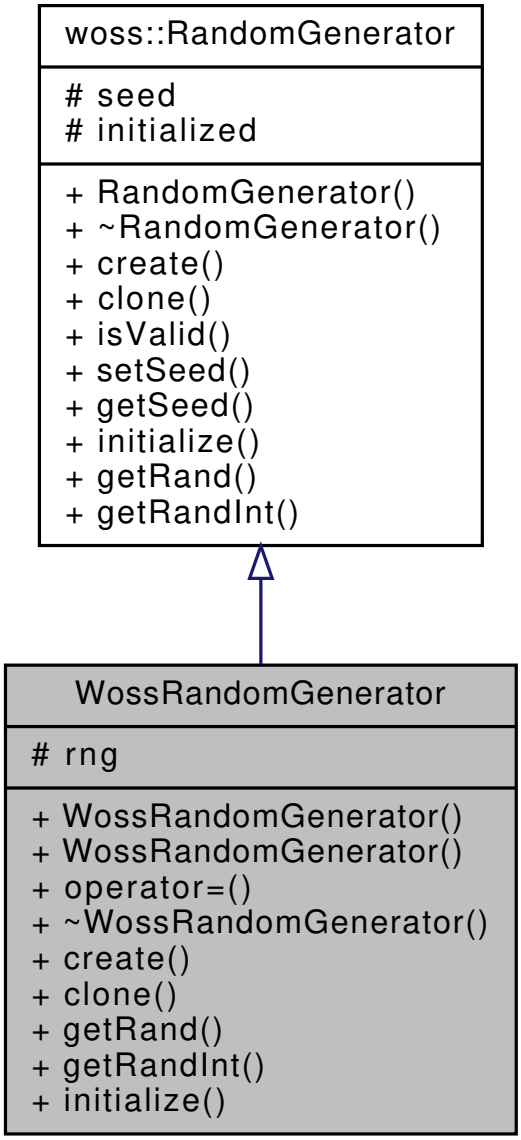
- woss_phy/uw-woss-position.h
- woss_phy/uw-woss-position.cpp

13.99 WossRandomGenerator Class Reference

Inheritance diagram for WossRandomGenerator:



Collaboration diagram for WossRandomGenerator:



Public Member Functions

- **WossRandomGenerator** (int seed=0)
- **WossRandomGenerator** (const WossRandomGenerator &copy)
- WossRandomGenerator & **operator=** (const WossRandomGenerator &copy)
- virtual WossRandomGenerator * **create** (int s)
- virtual WossRandomGenerator * clone () const
- virtual double getRand () const
- virtual int getRandInt () const
- virtual void initialize ()

Protected Attributes

- RNG * **rng**

13.99.1 Member Function Documentation

13.99.1.1 clone() `virtual WossRandomGenerator * WossRandomGenerator::clone ( ) const [inline], [virtual]`

RandomGenerator virtual factory method

Returns

a heap-allocated RandomGenerator copy of **this** instance

Reimplemented from woss::RandomGenerator.

13.99.1.2 getRand() `double WossRandomGenerator::getRand ( ) const [virtual]`

Gets a random value

Returns

a random value between 0 and 1

Reimplemented from woss::RandomGenerator.

References woss::RandomGenerator::initialized.

13.99.1.3 getRandInt() `int WossRandomGenerator::getRandInt ( ) const [virtual]`

Gets a random integer value

Returns

a random integer

Reimplemented from woss::RandomGenerator.

References woss::RandomGenerator::initialized.

13.99.1.4 initialize() `void WossRandomGenerator::initialize ( ) [virtual]`

Mandatory function to initialize the instance

Reimplemented from woss::RandomGenerator.

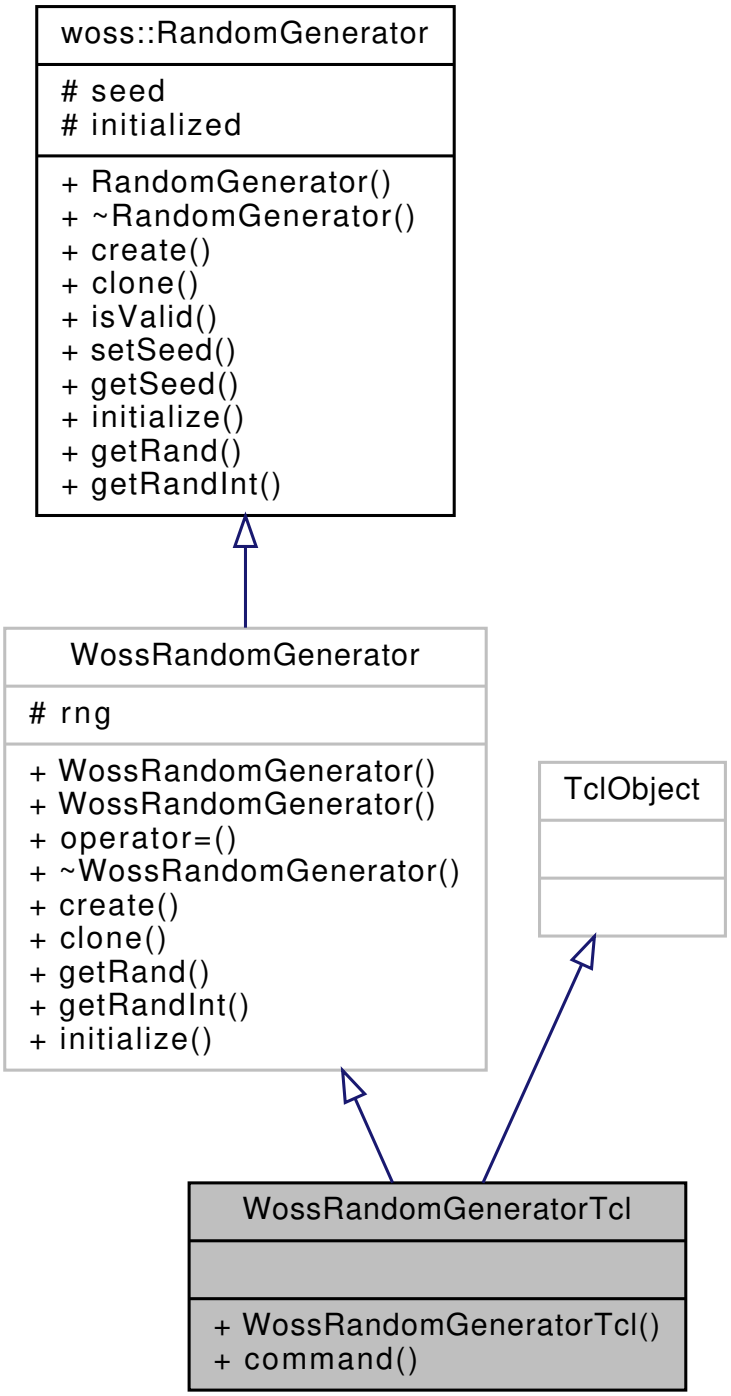
References woss::RandomGenerator::initialized, and woss::RandomGenerator::seed.

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

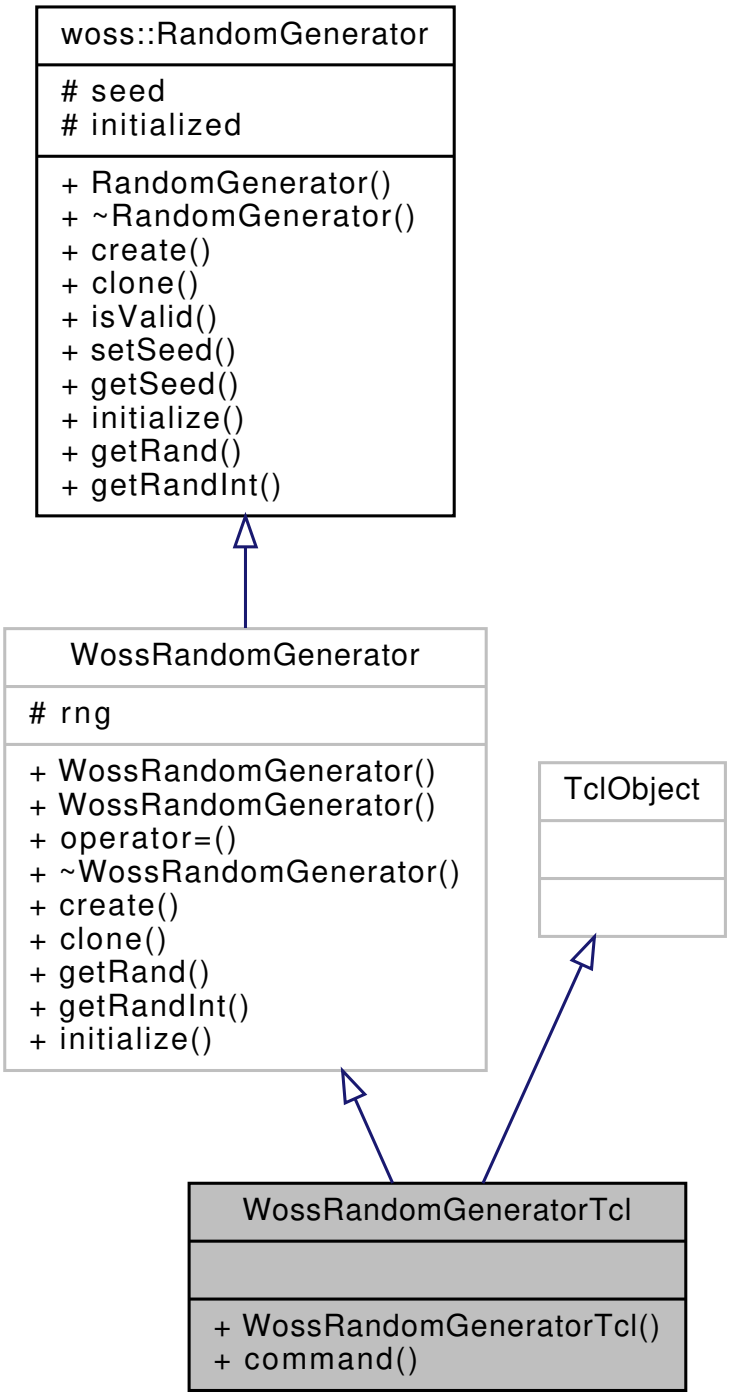
- woss_phy/uw-woss-random-generator.h
- woss_phy/uw-woss-random-generator.cpp

13.100 WossRandomGeneratorTcl Class Reference

Inheritance diagram for WossRandomGeneratorTcl:



Collaboration diagram for WossRandomGeneratorTcl:



Public Member Functions

- virtual int **command** (int argc, const char ∗const ∗argv)

Additional Inherited Members

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

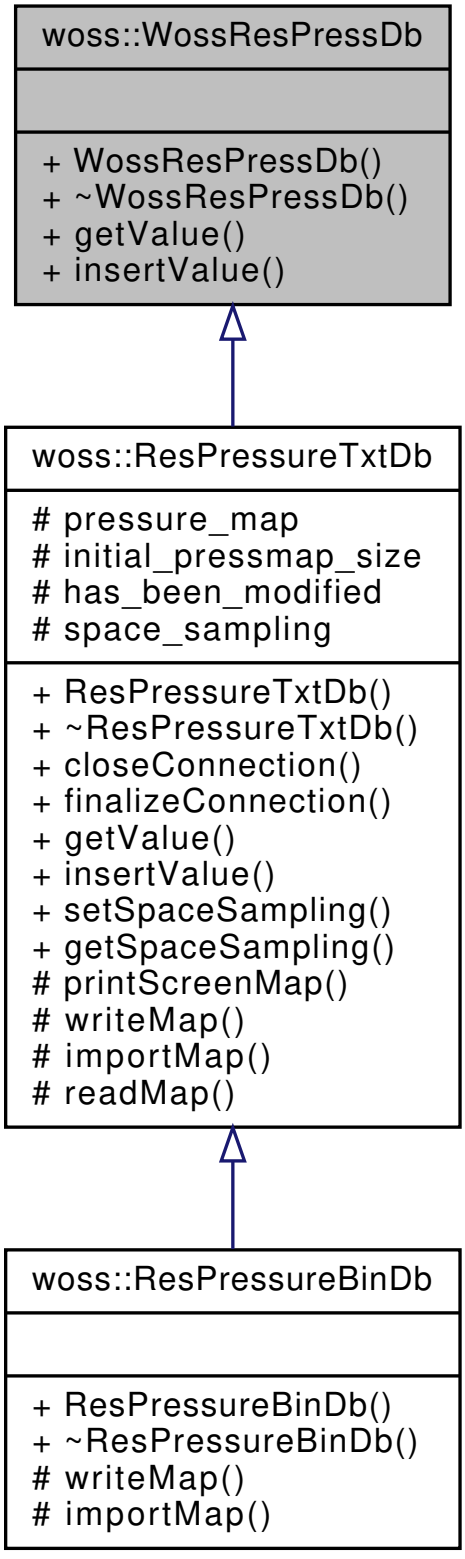
- woss_phy/uw-woss-random-generator.h
- woss_phy/uw-woss-random-generator.cpp

13.101 woss::WossResPressDb Class Reference

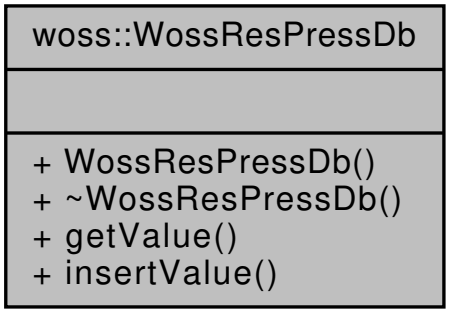
Data behaviour class for storing calculated Pressure.

```
#include <woss-db.h>
```

Inheritance diagram for woss::WossResPressDb:



Collaboration diagram for woss::WossResPressDb:



Public Member Functions

- virtual Pressure * getValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const =0
- virtual bool insertValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const Pressure &pressure)=0

13.101.1 Detailed Description

Data behaviour class for storing calculated Pressure.

WossResPressDb is the prototype of any class that implements a database database for calculated Pressure

See also

ResPressureTxtDb

13.101.2 Member Function Documentation

13.101.2.1 getValue()

```
virtual Pressure * woss::WossResPressDb::getValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [pure virtual]
```

Returns a heap-created Pressure value for given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]

Returns

valid TimeArr if parameters are found, *not valid* otherwise

Implemented in woss::ResPressureTxtDb.

Referenced by woss::WossDbManager::getPressure().

13.101.2.2 insertValue()

```
virtual bool woss::WossResPressDb::insertValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value,
    const Pressure & pressure )  [pure virtual]
```

Inserts the given Pressure value in the database at given frequency, transmitter and receiver coordinates

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
pressure	computed Pressure

Returns

true if method was successful, *false* otherwise

Implemented in woss::ResPressureTxtDb.

Referenced by woss::WossDbManager::insertPressure().

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

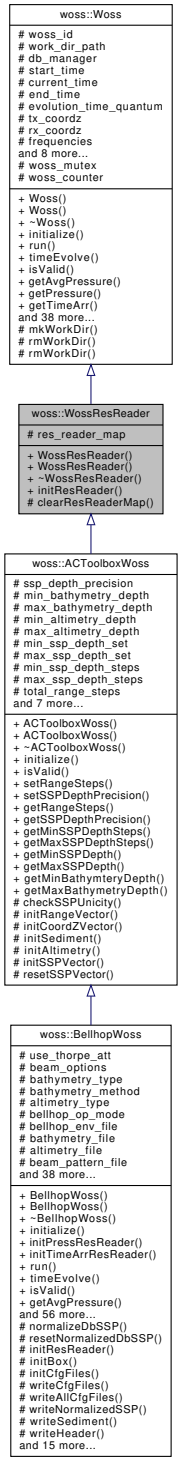
- woss/woss_db/woss-db.h

13.102 woss::WossResReader Class Reference

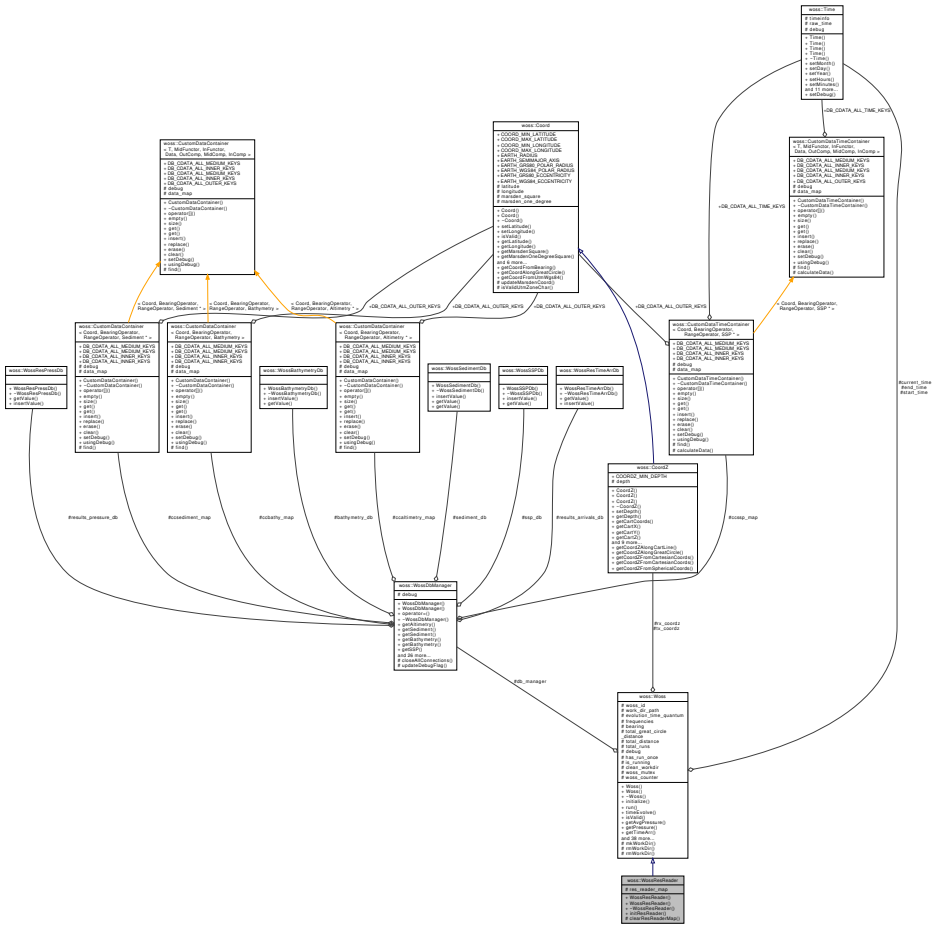
Woss class with ResReader objects for reading simulated results.

```
#include <woss.h>
```

Inheritance diagram for woss::WossResReader:



Collaboration diagram for woss::WossResReader:



Public Member Functions

- WossResReader ()
- WossResReader (const CoordZ &tx, const CoordZ &rx, const Time &start_t, const Time &end_t, double start_freq, double end_freq, double freq_step)
- virtual bool initResReader (double curr_frequency)=0

Protected Member Functions

- void clearResReaderMap ()

Protected Attributes

- ResReaderMap res_reader_map

Additional Inherited Members

13.102.1 Detailed Description

Woss class with ResReader objects for reading simulated results.

WossResReader implements Woss with the add of a ResReaderMap object for reading and further refination purposes

13.102.2 Constructor & Destructor Documentation

13.102.2.1 WossResReader() [1/2] woss::WossResReader::WossResReader ( ) [inline]

Default constructor

13.102.2.2 WossResReader() [2/2] woss::WossResReader::WossResReader (

```
const CoordZ & tx,
const CoordZ & rx,
const Time & start_t,
const Time & end_t,
double start_freq,
double end_freq,
double freq_step ) [inline]
```

Woss constructor

Parameters

tx	const reference to a valid CoordZ object (transmitter)
rx	const reference to a valid CoordZ object (receiver)
start_t	const reference to a valid Time object for SSP's averaging purposes (start date time)
end_t	const reference to a valid Time object for SSP's averaging purposes (end date time)
start_freq	start frequency [Hz]
end_freq	end frequency [Hz]
freq_step	frequency step [Hz]

13.102.3 Member Function Documentation

13.102.3.1 clearResReaderMap() void WossResReader::clearResReaderMap ( ) [protected]

Clears res_reader_map

References res_reader_map.

13.102.3.2 initResReader() virtual bool woss::WossResReader::initResReader (

```
double curr_frequency ) [pure virtual]
```

Initializes current ResReader object

Parameters

curr_frequency	frequency in use [Hz]

Returns

true if method succeeded, *false* otherwise

Implemented in woss::BellhopWoss.

13.102.4 Member Data Documentation

13.102.4.1 res_reader_map `ResReaderMap woss::WossResReader::res_reader_map [protected]`

ResReader map

Referenced by clearResReaderMap(), woss::BellhopWoss::getAvgPressure(), woss::BellhopWoss::getPressure(), woss::BellhopWoss::getTimeArr(), woss::BellhopWoss::initPressResReader(), woss::BellhopWoss::initResReader(), and woss::BellhopWoss::initTimeArrResReader().

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

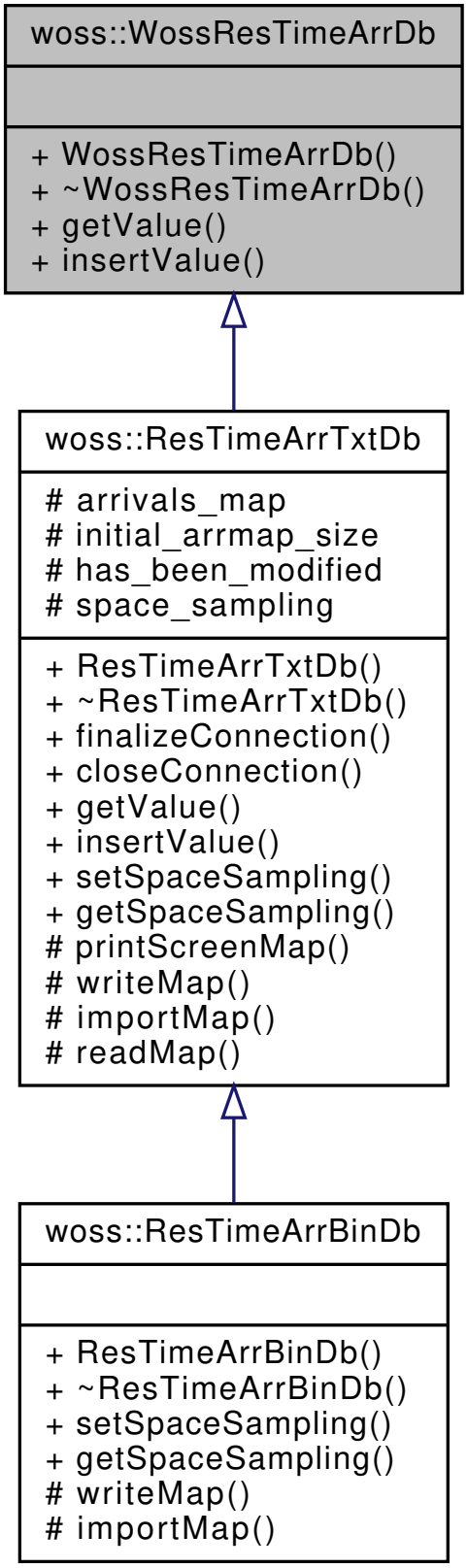
- woss/woss.h
- woss/woss.cpp

13.103 woss::WossResTimeArrDb Class Reference

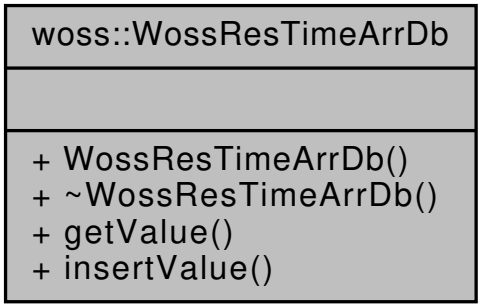
Data behaviour class for storing calculated TimeArr.

`#include <woss-db.h>`

Inheritance diagram for woss::WossResTimeArrDb:



Collaboration diagram for woss::WossResTimeArrDb:



Public Member Functions

- virtual TimeArr * getValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value) const =0
- virtual bool insertValue (const CoordZ &coord_tx, const CoordZ &coord_rx, const double frequency, const Time &time_value, const TimeArr &channel)=0

13.103.1 Detailed Description

Data behaviour class for storing calculated TimeArr.

WossResTimeArrDb is the prototype of any class that implements a database database for calculated TimeArr

See also

ResTimeArrTxtDb

13.103.2 Member Function Documentation

13.103.2.1 getValue()

```
virtual TimeArr * woss::WossResTimeArrDb::getValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value ) const  [pure virtual]
```

Returns a heap-created TimeArr value for given frequency, transmitter and receiver coordinates if present in the database. **User is responsible of pointer's ownership**

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]

Returns

valid TimeArr if parameters are found, *not valid* otherwise

Implemented in woss::ResTimeArrTxtDb.

Referenced by woss::WossDbManager::getTimeArr().

13.103.2.2 insertValue()

```
virtual bool woss::WossResTimeArrDb::insertValue (
    const CoordZ & coord_tx,
    const CoordZ & coord_rx,
    const double frequency,
    const Time & time_value,
    const TimeArr & channel )  [pure virtual]
```

Inserts the given TimeArr value in the database at given frequency, transmitter and receiver coordinates

Parameters

coord_tx	const reference to a valid CoordZ object
coord_rx	const reference to a valid CoordZ object
frequency	used frequency [hz]
channel	computed TimeArr

Returns

true if method was successful, *false* otherwise

Implemented in woss::ResTimeArrTxtDb.

Referenced by woss::WossDbManager::insertTimeArr().

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

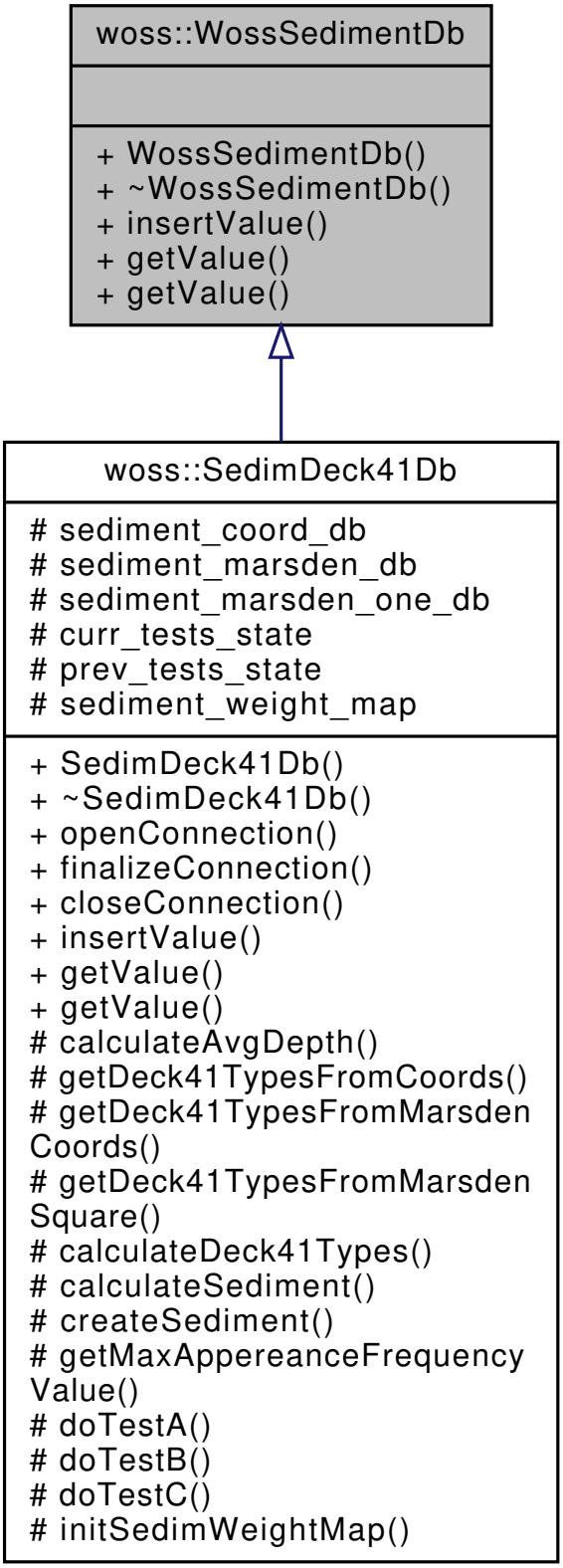
- woss/woss_db/woss-db.h

13.104 woss::WossSedimentDb Class Reference

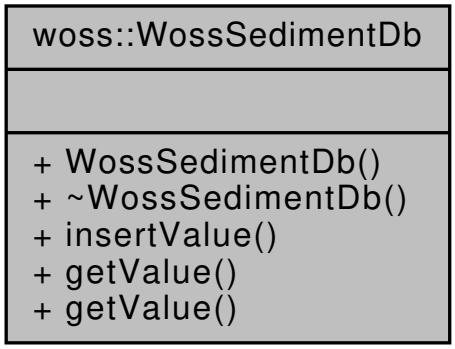
Data behaviour class for Sediment database.

```
#include <woss-db.h>
```

Inheritance diagram for woss::WossSedimentDb:



Collaboration diagram for woss::WossSedimentDb:



Public Member Functions

- virtual bool insertValue (const Coord &coordinates, const Sediment &sediment_value)=0
- virtual Sediment * getValue (const CoordZ &coords) const =0
- virtual Sediment * getValue (const CoordZVector &coordz_vector) const =0

13.104.1 Detailed Description

Data behaviour class for Sediment database.

WossSedimentDb is the prototype of any Sediment dabase in WOSS

See also

SedimDeck41Db

13.104.2 Member Function Documentation

13.104.2.1 getValue() [1/2]

```
virtual Sediment * woss::WossSedimentDb::getValue (
    const CoordZ & coords ) const  [pure virtual]
```

Returns a pointer to a heap-created Sediment value for given coordinates and depth, if present in the database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid CoordZ object

Returns

valid Sediment if coordinates are found, *not valid* otherwise

Implemented in woss::SedimDeck41Db.

Referenced by woss::WossDbManager::getSediment().

13.104.2.2 getValue() [2/2] virtual Sediment * woss::WossSedimentDb::getValue (const CoordZVector & *coordz_vector*) const [pure virtual]

Returns a pointer to a heap-created Sediment value for given coordinates and depth vector, if at least one set of coordinates is present in the database. **User is responsible of pointer's ownership**

Parameters

coordz_vector	const reference to a valid CoordZ vector

Returns

valid Sediment if at least one set of coordinates is found, *not valid* otherwise

Implemented in woss::SedimDeck41Db.

13.104.2.3 insertValue() virtual bool woss::WossSedimentDb::insertValue (const Coord & *coordinates*, const Sediment & *sediment_value*) [pure virtual]

Inserts the given woss::Sediment value in the database for given coordinates

Parameters

coordinates	const reference to a valid woss::Coord object
bathymetry_value	const reference to woss::Sediment value to be inserted

Returns

true if method was successful, *false* otherwise

Implemented in woss::SedimDeck41Db.

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

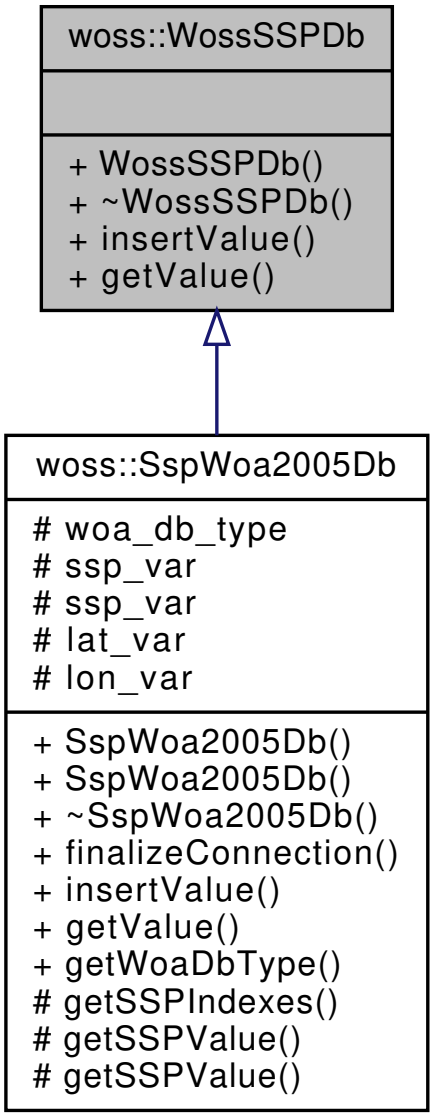
- woss/woss_db/woss-db.h

13.105 woss::WossSSPDb Class Reference

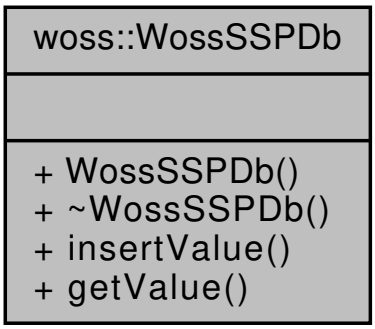
Data behaviour class for SSP database.

`#include <woss-db.h>`

Inheritance diagram for woss::WossSSPDb:



Collaboration diagram for woss::WossSSPDb:



Public Member Functions

- virtual bool insertValue (const Coord &coordinates, const Time &time_value, const SSP &ssp_value)=0
- virtual SSP ∗ getValue (const Coord &coords, const Time &time, long double ssp_depth_precision) const =0

13.105.1 Detailed Description

Data behaviour class for SSP database.

WossSSPDb is the prototype of any SSP dabase in WOSS

See also

SspWoa2005Db

13.105.2 Member Function Documentation

13.105.2.1 getValue()

```
virtual SSP * woss::WossSSPDb::getValue (
    const Coord & coords,
    const Time & time,
    long double ssp_depth_precision ) const  [pure virtual]
```

Returns a pointer to a heap-created SSP object for given coordinates and date time if both present in the database. **User is responsible of pointer's ownership**

Parameters

coords	const reference to a valid Coord object
time	const reference to a valid Time object
ssp_depth_precision	ssp depth precision [m]

Returns

valid SSP if coordinates and time date are found, *not valid* otherwise

Implemented in woss::SspWoa2005Db.

Referenced by woss::WossDbManager::getAverageSSP(), and woss::WossDbManager::getSSP().

13.105.2.2 insertValue()

```
virtual bool woss::WossSSPDb::insertValue (
    const Coord & coordinates,
    const Time & time_value,
    const SSP & ssp_value )  [pure virtual]
```

Inserts the given woss::SSP value in the database for given coordinates

Parameters

coordinates	const reference to a valid woss::Coord object
time_value	const reference to a valid woss::Time object
ssp_value	const reference to woss::SSP value to be inserted

Returns

true if method was successful, *false* otherwise

Implemented in woss::SspWoa2005Db.

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

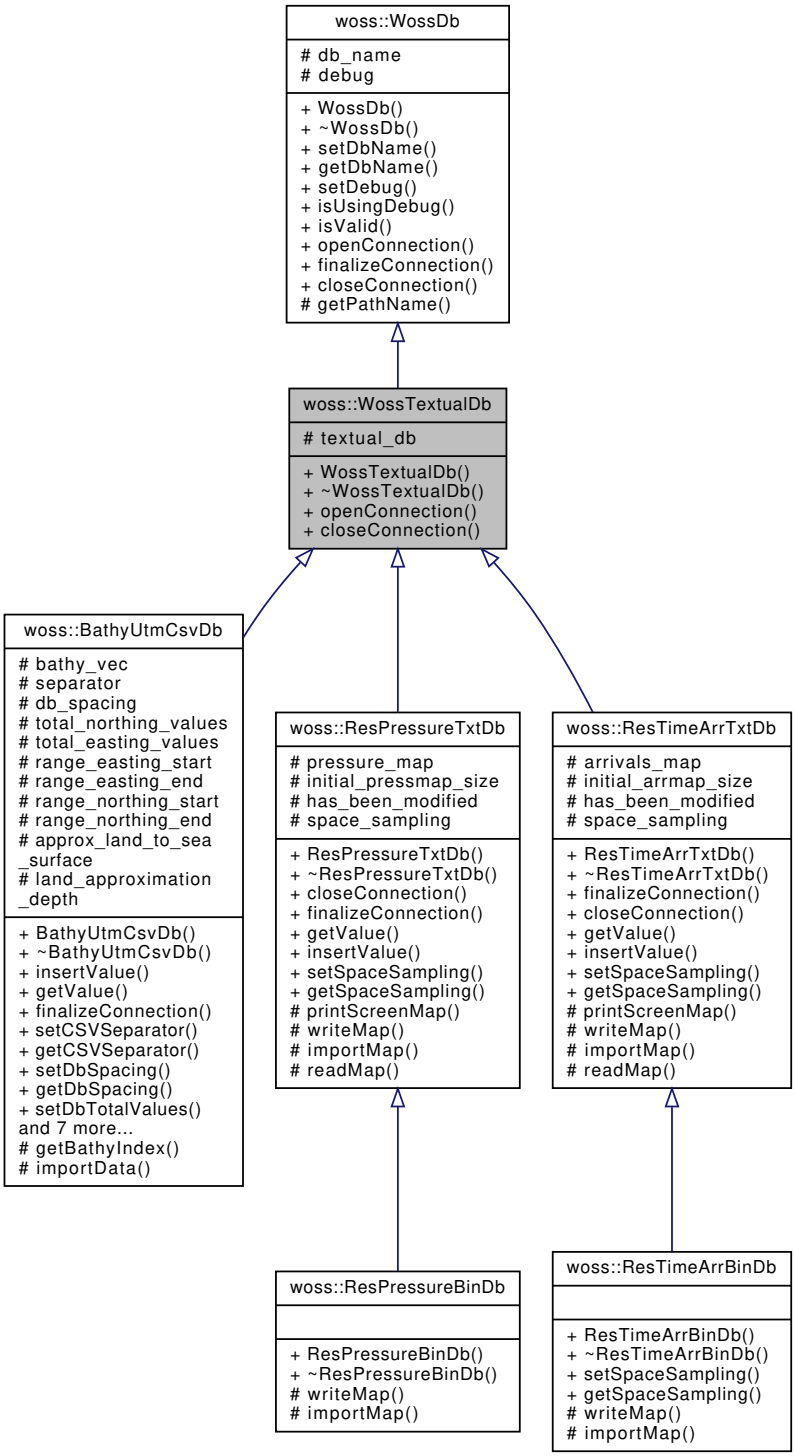
- woss/woss_db/woss-db.h

13.106 woss::WossTextualDb Class Reference

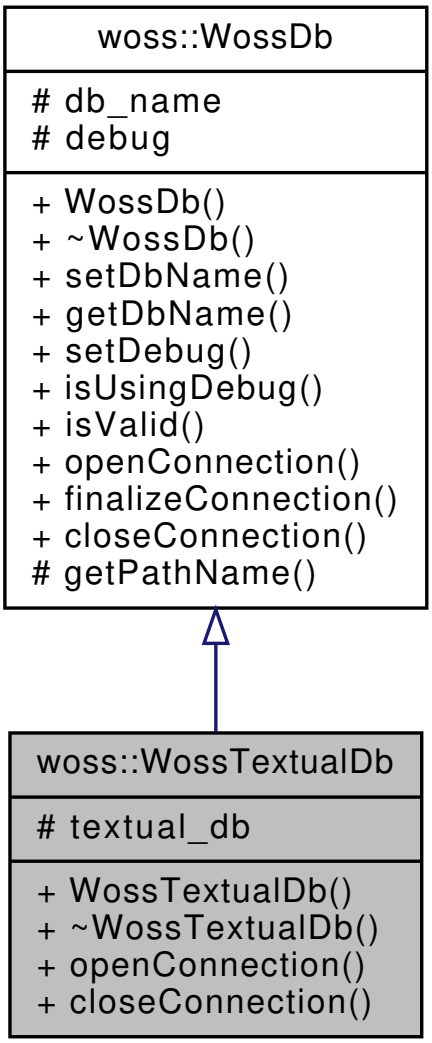
Textual implementation of WossDb.

`#include <woss-db.h>`

Inheritance diagram for woss::WossTextualDb:



Collaboration diagram for woss::WossTextualDb:



Public Member Functions

- WossTextualDb (const ::std::string &name)
- virtual bool openConnection ()
- virtual bool closeConnection ()

Protected Attributes

- ::std::fstream textual_db

Additional Inherited Members

13.106.1 Detailed Description

Textual implementation of WossDb.

WossNetcdfDb is the textual specialization of WossDb class. It sets up connection to the file and properly initializes a fstream object. No behaviour is superimposed by this class. User has the task to define method finalizeConnection()

See also

ResTimeArrTxtDb, ResPressureTxtDb

13.106.2 Constructor & Destructor Documentation

13.106.2.1 WossTextualDb() `WossTextualDb::WossTextualDb ( const ::std::string & name )`

WossTextualDb constructor

Parameters

name	pathname of database

13.106.3 Member Function Documentation

13.106.3.1 closeConnection() `bool WossTextualDb::closeConnection ( ) [virtual]`

Closes the connection to the textual file provided

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

Reimplemented in woss::ResPressureTxtDb, and woss::ResTimeArrTxtDb.

References textual_db.

Referenced by woss::ResPressureTxtDb::closeConnection(), and woss::ResTimeArrTxtDb::closeConnection().

13.106.3.2 openConnection() `bool WossTextualDb::openConnection ( ) [virtual]`

Opens the connection to the textual file provided

Returns

true if method was successful, *false* otherwise

Implements woss::WossDb.

References woss::WossDb::db_name.

13.106.4 Member Data Documentation

13.106.4.1 textual_db `::std::fstream woss::WossTextualDb::textual_db [protected]`

fstream object to a textual database. It will be properly initialized by openConnection()

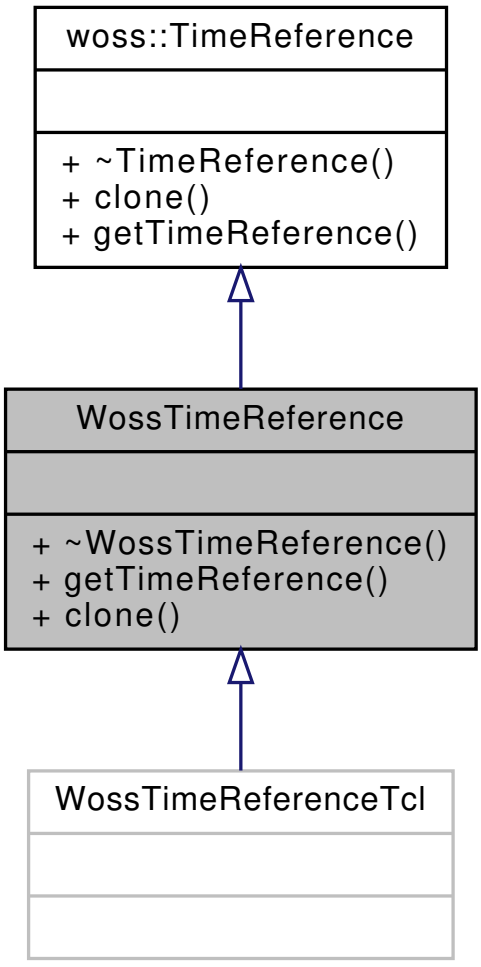
Referenced by closeConnection(), woss::BathyUtmCsvDb::importData(), woss::ResPressureBinDb::importMap(), woss::ResPressureTxtDb::importMap(), woss::ResTimeArrBinDb::importMap(), woss::ResTimeArrTxtDb::importMap(), woss::ResPressureBinDb::writeMap(), woss::ResPressureTxtDb::writeMap(), woss::ResTimeArrBinDb::writeMap(), and woss::ResTimeArrTxtDb::writeMap().

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

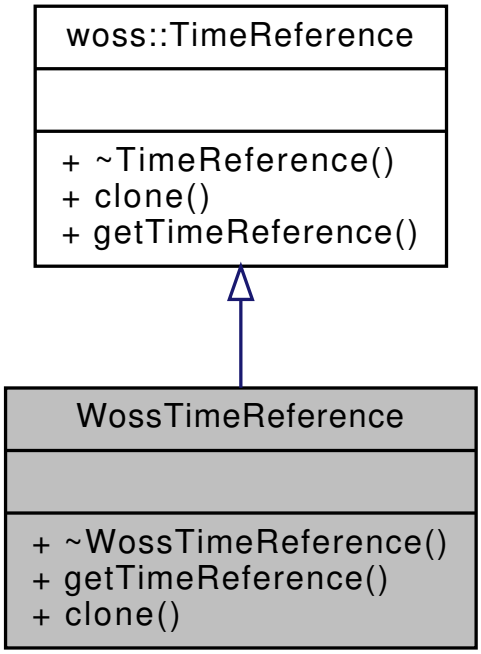
- woss/woss_db/woss-db.h
- woss/woss_db/woss-db.cpp

13.107 WossTimeReference Class Reference

Inheritance diagram for WossTimeReference:



Collaboration diagram for WossTimeReference:



Public Member Functions

- virtual double getTimeReference () const
- virtual WossTimeReference * clone ()

13.107.1 Member Function Documentation

13.107.1.1 clone() `virtual WossTimeReference * WossTimeReference::clone ( ) [inline], [virtual]`

woss::TimeReference virtual factory method

Returns

a heap-allocated copy of **this** instance

Implements woss::TimeReference.

13.107.1.2 getTimeReference() `virtual double WossTimeReference::getTimeReference ( ) const [inline], [virtual]`

Retuns simulation time

Returns

value in seconds

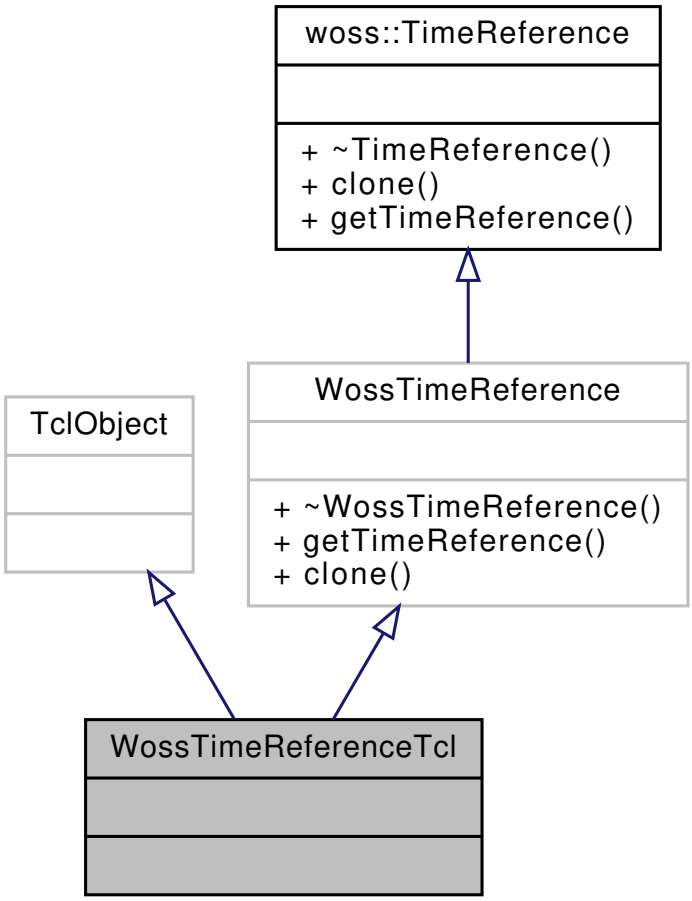
Implements woss::TimeReference.

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

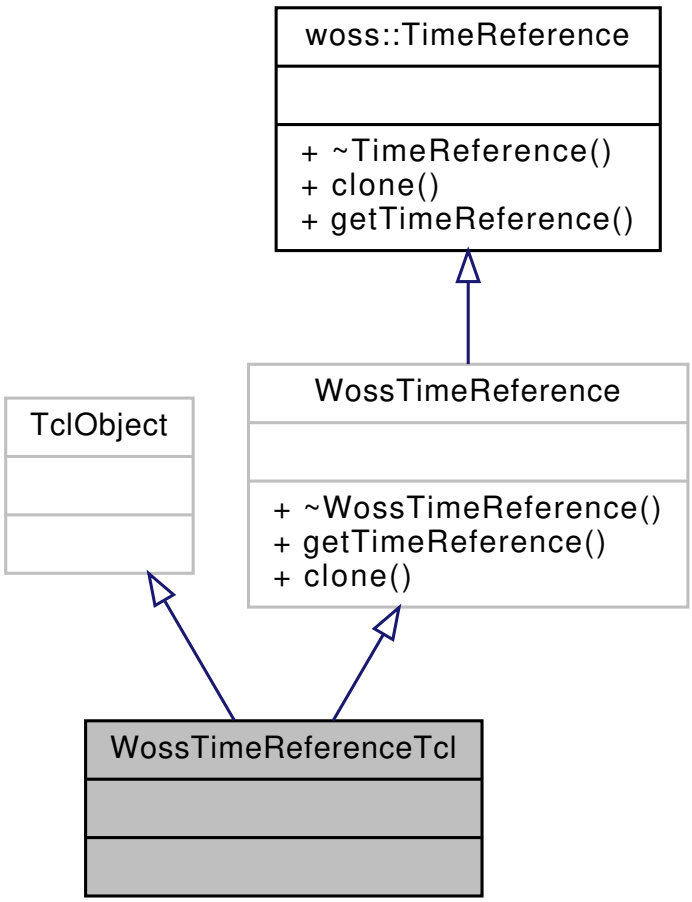
- woss_phy/uw-woss-time-reference.h

13.108 WossTimeReferenceTcl Class Reference

Inheritance diagram for WossTimeReferenceTcl:



Collaboration diagram for WossTimeReferenceTcl:



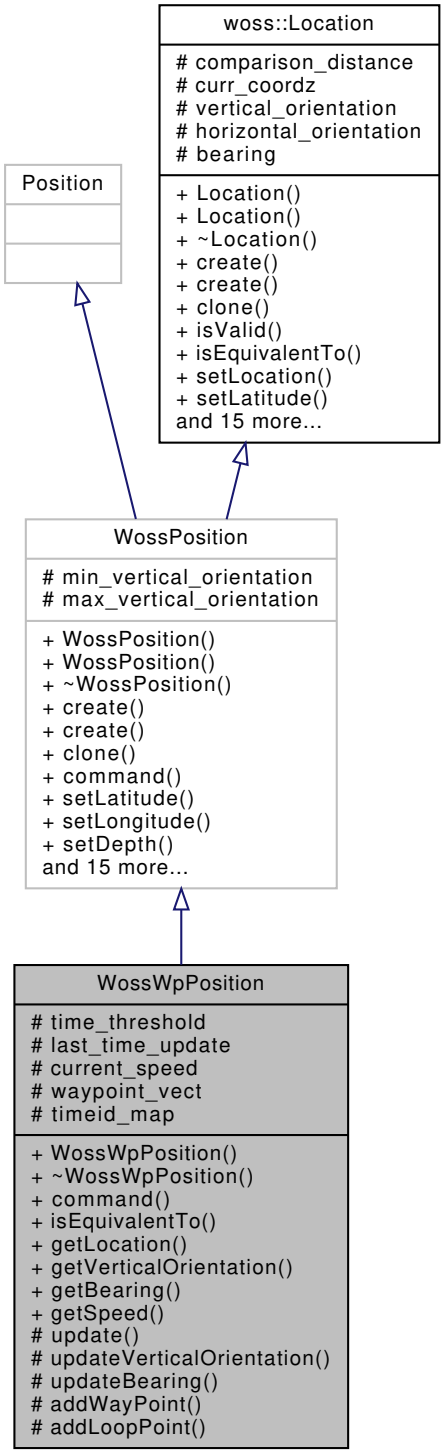
Additional Inherited Members

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

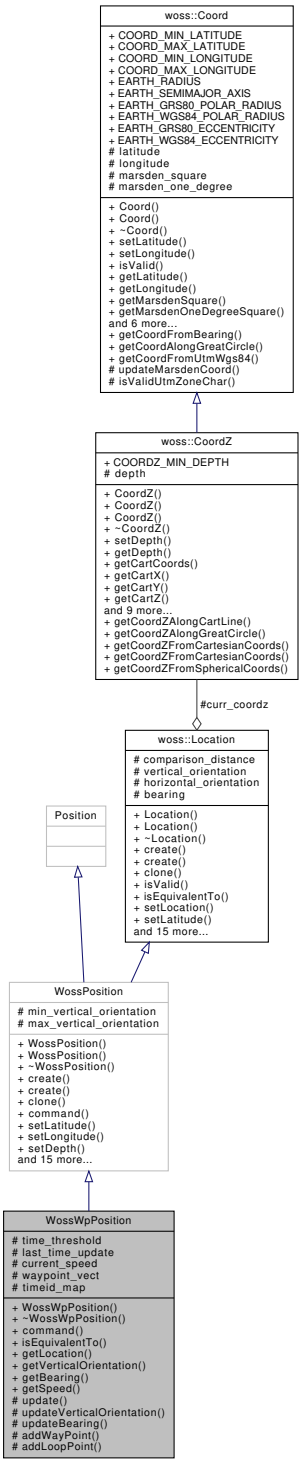
- woss_phy/uw-woss-time-reference.h

13.109 WossWpPosition Class Reference

Inheritance diagram for WossWpPosition:



Collaboration diagram for WossWpPosition:



Classes

- class WayPoint

Public Member Functions

- virtual int command (int argc, const char ∗const ∗argv)

- virtual bool isEquivalentTo (const woss::CoordZ &coordz)
- virtual woss::CoordZ getLocation ()
- virtual double getVerticalOrientation ()
- virtual double getBearing ()
- virtual double **getSpeed** ()

Protected Types

- typedef ::std::vector< WayPoint > **WayPointVect**
- typedef ::std::map< double, int > **TimeIdMap**
- typedef TimeIdMap::iterator **TIMIter**
- typedef TimeIdMap::reverse_iterator **TIMRIter**

Protected Member Functions

- virtual void **update** (double now)
- virtual void **updateVerticalOrientation** (const woss::CoordZ &prev, const woss::CoordZ &curr)
- virtual void **updateBearing** (const woss::CoordZ &prev, const woss::CoordZ &curr)
- virtual double **addWayPoint** (const WayPoint &waypoint)
- virtual double **addLoopPoint** (const WayPoint &waypoint)

Protected Attributes

- double **time_threshold**
- double **last_time_update**
- double **current_speed**
- WayPointVect **waypoint_vect**
- TimeIdMap **timeid_map**

13.109.1 Member Function Documentation

13.109.1.1 command()

```
int WossWpPosition::command (
        int argc,
        const char *const * argv )  [virtual]
```

Reimplemented from WossPosition.

13.109.1.2 getBearing()

```
double WossWpPosition::getBearing ( )  [virtual]
```

Gets current bearing in [-pi,pi]

Returns

bearing [dec degrees]

Reimplemented from woss::Location.

References woss::Location::bearing.

13.109.1.3 getLocation() `woss::CoordZ WossWpPosition::getLocation ( ) [virtual]`

Gets current coordinates

Returns

valid woss::CoordZ

Reimplemented from woss::Location.

References woss::Location::getLocation().

Here is the call graph for this function:

13.109.1.4 getVerticalOrientation() `double WossWpPosition::getVerticalOrientation ( ) [virtual]`

Gets current vertical orientation from reference line (0 degrees = parallel to sea surface / bottom). Negative values are towards the surface, while positive ones are towards sea bottom

Returns

difference angle [dec degrees]

Reimplemented from woss::Location.

References woss::Location::vertical_orientation.

13.109.1.5 isEquivalentTo() `bool WossWpPosition::isEquivalentTo ( const woss::CoordZ & coordz ) [virtual]`

Checks if the woss::CoordZ given is equivalent to this Location

Parameters

coordz	valid woss::CoordZ to check

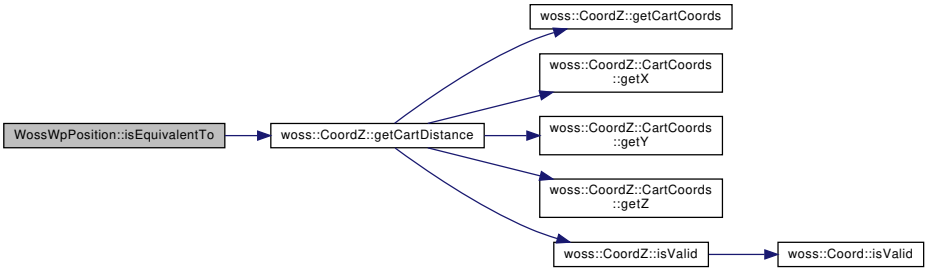
Returns

true if assumption is valid, *false* otherwise

Reimplemented from woss::Location.

References woss::Location::comparison_distance, woss::Location::curr_coordz, and woss::CoordZ::getCartDistance().

Here is the call graph for this function:



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

- woss_phy/uw-woss-waypoint-position.h
- woss_phy/uw-woss-waypoint-position.cpp

14 File Documentation

14.1 woss/ac-toolbox-arr-asc-reader.cpp File Reference

Provides the implementation of woss::ArrAscResReader and woss::ArrData classes.

14.1.1 Detailed Description

Provides the implementation of woss::ArrAscResReader and woss::ArrData classes.

Author

Federico Guerra

Provides the implementation of the woss::ArrAscResReader and woss::ArrData classes

14.2 woss/ac-toolbox-arr-asc-reader.h File Reference

Provides the interface for woss::ArrAscResReader and woss::ArrData classes.

Classes

- class woss::ArrData

  class for storing data of any acoustic toolbox ARR file
- class woss::ArrAscResReader

  Class for reading and manipulating results provided by any acoustic toolbox textual ARR file.

14.2.1 Detailed Description

Provides the interface for woss::ArrAscResReader and woss::ArrData classes.

Author

Federico Guerra

Provides the interface for the woss::ArrAscResReader and woss::ArrData classes

14.3 ac-toolbox-arr-asc-reader.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef AC_TOOLBOX_ARR_ASC_READER_DEFINITIONS_H
41 #define AC_TOOLBOX_ARR_ASC_READER_DEFINITIONS_H
42
43
44 #include <fstream>
45 #include <stdint.h>
46 #include <time-arrival-definitions.h>
47 #include "res-reader.h"
48
49
50 namespace woss {
51
52
58   class ArrData {
59
60
61     public:
62
63
64     ArrData();
65
66
70     ~ArrData() { delete[] tx_depths; delete[] rx_ranges; delete[] rx_depths; delete[] arr_values; }
71
72
76     float frequency;
77
78
82     int32_t Nsd;
83
87     float* tx_depths;
88
89
93     int32_t Nrd;
94
```

```
98     float* rx_depths;
99
100
104    int32_t Nrr;
105
109    float* rx_ranges;
110
111
115    TimeArr* arr_values;
116
117
121    void initialize() { tx_depths = NULL; rx_depths = NULL; rx_ranges = NULL; arr_values = NULL;
122                        Nrr = 0; Nrd = 0; Nsd = 0; frequency = 0.0; }
123
124
132    int getTimeArrIndex( double tx_depth, double rx_depth, double rx_range ) const;
133
141    int getIndex( float value, float* array, int array_size ) const;
142
143
144  };
145
146
153  class ArrAscResReader :  public ResReader {
154
155
156    public:
157
158
162    ArrAscResReader();
163
168    ArrAscResReader( const Woss* const woss );
169
170    virtual ~ArrAscResReader();
171
172
177    virtual bool initialize();
178
179
190    virtual Pressure* readAvgPressure( double tx_depth, double start_rx_depth, double start_rx_range,
       double end_rx_depth, double end_rx_range );
191
200    virtual Pressure* readPressure( double tx_depth, double rx_depth, double rx_range ) const;
201
202
210    virtual TimeArr* readTimeArr( double tx_depth, double rx_depth, double rx_range ) const;
211
212
213    protected:
214
215
219    bool arr_asc_header_collected;
220
224    bool arr_asc_file_collected;
225
226
230    ::std::ifstream file_reader;
231
235    ::std::streampos skip_header;
236
237
241    ArrData arr_file;
242
243    double last_tx_depth;
244
245    double last_start_rx_depth;
246
247    double last_start_rx_range;
248
249    double last_end_rx_depth;
250
251    double last_end_rx_range;
252
253    ::std::complex<double> last_ret_value;
254
262    TimeArr* accessMap( double tx_depth, double rx_depth, double rx_range )const {
263      return( arr_file.arr_values + arr_file.getTimeArrIndex( tx_depth, rx_depth, rx_range ) );
264    }
265
266
276    ::std::complex<double> readMapAvgPressure( double tx_depth, double start_rx_depth, double
       start_rx_range, double end_rx_depth, double end_rx_range );
277
278
283    bool getArrAscHeader();
284
289    bool getArrAscFile();
```

```
290
291
292   };
293
294
295 }
296
297
298 #endif /* AC_TOOLBOX_ARR_ASC_READER_DEFINITIONS_H */
299
300
301
302
```

14.4 woss/ac-toolbox-arr-bin-reader.cpp File Reference

Provides the implementation of woss::ArrBinResReader class.

14.4.1 Detailed Description

Provides the implementation of woss::ArrBinResReader class.

Author

Federico Guerra

Provides the implementation of the woss::ArrBinResReader class

14.5 woss/ac-toolbox-arr-bin-reader.h File Reference

Provides the interface for woss::ArrBinResReader class.

Classes

- class woss::ArrBinResReader
  Class for reading and manipulating results provided by any acoustic toolbox binary ARR file.

14.5.1 Detailed Description

Provides the interface for woss::ArrBinResReader class.

Author

Federico Guerra

Provides the interface for the woss::ArrBinResReader class

14.6 ac-toolbox-arr-bin-reader.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef AC_TOOLBOX_ARR_BIN_READER_DEFINITIONS_H
41 #define AC_TOOLBOX_ARR_BIN_READER_DEFINITIONS_H
42
43
44 #include "ac-toolbox-arr-asc-reader.h"
45
46
47 namespace woss {
48
49
56   class ArrBinResReader :  public ResReader {
57
58
59     public:
60
61
65     ArrBinResReader();
66
71     ArrBinResReader( const Woss* const woss );
72
73
74     virtual ~ArrBinResReader();
75
76
81     virtual bool initialize();
82
83
94     virtual Pressure* readAvgPressure( double tx_depth, double start_rx_depth, double start_rx_range,
      double end_rx_depth, double end_rx_range );
95
104     virtual Pressure* readPressure( double tx_depth, double rx_depth, double rx_range ) const;
105
106
114     virtual TimeArr* readTimeArr( double tx_depth, double rx_depth, double rx_range ) const;
115
116
117     protected:
118
119
123     bool arr_bin_header_collected;
124
128     bool arr_bin_file_collected;
129
130
134     ::std::ifstream file_reader;
135
139     ::std::streampos skip_header;
140
141
145     ArrData arr_file;
146
147     double last_tx_depth;
148
149     double last_start_rx_depth;
```

```
150
151     double last_start_rx_range;
152
153     double last_end_rx_depth;
154
155     double last_end_rx_range;
156
157     ::std::complex<double> last_ret_value;
158
159
167     TimeArr* accessMap(double tx_depth, double rx_depth, double rx_range)const {
168       return( arr_file.arr_values + arr_file.getTimeArrIndex( tx_depth, rx_depth, rx_range ) );
169     }
170
180     ::std::complex<double> readMapAvgPressure( double tx_depth, double start_rx_depth, double
      start_rx_range, double end_rx_depth, double end_rx_range );
181
182
187     bool getArrBinHeader();
188
193     bool getArrBinFile();
194
195
196   };
197
198
199 }
200
201
202 #endif /* AC_TOOLBOX_ARR_BIN_READER_DEFINITIONS_H */
203
204
205
206
207
```

14.7 woss/ac-toolbox-shd-reader.cpp File Reference

Provides the implementation of woss::ShdResReader and woss::ShdData classes.

14.7.1 Detailed Description

Provides the implementation of woss::ShdResReader and woss::ShdData classes.

Author

Federico Guerra

Provides the implementation of the woss::ShdResReader and woss::ShdData classes

14.8 woss/ac-toolbox-shd-reader.h File Reference

Provides the interface for woss::ShdResReader and woss::ShdData classes.

Classes

- class woss::ShdData

  class for storing data of any acoustic toolbox SHD file
- class woss::ShdResReader

  Class for reading and manipulating results provided by any acoustic toolbox SHD file.

14.8.1 Detailed Description

Provides the interface for woss::ShdResReader and woss::ShdData classes.

Author

Federico Guerra

Provides the interface for the woss::ShdResReader and woss::ShdData classes

14.9 ac-toolbox-shd-reader.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef AC_TOOLBOX_RES_READER_DEFINITIONS_H
41 #define AC_TOOLBOX_RES_READER_DEFINITIONS_H
42
43
44 #include <fstream>
45 #include <time-arrival-definitions.h>
46 #include "res-reader.h"
47
48
49 namespace woss {
50
51
57   class ShdData {
58
59
60     public:
61
62
63     ShdData();
64
68     ~ShdData() { delete[] theta; delete[] tx_depths; delete[] rx_ranges; delete[] rx_depths; delete[]
      press_values ; }
69
70
74     int32_t record_length;
75
76
80     char* plot_type;
81
82
86     float frequency;
87
88
92     int32_t Ntheta;
93
```

```
97     float* theta;
98
99
103    int32_t Nsd;
104
108    float* tx_depths;
109
110
114    int32_t Nrd;
115
119    float* rx_depths;
120
121
125    int32_t Nrr;
126
130    float* rx_ranges;
131
135    int32_t Nrx_per_range;
136
137
141    ::std::complex<double>* press_values;
142
143
147    void initialize() { plot_type = NULL; Ntheta = 0; theta = NULL; Nrx_per_range = 0; record_length =
       0; tx_depths = NULL;
148                        rx_depths = NULL; rx_ranges = NULL; Nrr = 0; Nrd = 0; Nsd = 0; press_values =
       NULL; frequency = 0.0; }
149
150
159    int getPressureIndex( double tx_depth, double rx_depth, double rx_range, double theta = 0.0 ) const;
160
168    int getIndex( float value, float* array, int32_t array_size ) const;
169
170
171  };
172
173
180  class ShdResReader :  public ResReader {
181
182
183    public:
184
185
189    ShdResReader();
190
195    ShdResReader( const Woss* const woss );
196
197    virtual ~ShdResReader();
198
203    virtual bool initialize();
204
214    virtual Pressure* readAvgPressure( double tx_depth, double start_rx_depth, double start_rx_range,
       double end_rx_depth, double end_rx_range );
215
223    virtual Pressure* readPressure( double tx_depth, double rx_depth, double rx_range ) const;
224
225
234    virtual TimeArr* readTimeArr( double tx_depth, double rx_depth, double rx_range ) const;
235
236
237    protected:
238
239
243    bool shd_header_collected;
244
248    bool shd_file_collected;
249
250
254    ::std::ifstream file_reader;
255
256
260    ShdData shd_file;
261
262    double last_tx_depth;
263
264    double last_start_rx_depth;
265
266    double last_start_rx_range;
267
268    double last_end_rx_depth;
269
270    double last_end_rx_range;
271
272    ::std::complex<double> last_ret_value;
273
274
285    ::std::complex<double> readMapAvgPressure( double tx_depth, double start_rx_depth, double
```

```
    start_rx_range, double end_rx_depth, double end_rx_range, double theta = 0.0 );
286
287
296     ::std::complex<double> accessMap( double tx_depth, double rx_depth, double rx_range, double theta =
    0.0 ) const;
297
298
303     bool getShdFile();
304
309     bool getShdHeader();
310
311
312   };
313
314
316   // inline functions
317
318   inline ::std::complex<double> ShdResReader::accessMap( double tx_depth, double rx_depth, double
    rx_range, double theta )const {
319     return( shd_file.press_values[ shd_file.getPressureIndex( tx_depth, rx_depth, rx_range, theta ) ] );
320   }
321
322
323 }
324
325
326 #endif /* AC_TOOLBOX_RES_READER_DEFINITIONS_H */
327
328
```

14.10 woss/ac-toolbox-woss.cpp File Reference

Provides the implementation of woss::ACToolboxWoss class.

14.10.1 Detailed Description

Provides the implementation of woss::ACToolboxWoss class.

Author

Federico Guerra

Provides the implementation of the woss::ACToolboxWoss class

14.11 woss/ac-toolbox-woss.h File Reference

Provides the interface for woss::ACToolboxWoss class.

Classes

- class woss::ACToolboxWoss

  base class for implementing acoustic-toolbox channel simulators (Bellhop, Kraken, etc...)

Typedefs

- typedef ::std::vector< SSP * > woss::SSPVector

14.11.1 Detailed Description

Provides the interface for woss::ACToolboxWoss class.

Author

Federico Guerra

Provides the interface for the woss::ACToolboxWoss class

14.11.2 Typedef Documentation

14.11.2.1 SSPVector `typedef ::std::vector< SSP* > woss::SSPVector`

A vector of SSP

14.12 ac-toolbox-woss.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef AC_TOOLBOX_WOSS_DEFINITIONS_H
41 #define AC_TOOLBOX_WOSS_DEFINITIONS_H
42
43
44 #include <ssp-definitions.h>
45 #include "woss.h"
46
47
48 namespace woss {
49
50
51   class Sediment;
52   class Altimetry;
53
57   typedef ::std::vector< SSP* > SSPVector;
58
59
66   class ACToolboxWoss :  public WossResReader {
```

```
67
68
69      public:
70
71
75      ACToolboxWoss();
76
87      ACToolboxWoss( const CoordZ& tx, const CoordZ& rx, const Time& start_t, const Time& end_t, double
        start_freq, double end_freq, double freq_step ) ;
88
92      virtual ~ACToolboxWoss();
93
94
99      virtual bool initialize();
100
101
106      virtual bool isValid() const ;
107
108
113      ACToolboxWoss& setRangeSteps( int steps ) { total_range_steps = steps; coordz_vector.reserve(steps +
        2);
114                                          range_vector.reserve(steps + 2); ssp_vector.reserve(steps + 2);
        return *this; }
115
120      ACToolboxWoss& setSSPDepthPrecision( long double precision ) { ssp_depth_precision = precision;
        return *this; }
121
122
127      int getRangeSteps()const { return total_range_steps; }
128
133      long double getSSPDepthPrecision()const { return ssp_depth_precision; }
134
139      int getMinSSPDepthSteps()const { return min_ssp_depth_steps; }
140
145      int getMaxSSPDepthSteps()const { return max_ssp_depth_steps; }
146
151      double getMinSSPDepth()const { return( *(min_ssp_depth_set.begin()) ); }
152
157      double getMaxSSPDepth()const { return( *(max_ssp_depth_set.rbegin()) ); }
158
163      double getMinBathymteryDepth()const { return min_bathymetry_depth; }
164
169      double getMaxBathymetryDepth()const { return max_bathymetry_depth; }
170
171
172      protected:
173
174
178      long double ssp_depth_precision;
179
180
184      double min_bathymetry_depth;
185
189      double max_bathymetry_depth;
190
191
195      double min_altimetry_depth;
196
200      double max_altimetry_depth;
201
202
206      ::std::set< double > min_ssp_depth_set;
207
211      ::std::set< double > max_ssp_depth_set;
212
213
217      int min_ssp_depth_steps;
218
222      int max_ssp_depth_steps;
223
227      int total_range_steps;
228
229
233      CoordZVector coordz_vector;
234
238      RangeVector range_vector;
239
243      SSPVector ssp_vector;
244
248      ::std::set< int > ssp_unique_indexes;
249
253      Sediment* sediment_value;
254
258      Altimetry* altimetry_value;
259
260
265      bool is_ssp_vector_transformable;
```

```
266
267
273     virtual bool checkSSPUnicity( SSP*& ptr );
274
275
280     virtual bool initRangeVector();
281
286     virtual bool initCoordZVector();
287
292     virtual bool initSediment();
293
298     virtual bool initAltimetry();
299
304     virtual bool initSSPVector();
305
309     virtual void resetSSPVector();
310
311   };
312
313
314 }
315
316
317 #endif /* AC_TOOLBOX_WOSS_DEFINITIONS_H */
318
319
```

14.13 woss/bellhop-creator.cpp File Reference

Provides the implementation of woss::BellhopCreator class.

14.13.1 Detailed Description

Provides the implementation of woss::BellhopCreator class.

Author

Federico Guerra

Provides the implementation of the woss::BellhopCreator class

14.14 woss/bellhop-creator.h File Reference

Provides the interface for woss::BellhopCreator class.

Classes

- struct woss::CustomAngles

  Bellhop min max angles.
- class woss::BellhopCreator

  class that provides correctly initialized BellhopWoss objects

14.14.1 Detailed Description

Provides the interface for woss::BellhopCreator class.

Author

Federico Guerra

Provides the interface for the woss::BellhopCreator class

14.15 bellhop-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_BELLHOP_CREATOR_DEFINITIONS_H
41 #define WOSS_BELLHOP_CREATOR_DEFINITIONS_H
42
43
44 #include "woss-creator.h"
45 #include "bellhop-woss.h"
46
47
48 namespace woss {
49
50
56   struct CustomAngles {
57
58
64     CustomAngles( double min = double(), double max = double() ) :  min_angle(min), max_angle(max) { }
65
66
67     friend std::ostream& operator«( std::ostream& os, const CustomAngles& instance ) {
68       os « "min angle = " « instance.min_angle « "; max_angle = " « instance.max_angle;
69       return os;
70     }
71
75     double min_angle;
76
80     double max_angle;
81
82
83   };
84
85
91   class BellhopCreator :  public WossCreator {
92
93
94     public:
95
99     BellhopCreator();
100
101     virtual ~BellhopCreator() { }
102
103
112     virtual BellhopWoss* const createWoss( const CoordZ& tx, const CoordZ& rx, double start_frequency,
      double end_frequency ) const;
113
114
120     BellhopCreator& setThorpeAttFlag( bool flag ) { use_thorpe_att = flag; return *this; }
121
126     bool getThorpeAttFlag() { return use_thorpe_att; }
127
128
136     BellhopCreator& setTotalRangeSteps( int steps, const CoordZ& tx, const CoordZ& rx ) {
137       cctotal_range_steps.replace(steps, tx, rx); return *this; }
138
146     BellhopCreator& setTotalRangeSteps( int steps, Location* const tx = CCInt::ALL_LOCATIONS, Location*
      const rx = CCInt::ALL_LOCATIONS ) {
```

```
147       cctotal_range_steps.replace(steps, tx, rx); return *this; }
148
155     BellhopCreator& eraseTotalRangeSteps( const CoordZ& tx, const CoordZ& rx ) {
156       cctotal_range_steps.erase(tx, rx); return *this; }
157
164     BellhopCreator& eraseTotalRangeSteps( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx
    = CCInt::ALL_LOCATIONS ) {
165       cctotal_range_steps.erase(tx, rx); return *this; }
166
173     double getTotalRangeSteps( const CoordZ& tx, const CoordZ& rx ) {
174       return cctotal_range_steps.get(tx, rx); }
175
182     double getTotalRangeSteps( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
183       return cctotal_range_steps.get(tx, rx); }
184
185
193     BellhopCreator& setTxMinDepthOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
194       cctx_min_depth_offset.replace(offset, tx, rx); return *this; }
195
203     BellhopCreator& setTxMinDepthOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
204       cctx_min_depth_offset.replace(offset, tx, rx); return *this; }
205
212     BellhopCreator& eraseTxMinDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
213       cctx_min_depth_offset.erase(tx, rx); return *this; }
214
221     BellhopCreator& eraseTxMinDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
222       cctx_min_depth_offset.erase(tx, rx); return *this; }
223
230     double getTxMinDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
231       return cctx_min_depth_offset.get(tx, rx); }
232
239     double getTxMinDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
240       return cctx_min_depth_offset.get(tx, rx); }
241
242
250     BellhopCreator& setTxMaxDepthOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
251       cctx_max_depth_offset.replace(offset, tx, rx); return *this; }
252
260     BellhopCreator& setTxMaxDepthOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
261       cctx_max_depth_offset.replace(offset, tx, rx); return *this; }
262
269     BellhopCreator& eraseTxMaxDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
270       cctx_max_depth_offset.erase(tx, rx); return *this; }
271
278     BellhopCreator& eraseTxMaxDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
279       cctx_max_depth_offset.erase(tx, rx); return *this; }
280
287     double getTxMaxDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
288       return cctx_max_depth_offset.get(tx, rx); }
289
296     double getTxMaxDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
297       return cctx_max_depth_offset.get(tx, rx); }
298
299
307     BellhopCreator& setTotalTransmitters( int sources, const CoordZ& tx, const CoordZ& rx ) {
308       cctotal_transmitters.replace(sources, tx, rx); return *this; }
309
317     BellhopCreator& setTotalTransmitters( int sources, Location* const tx = CCInt::ALL_LOCATIONS,
    Location* const rx = CCInt::ALL_LOCATIONS ) {
318       cctotal_transmitters.replace(sources, tx, rx); return *this; }
319
326     BellhopCreator& eraseTotalTransmitters( const CoordZ& tx, const CoordZ& rx ) {
327       cctotal_transmitters.erase(tx, rx); return *this; }
328
335     BellhopCreator& eraseTotalTransmitters( Location* const tx = CCInt::ALL_LOCATIONS, Location* const
    rx = CCInt::ALL_LOCATIONS ) {
336       cctotal_transmitters.erase(tx, rx); return *this; }
337
344     int getTotalTransmitters( const CoordZ& tx, const CoordZ& rx ) {
345       return cctotal_transmitters.get(tx, rx); }
346
353     int getTotalTransmitters( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
354       return cctotal_transmitters.get(tx, rx); }
355
356
364     BellhopCreator& setRxMinDepthOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
365       ccrx_min_depth_offset.replace(offset, tx, rx); return *this; }
366
374     BellhopCreator& setRxMinDepthOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
```

```
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
375     ccrx_min_depth_offset.replace(offset, tx, rx); return *this; }
376
383   BellhopCreator& eraseRxMinDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
384     ccrx_min_depth_offset.erase(tx, rx); return *this; }
385
392   BellhopCreator& eraseRxMinDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
393     ccrx_min_depth_offset.erase(tx, rx); return *this; }
394
401   double getRxMinDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
402     return ccrx_min_depth_offset.get(tx, rx); }
403
410   double getRxMinDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
411     return ccrx_min_depth_offset.get(tx, rx); }
412
413
421   BellhopCreator& setRxMaxDepthOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
422     ccrx_max_depth_offset.replace(offset, tx, rx); return *this; }
423
431   BellhopCreator& setRxMaxDepthOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
432     ccrx_max_depth_offset.replace(offset, tx, rx); return *this; }
433
440   BellhopCreator& eraseRxMaxDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
441     ccrx_max_depth_offset.erase(tx, rx); return *this; }
442
449   BellhopCreator& eraseRxMaxDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
450     ccrx_max_depth_offset.erase(tx, rx); return *this; }
451
458   double getRxMaxDepthOffset( const CoordZ& tx, const CoordZ& rx ) {
459     return ccrx_max_depth_offset.get(tx, rx); }
460
468   double getRxMaxDepthOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
469     return ccrx_max_depth_offset.get(tx, rx); }
470
471
479   BellhopCreator& setRxMinRangeOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
480     ccrx_min_range_offset.replace(offset, tx, rx); return *this; }
481
489   BellhopCreator& setRxMinRangeOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
490     ccrx_min_range_offset.replace(offset, tx, rx); return *this; }
491
498   BellhopCreator& eraseRxMinRangeOffset( const CoordZ& tx, const CoordZ& rx ) {
499     ccrx_min_range_offset.erase(tx, rx); return *this; }
500
507   BellhopCreator& eraseRxMinRangeOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
508     ccrx_min_range_offset.erase(tx, rx); return *this; }
509
516   double getRxMinRangeOffset( const CoordZ& tx, const CoordZ& rx ) {
517     return ccrx_min_range_offset.get(tx, rx); }
518
525   double getRxMinRangeOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
526     return ccrx_min_range_offset.get(tx, rx); }
527
528
536   BellhopCreator& setRxMaxRangeOffset( double offset, const CoordZ& tx, const CoordZ& rx ) {
537     ccrx_max_range_offset.replace(offset, tx, rx); return *this; }
538
546   BellhopCreator& setRxMaxRangeOffset( double offset, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
547     ccrx_max_range_offset.replace(offset, tx, rx); return *this; }
548
555   BellhopCreator& eraseRxMaxRangeOffset( const CoordZ& tx, const CoordZ& rx ) {
556     ccrx_max_range_offset.erase(tx, rx); return *this; }
557
564   BellhopCreator& eraseRxMaxRangeOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
    rx = CCDouble::ALL_LOCATIONS ) {
565     ccrx_max_range_offset.erase(tx, rx); return *this; }
566
573   double getRxMaxRangeOffset( const CoordZ& tx, const CoordZ& rx ) {
574     return ccrx_max_range_offset.get(tx, rx); }
575
582   double getRxMaxRangeOffset( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
583     return ccrx_max_range_offset.get(tx, rx); }
584
585
593   BellhopCreator& setRxTotalDepths( int number, const CoordZ& tx, const CoordZ& rx ) {
594     cctotal_rx_depths.replace(number, tx, rx); return *this; }
595
```

```
603     BellhopCreator& setRxTotalDepths( int number, Location* const tx = CCInt::ALL_LOCATIONS, Location*
    const rx = CCInt::ALL_LOCATIONS ) {
604       cctotal_rx_depths.replace(number, tx, rx); return *this; }
605
612     BellhopCreator& eraseRxTotalDepths( const CoordZ& tx, const CoordZ& rx ) {
613       cctotal_rx_depths.erase(tx, rx); return *this; }
614
621     BellhopCreator& eraseRxTotalDepths( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
622       cctotal_rx_depths.erase(tx, rx); return *this; }
623
630     int getRxTotalDepths( const CoordZ& tx, const CoordZ& rx ) {
631       return cctotal_rx_depths.get(tx, rx); }
632
639     int getRxTotalDepths( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
640       return cctotal_rx_depths.get(tx, rx); }
641
642
650     BellhopCreator& setRxTotalRanges( int number, const CoordZ& tx, const CoordZ& rx ) {
651       cctotal_rx_ranges.replace(number, tx, rx); return *this; }
652
660     BellhopCreator& setRxTotalRanges( int number, Location* const tx = CCInt::ALL_LOCATIONS, Location*
    const rx = CCInt::ALL_LOCATIONS ) {
661       cctotal_rx_ranges.replace(number, tx, rx); return *this; }
662
669     BellhopCreator& eraseRxTotalRanges( const CoordZ& tx, const CoordZ& rx ) {
670       cctotal_rx_ranges.erase(tx, rx); return *this; }
671
678     BellhopCreator& eraseRxTotalRanges( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
679       cctotal_rx_ranges.erase(tx, rx); return *this; }
680
687     int getRxTotalRanges( const CoordZ& tx, const CoordZ& rx ) {
688       return cctotal_rx_ranges.get(tx, rx); }
689
696     int getRxTotalRanges( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
697       return cctotal_rx_ranges.get(tx, rx); }
698
699
707     BellhopCreator& setRaysNumber( int number, const CoordZ& tx, const CoordZ& rx ) {
708       cctotal_rays.replace(number, tx, rx); return *this; }
709
717     BellhopCreator& setRaysNumber( int number, Location* const tx = CCInt::ALL_LOCATIONS, Location*
    const rx = CCInt::ALL_LOCATIONS ) {
718       cctotal_rays.replace(number, tx, rx); return *this; }
719
726     BellhopCreator& eraseRaysNumber( const CoordZ& tx, const CoordZ& rx ) {
727       cctotal_rays.erase(tx, rx); return *this; }
728
735     BellhopCreator& eraseRaysNumber( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
736       cctotal_rays.erase(tx, rx); return *this; }
737
744     int getRaysNumber( const CoordZ& tx, const CoordZ& rx ) {
745       return cctotal_rays.get(tx, rx); }
746
753     int getRaysNumber( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
754       return cctotal_rays.get(tx, rx); }
755
756
762     BellhopCreator& setBellhopPath( const ::std::string& path ) { bellhop_path = path; return *this; }
763
768     ::std::string getBellhopPath() { return bellhop_path; }
769
775     BellhopCreator& setBellhopArrSyntax( BellhopArrSyntax syntax ) {
776       bellhop_arr_syntax = syntax; return *this; }
777
782     BellhopArrSyntax getBellhopArrSyntax() { return bellhop_arr_syntax; }
783
789     BellhopCreator& setBellhopShdSyntax( BellhopShdSyntax syntax ) {
790       bellhop_shd_syntax = syntax; return *this; }
791
796     BellhopShdSyntax getBellhopShdSyntax() { return bellhop_shd_syntax; }
797
806     BellhopCreator& setBeamOptions( const ::std::string& options, const CoordZ& tx, const CoordZ& rx ) {
807       ccbeam_options.replace(options, tx, rx); return *this; }
808
817     BellhopCreator& setBeamOptions( const ::std::string& options, Location* const tx =
    CCString::ALL_LOCATIONS, Location* const rx = CCString::ALL_LOCATIONS ) {
818       ccbeam_options.replace(options, tx, rx); return *this; }
819
826     BellhopCreator& eraseBeamOptions( const CoordZ& tx, const CoordZ& rx ) {
827       ccbeam_options.erase(tx, rx); return *this; }
828
```

```
835     BellhopCreator& eraseBeamOptions( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
      CCString::ALL_LOCATIONS ) {
836       ccbeam_options.erase(tx, rx); return *this; }
837
844     ::std::string getBeamOptions( const CoordZ& tx, const CoordZ& rx ) {
845       return ccbeam_options.get(tx, rx); }
846
853     ::std::string getBeamOptions( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
      CCString::ALL_LOCATIONS ) {
854       return ccbeam_options.get(tx, rx); }
855
856
865     BellhopCreator& setBhMode( const ::std::string& options, const CoordZ& tx, const CoordZ& rx ) {
866       ccbellhop_mode.replace(options, tx, rx); return *this; }
867
876     BellhopCreator& setBhMode( const ::std::string& options, Location* const tx =
      CCString::ALL_LOCATIONS, Location* const rx = CCString::ALL_LOCATIONS ) {
877       ccbellhop_mode.replace(options, tx, rx); return *this; }
878
885     BellhopCreator& eraseBhMode( const CoordZ& tx, const CoordZ& rx ) {
886       ccbellhop_mode.erase(tx, rx); return *this; }
887
894     BellhopCreator& eraseBhMode( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
      CCString::ALL_LOCATIONS ) {
895       ccbellhop_mode.erase(tx, rx); return *this; }
896
903     ::std::string getBhMode( const CoordZ& tx, const CoordZ& rx ) {
904       return ccbellhop_mode.get(tx, rx); }
905
912     ::std::string getBhMode( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
      CCString::ALL_LOCATIONS ) {
913       return ccbellhop_mode.get(tx, rx); }
914
915
924     BellhopCreator& setBathymetryType( const ::std::string& options, const CoordZ& tx, const CoordZ& rx
      ) {
925       ccbathymetry_type.replace(options, tx, rx); return *this; }
926
935     BellhopCreator& setBathymetryType( const ::std::string& options, Location* const tx =
      CCString::ALL_LOCATIONS, Location* const rx = CCString::ALL_LOCATIONS ) {
936       ccbathymetry_type.replace(options, tx, rx); return *this; }
937
944     BellhopCreator& eraseBathymetryType( const CoordZ& tx, const CoordZ& rx ) {
945       ccbathymetry_type.erase(tx, rx); return *this; }
946
953     BellhopCreator& eraseBathymetryType( Location* const tx = CCString::ALL_LOCATIONS, Location* const
      rx = CCString::ALL_LOCATIONS ) {
954       ccbathymetry_type.erase(tx, rx); return *this; }
955
962     ::std::string getBathymetryType( const CoordZ& tx, const CoordZ& rx ) {
963       return ccbathymetry_type.get(tx, rx); }
964
971     ::std::string getBathymetryType( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
      CCString::ALL_LOCATIONS ) {
972       return ccbathymetry_type.get(tx, rx); }
973
982     BellhopCreator& setBathymetryMethod( const ::std::string& options, const CoordZ& tx, const CoordZ&
      rx ) {
983       ccbathymetry_method.replace(options, tx, rx); return *this; }
984
993     BellhopCreator& setBathymetryMethod( const ::std::string& options, Location* const tx =
      CCString::ALL_LOCATIONS, Location* const rx = CCString::ALL_LOCATIONS ) {
994       ccbathymetry_method.replace(options, tx, rx); return *this; }
995
1002    BellhopCreator& eraseBathymetryMethod( const CoordZ& tx, const CoordZ& rx ) {
1003      ccbathymetry_method.erase(tx, rx); return *this; }
1004
1011    BellhopCreator& eraseBathymetryMethod( Location* const tx = CCString::ALL_LOCATIONS, Location*
      const rx = CCString::ALL_LOCATIONS ) {
1012      ccbathymetry_method.erase(tx, rx); return *this; }
1013
1020    ::std::string getBathymetryMethod( const CoordZ& tx, const CoordZ& rx ) {
1021      return ccbathymetry_method.get(tx, rx); }
1022
1029    ::std::string getBathymetryMethod( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx
      = CCString::ALL_LOCATIONS ) {
1030      return ccbathymetry_method.get(tx, rx); }
1031
1040    BellhopCreator& setAltimetryType( const ::std::string& options, const CoordZ& tx, const CoordZ& rx
      ) {
1041      ccaltimetry_type.replace(options, tx, rx); return *this; }
1042
1051    BellhopCreator& setAltimetryType( const ::std::string& options, Location* const tx =
      CCString::ALL_LOCATIONS, Location* const rx = CCString::ALL_LOCATIONS ) {
1052      ccaltimetry_type.replace(options, tx, rx); return *this; }
1053
1060    BellhopCreator& eraseAltimetryType( const CoordZ& tx, const CoordZ& rx ) {
```

```
1061       ccaltimetry_type.erase(tx, rx); return *this; }
1062
1069     BellhopCreator& eraseAltimetryType( Location* const tx = CCString::ALL_LOCATIONS, Location* const
       rx = CCString::ALL_LOCATIONS ) {
1070       ccaltimetry_type.erase(tx, rx); return *this; }
1071
1078     ::std::string getAltimetryType( const CoordZ& tx, const CoordZ& rx ) {
1079       return ccaltimetry_type.get(tx, rx); }
1080
1087     ::std::string getAltimetryType( Location* const tx = CCString::ALL_LOCATIONS, Location* const rx =
       CCString::ALL_LOCATIONS ) {
1088       return ccaltimetry_type.get(tx, rx); }
1089
1090
1098     BellhopCreator& setAngles( const CustomAngles& angles, const CoordZ& tx, const CoordZ& rx ) {
1099       ccangles_map.replace(angles, tx, rx); return *this; }
1100
1108     BellhopCreator& setAngles( const CustomAngles& angles, Location* const tx =
       CCAngles::ALL_LOCATIONS, Location* const rx = CCAngles::ALL_LOCATIONS ) {
1109       ccangles_map.replace(angles, tx, rx); return *this; }
1110
1117     CustomAngles getAngles( const CoordZ& tx, const CoordZ& rx )const { return ccangles_map.get(tx,
       rx); }
1118
1125     CustomAngles getAngles( Location* const tx = CCAngles::ALL_LOCATIONS, Location* const rx =
       CCAngles::ALL_LOCATIONS )const {
1126       return ccangles_map.get(tx, rx); }
1127
1134     BellhopCreator& eraseAngles( const CoordZ& tx, const CoordZ& rx ) { ccangles_map.erase(tx, rx);
       return *this; }
1135
1142     BellhopCreator& eraseAngles( Location* const tx = CCAngles::ALL_LOCATIONS, Location* const rx =
       CCAngles::ALL_LOCATIONS ) {
1143       ccangles_map.erase(tx, rx); return *this;
1144     }
1145
1153         BellhopCreator& setBoxDepth( double box_depth, const CoordZ& tx, const CoordZ& rx ) {
1154           ccbox_depth.replace(box_depth, tx, rx); return *this; }
1155
1163       BellhopCreator& setBoxDepth( double box_depth, Location* const tx = CCDouble::ALL_LOCATIONS,
       Location* const rx = CCDouble::ALL_LOCATIONS ) {
1164         ccbox_depth.replace(box_depth, tx, rx); return *this; }
1165
1166
1173       BellhopCreator& eraseBoxDepth( const CoordZ& tx, const CoordZ& rx ) {
1174         ccbox_depth.erase(tx, rx); return *this; }
1175
1182           BellhopCreator& eraseBoxDepth( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
       rx = CCDouble::ALL_LOCATIONS ) {
1183                 ccbox_depth.erase(tx, rx); return *this; }
1184
1191       double getBoxDepth( const CoordZ& tx, const CoordZ& rx ) {
1192         return ccbox_depth.get(tx, rx); }
1193
1200       double getBoxDepth( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
       CCDouble::ALL_LOCATIONS ) {
1201         return ccbox_depth.get(tx, rx); }
1202
1210         BellhopCreator& setBoxRange( double box_range, const CoordZ& tx, const CoordZ& rx ) {
1211           ccbox_range.replace(box_range, tx, rx); return *this; }
1212
1220                 BellhopCreator& setBoxRange( double box_range, Location* const tx =
       CCDouble::ALL_LOCATIONS, Location* const rx = CCDouble::ALL_LOCATIONS ) {
1221                   ccbox_range.replace(box_range, tx, rx); return *this; }
1222
1223
1230                 BellhopCreator& eraseBoxRange( const CoordZ& tx, const CoordZ& rx ) {
1231                   ccbox_range.erase(tx, rx); return *this; }
1232
1239           BellhopCreator& eraseBoxRange( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const
       rx = CCDouble::ALL_LOCATIONS ) {
1240                 ccbox_range.erase(tx, rx); return *this; }
1241
1248                 double getBoxRange( const CoordZ& tx, const CoordZ& rx ) {
1249                   return ccbox_range.get(tx, rx); }
1250
1257                 double getBoxRange( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
       CCDouble::ALL_LOCATIONS ) {
1258                   return ccbox_range.get(tx, rx); }
1259
1267     BellhopCreator& setSspDepthPrecision( double ssp_precision, const CoordZ& tx, const CoordZ& rx ) {
1268       ccssp_depth_precision.replace(ssp_precision, tx, rx); return *this; }
1269
1277     BellhopCreator& setSspDepthPrecision( double ssp_precision, Location* const tx =
       CCDouble::ALL_LOCATIONS, Location* const rx = CCDouble::ALL_LOCATIONS ) {
1278       ccssp_depth_precision.replace(ssp_precision, tx, rx); return *this; }
1279
```

```
1286     BellhopCreator& eraseSspDepthPrecision( const CoordZ& tx, const CoordZ& rx ) {
1287       ccssp_depth_precision.erase(tx, rx); return *this; }
1288
1295     BellhopCreator& eraseSspDepthPrecision( Location* const tx = CCDouble::ALL_LOCATIONS, Location*
    const rx = CCDouble::ALL_LOCATIONS ) {
1296       ccssp_depth_precision.erase(tx, rx); return *this; }
1297
1304     double getSspDepthPrecision( const CoordZ& tx, const CoordZ& rx ) {
1305       return ccssp_depth_precision.get(tx, rx); }
1306
1313     double getSspDepthPrecision( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
1314       return ccssp_depth_precision.get(tx, rx); }
1315
1316
1324     BellhopCreator& setSspDepthSteps( int ssp_depth_steps, const CoordZ& tx, const CoordZ& rx ) {
1325       ccnormalized_ssp_depth_steps.replace(ssp_depth_steps, tx, rx); return *this; }
1326
1334     BellhopCreator& setSspDepthSteps( int ssp_depth_steps, Location* const tx = CCInt::ALL_LOCATIONS,
    Location* const rx = CCInt::ALL_LOCATIONS ) {
1335       ccnormalized_ssp_depth_steps.replace(ssp_depth_steps, tx, rx); return *this; }
1336
1343     BellhopCreator& eraseSspDepthSteps( const CoordZ& tx, const CoordZ& rx ) {
1344       ccnormalized_ssp_depth_steps.erase(tx, rx); return *this; }
1345
1352     BellhopCreator& eraseSspDepthSteps( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
1353       ccnormalized_ssp_depth_steps.erase(tx, rx); return *this; }
1354
1361     int getSspDepthSteps( const CoordZ& tx, const CoordZ& rx ) {
1362       return ccnormalized_ssp_depth_steps.get(tx, rx); }
1363
1370     int getSspDepthSteps( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
1371       return ccnormalized_ssp_depth_steps.get(tx, rx); }
1372
1373
1381     BellhopCreator& setCustomTransducer( const CustomTransducer& type, const CoordZ& tx, const CoordZ&
    rx ) {
1382       cctransducer.replace(type, tx, rx); return *this; }
1383
1391     BellhopCreator& setCustomTransducer( const CustomTransducer& type, Location* const tx =
    CCTransducer::ALL_LOCATIONS, Location* const rx = CCTransducer::ALL_LOCATIONS ) {
1392       cctransducer.replace(type, tx, rx); return *this; }
1393
1400     CustomTransducer getCustomTransducer( const CoordZ& tx, const CoordZ& rx )const {
1401       return cctransducer.get(tx, rx); }
1402
1409     CustomTransducer getCustomTransducer( Location* const tx = CCTransducer::ALL_LOCATIONS, Location*
    const rx = CCTransducer::ALL_LOCATIONS )const {
1410       return cctransducer.get(tx, rx); }
1411
1418     BellhopCreator& eraseCustomTransducer( const CoordZ& tx, const CoordZ& rx ) {
    ccangles_map.erase(tx, rx); return *this; }
1419
1426     BellhopCreator& eraseCustomTransducer( Location* const tx = CCTransducer::ALL_LOCATIONS, Location*
    const rx = CCTransducer::ALL_LOCATIONS ) {
1427       cctransducer.erase(tx, rx); return *this;
1428     }
1429
1430
1434     typedef WossCreatorContainer< CustomAngles > CCAngles;
1435
1436     typedef WossCreatorContainer< CustomTransducer > CCTransducer;
1437
1438     typedef WossCreatorContainer< ::std::string > CCString;
1439
1440
1441     protected:
1442
1446     bool use_thorpe_att;
1447
1448
1452     ::std::string bellhop_path;
1453
1457     BellhopArrSyntax bellhop_arr_syntax;
1458
1462     BellhopShdSyntax bellhop_shd_syntax;
1463
1467     CCString ccbellhop_mode;
1468
1472     CCString ccbeam_options;
1473
1477     CCString ccbathymetry_type;
1478
1482     CCString ccbathymetry_method;
1483
```

```
1487     CCString ccaltimetry_type;
1488
1489
1493     CCAngles ccangles_map;
1494
1495
1499     CCInt cctotal_range_steps;
1500
1501
1505     CCInt cctotal_transmitters;
1506
1510     CCDouble cctx_min_depth_offset;
1511
1515     CCDouble cctx_max_depth_offset;
1516
1517
1521     CCInt cctotal_rx_depths;
1522
1526     CCDouble ccrx_min_depth_offset;
1527
1531     CCDouble ccrx_max_depth_offset;
1532
1533
1540     CCInt cctotal_rx_ranges;
1541
1545     CCDouble ccrx_min_range_offset;
1546
1550     CCDouble ccrx_max_range_offset;
1551
1552
1556     CCInt cctotal_rays;
1557
1558
1562     CCDouble ccssp_depth_precision;
1563
1567     CCInt ccnormalized_ssp_depth_steps;
1568
1572     CCTransducer cctransducer;
1573
1577     CCDouble ccbox_depth;
1578
1582     CCDouble ccbox_range;
1583
1589     virtual bool initializeWoss( Woss* const woss_ptr ) const ;
1590
1596     bool initializeBhWoss( BellhopWoss* const woss_ptr ) const ;
1597
1598
1599     virtual const BellhopWoss* createNotValidWoss() const;
1600
1601     virtual void updateDebugFlag();
1602
1603   };
1604
1605
1606 }
1607
1608
1609 #endif /* WOSS_BELLHOP_CREATOR_DEFINITIONS_H */
1610
```

14.16 woss/bellhop-woss.cpp File Reference

Provides the implementation of woss::BellhopWoss class.

14.16.1 Detailed Description

Provides the implementation of woss::BellhopWoss class.

Author

Federico Guerra

Provides the implementation of woss::BellhopWoss class

14.17 woss/bellhop-woss.h File Reference

Provides the interface for woss::BellhopWoss class.

Classes

- class woss::BellhopWoss

  Implempentation of ACToolboxWoss for Bellhop raytracing program.

Typedefs

- typedef ::std::map< double, SSP * > **woss::NormSSPMap**
- typedef NormSSPMap::iterator **woss::NSMIter**
- typedef NormSSPMap::const_iterator **woss::NSMCIter**
- typedef NormSSPMap::reverse_iterator **woss::NSMRIter**
- typedef NormSSPMap::const_reverse_iterator **woss::NSMCRIter**

Enumerations

- enum woss::BellhopArrSyntax { woss::BELLHOP_CREATOR_ARR_FILE_SYNTAX_0 = 0 , woss::BELLHOP_CREATOR_ARR_ , woss::BELLHOP_CREATOR_ARR_FILE_SYNTAX_2 , woss::BELLHOP_CREATOR_ARR_FILE_INVALID }
- enum woss::BellhopShdSyntax { **BELLHOP_CREATOR_SHD_FILE_SYNTAX_0** = 0 , **BELLHOP_CREATOR_SHD_FILE_SYNTAX_1** , woss::BELLHOP_CREATOR_SHD_FILE_INVALID }

14.17.1 Detailed Description

Provides the interface for woss::BellhopWoss class.

Author

Federico Guerra

Provides the interface for woss::BellhopWoss class

14.17.2 Enumeration Type Documentation

14.17.2.1 BellhopArrSyntax `enum woss::BellhopArrSyntax`

Enumerator

BELLHOP_CREATOR_ARR_FILE_SYNTAX_0	Pre 31 august 2016 syntax, without imaginary time delay.
BELLHOP_CREATOR_ARR_FILE_SYNTAX_1	Post 31 August 2016 syntax, with imaginary time delay.
BELLHOP_CREATOR_ARR_FILE_SYNTAX_2	Post 31 March 2019 syntax, with different header syntax.
BELLHOP_CREATOR_ARR_FILE_INVALID	invalid syntax, must always be the last element

14.17.2.2 BellhopShdSyntax enum woss::BellhopShdSyntax

Enumerator

BELLHOP_CREATOR_SHD_FILE_INVALID	invalid syntax, must always be the last element

14.18 bellhop-woss.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
39 #ifndef WOSS_BELLHOP_H
40 #define WOSS_BELLHOP_H
41
42
43 #include <iomanip>
44 #include <definitions.h>
45 #include <sediment-definitions.h>
46 #include <transducer-definitions.h>
47 #include "ac-toolbox-arr-asc-reader.h"
48 #include "ac-toolbox-arr-bin-reader.h"
49 #include "ac-toolbox-shd-reader.h"
50 #include "ac-toolbox-woss.h"
51
52
53 namespace woss {
54
55
56   typedef ::std::map< double, SSP* > NormSSPMap;
57   typedef NormSSPMap::iterator NSMIter;
58   typedef NormSSPMap::const_iterator NSMCIter;
59   typedef NormSSPMap::reverse_iterator NSMRIter;
60   typedef NormSSPMap::const_reverse_iterator NSMCRIter;
61
62   /*
63 * .arr file syntax to be used during parsing
64 */
65   enum BellhopArrSyntax {
66     BELLHOP_CREATOR_ARR_FILE_SYNTAX_0 = 0,
67     BELLHOP_CREATOR_ARR_FILE_SYNTAX_1,
68     BELLHOP_CREATOR_ARR_FILE_SYNTAX_2,
69     BELLHOP_CREATOR_ARR_FILE_INVALID
70   };
71
72
73   /*
74 * .shd file syntax to be used during parsing
```

```
75 */
76   enum BellhopShdSyntax {
77     BELLHOP_CREATOR_SHD_FILE_SYNTAX_0 = 0,
78     BELLHOP_CREATOR_SHD_FILE_SYNTAX_1,
79     BELLHOP_CREATOR_SHD_FILE_INVALID
80   };
81
87   class BellhopWoss :  public ACToolboxWoss {
88
89
90     public:
91
92
96     BellhopWoss();
97
108     BellhopWoss(const CoordZ& tx, const CoordZ& rx, const Time& start_t, const Time& end_t, double
      start_freq, double end_freq, double freq_step ) ;
109
110     virtual ~BellhopWoss();
111
112
117     virtual bool initialize();
118
124     virtual bool initPressResReader( double curr_frequency );
125
131     virtual bool initTimeArrResReader( double curr_frequency );
132
133
139     virtual bool run();
140
141
147     virtual bool timeEvolve( const Time& time_value );
148
153     virtual bool isValid() const;
154
155
156     virtual Pressure* getAvgPressure( double frequency, double tx_depth, double start_rx_depth =
      WOSS_MIN_DEPTH, double start_rx_range = WOSS_MIN_RANGE, double end_rx_depth = WOSS_MAX_DEPTH , double
      end_rx_range = WOSS_MAX_RANGE ) const;
157
158     virtual Pressure* getPressure( double frequency, double tx_depth, double rx_depth, double rx_range )
      const;
159
160
161     virtual TimeArr* getTimeArr( double frequency, double tx_depth, double rx_depth, double rx_range )
      const;
162
168     BellhopWoss& setThorpeAttFlag( bool flag ) { use_thorpe_att = flag; return *this; }
169
175     BellhopWoss& setTxMinDepthOffset( double offset ) { tx_min_depth_offset = offset; return *this; }
176
182     BellhopWoss& setTxMaxDepthOffset( double offset ) { tx_max_depth_offset = offset; return *this; }
183
189     BellhopWoss& setTotalTransmitters( int sources ) { total_transmitters = sources; return *this; }
190
196     BellhopWoss& setRxMinDepthOffset( double offset ) { rx_min_depth_offset = offset; return *this; }
197
203     BellhopWoss& setRxMaxDepthOffset( double offset ) { rx_max_depth_offset = offset; return *this; }
204
210     BellhopWoss& setRxMinRangeOffset( double offset ) { rx_min_range_offset = offset; return *this; }
211
217     BellhopWoss& setRxMaxRangeOffset( double offset ) { rx_max_range_offset = offset; return *this; }
218
224     BellhopWoss& setRxTotalDepths( int number ) { total_rx_depths = number; return *this; }
225
231     BellhopWoss& setRxTotalRanges( int number ) { total_rx_ranges = number; return *this; }
232
238     BellhopWoss& setRaysNumber( int number ) { total_rays = number; return *this; }
239
245     BellhopWoss& setMinAngle( double angle ) { min_angle = angle; return *this; }
246
252     BellhopWoss& setMaxAngle( double angle ) { max_angle = angle; return *this; }
253
259     BellhopWoss& setBoxDepth( double depth ) { box_depth = depth; return *this; }
260
266     BellhopWoss& setBoxRange( double range ) { box_range = range; return *this; }
267
273     BellhopWoss& setBellhopPath( const ::std::string& path ) { bellhop_path = path; return *this; }
274
280     BellhopWoss& setBellhopArrSyntax( BellhopArrSyntax syntax ) { bellhop_arr_syntax = syntax; return
      *this; }
281
287     BellhopWoss& setBellhopShdSyntax( BellhopShdSyntax syntax ) { bellhop_shd_syntax = syntax; return
      *this; }
288
294     BellhopWoss& setBeamOptions( const ::std::string& options ) { beam_options = options; return *this;
      }
```

```
295
301     BellhopWoss& setBhMode( const ::std::string& mode );
302
308     BellhopWoss& setBathymetryType( const ::std::string& type ) { bathymetry_type = type; return *this;
    }
309
315     BellhopWoss& setBathymetryMethod( const ::std::string& type ) { bathymetry_method = type; return
    *this; }
316
322     BellhopWoss& setAltimetryType( const ::std::string& type ) { altimetry_type = type; return *this; }
323
324
330     BellhopWoss& setTransducer( const Transducer* const ptr ) { transducer = ptr; return *this; }
331
332
338     BellhopWoss& setTransformSSPDepthSteps( int depth_steps ) { transform_ssp_depth_steps = depth_steps;
    return *this; }
339
340
346     BellhopWoss& setBeamPatternParam( double init_bearing, double vert_rot = 0.0, double horiz_rot =
    0.0, double mult = 1.0, double add = 0.0 ) {
347       bp_initial_bearing = init_bearing; bp_vertical_rotation = vert_rot; bp_horizontal_rotation =
    horiz_rot;
348       bp_mult_costant = mult; bp_add_costant = add; return *this; }
349
350
351   //    BellhopWoss& useBeamBlock( bool command ) { using_beam_block = command; }
352
357     bool getThorpeAttFlag()const { return use_thorpe_att; }
358
363     double getTxMinDepthOffset()const { return tx_min_depth_offset; }
364
369     double getTxMaxDepthOffset()const { return tx_max_depth_offset; }
370
375     int getTotalTransmitters()const { return total_transmitters; }
376
381     double getRxMinDepthOffset()const { return rx_min_depth_offset; }
382
387     double getRxMaxDepthOffset()const { return rx_max_depth_offset; }
388
393     double getRxMinRangeOffset()const { return rx_min_range_offset; }
394
399     double getRxMaxRangeOffset()const { return rx_max_range_offset; }
400
405     int getRxTotalDepths()const { return total_rx_depths; }
406
411     int getRxTotalRanges()const { return total_rx_ranges; }
412
417     int getRaysNumber()const { return total_rays; }
418
423     double getMinAngle()const { return min_angle; }
424
429     double getMaxAngle()const { return max_angle; }
430
435     double getBoxDepth()const { return box_depth; }
436
441     double getBoxRange()const { return box_range; }
442
447     int getTransformSSPDepthSteps()const  { return transform_ssp_depth_steps; }
448
449
454     ::std::string getBellhopPath()const { return bellhop_path; }
455
460     BellhopArrSyntax getBellhopArrSyntax()const { return bellhop_arr_syntax; }
461
466     BellhopShdSyntax getBellhopShdSyntax()const { return bellhop_shd_syntax; }
467
472     ::std::string getBeamOptions()const { return beam_options; }
473
478     ::std::string getBathymetryType()const { return bathymetry_type; }
479
484     ::std::string getBathymetryMethod()const { return bathymetry_method; }
485
490     ::std::string getAltimetryType()const { return altimetry_type; }
491
496     const Transducer* const getTransducer()const { return transducer; }
497
498
503     bool isValidBhMode( const ::std::string& ) const ;
504
505
506   //    bool usingBeamBlock() const { return using_beam_block; }
507
512     bool usingPressMode()const { return using_press_mode; }
513
518     bool usingTimeArrMode()const { return using_time_arrival_mode; }
519
```

```
524     bool usingSSPFile()const { return using_ssp_file; }
525
526
527     protected:
528
532     bool use_thorpe_att;
533
537     ::std::string beam_options;
538
542     ::std::string bathymetry_type;
543
547     ::std::string bathymetry_method;
548
552     ::std::string altimetry_type;
553
557     ::std::string bellhop_op_mode;
558
559
563     ::std::string bellhop_env_file;
564
568     ::std::string bathymetry_file;
569
573     ::std::string altimetry_file;
574
578     ::std::string beam_pattern_file;
579
580
584     ::std::string ssp_file;
585
589     ::std::string shd_file;
590
594     ::std::string arr_file;
595
599     ::std::string bellhop_path;
600
604     BellhopArrSyntax bellhop_arr_syntax;
605
609     BellhopShdSyntax bellhop_shd_syntax;
610
614     ::std::string curr_path;
615
616     ::std::string transducer_type;
617
618
622     double tx_min_depth_offset;
623
627     double tx_max_depth_offset;
628
632     int total_transmitters;
633
634
638     int total_rx_depths;
639
643     double rx_min_depth_offset;
644
648     double rx_max_depth_offset;
649
650
657     int total_rx_ranges;
658
662     double rx_min_range_offset;
663
667     double rx_max_range_offset;
668
669
673     int total_rays;
674
675
679     double min_angle;
680
684     double max_angle;
685
686
690     double min_normalized_ssp_depth;
691
695     double max_normalized_ssp_depth;
696
700     int curr_norm_ssp_depth_steps;
701
702
706     int transform_ssp_depth_steps;
707
708
712     const Transducer* transducer;
713
714
715     double bp_initial_bearing;
```

```
716
717     double bp_vertical_rotation;
718
719     double bp_horizontal_rotation;
720
721     double bp_mult_costant;
722
723     double bp_add_costant;
724
725
726     bool using_ssp_file;
727
728     bool using_press_mode;
729
730     bool using_time_arrival_mode;
731
732   //   bool using_beam_block;
733
737     NormSSPMap normalized_ssp_map;
738
742     NormSSPMap randomized_ssp_map;
743
747     double box_depth;
748
752     double box_range;
753
754
758     ::std::ofstream f_out;
759
760
764     virtual void normalizeDbSSP();
765
769     virtual void resetNormalizedDbSSP();
770
771 //     bool initTransducer();
772
778     bool initResReader( double curr_frequency );
779
785     void initBox( double depth, double range );
786
792     void initCfgFiles( double curr_frequency, int curr_run );
793
794
800     void writeCfgFiles( double curr_frequency, int curr_run );
801
805     void writeAllCfgFiles();
806
811     void writeNormalizedSSP( int curr_run );
812
816     void writeSediment();
817
823     void writeHeader( double curr_frequency, int curr_run );
824
828     void writeTransmitter();
829
833     void writeReceiver();
834
838     void writeRayOptions();
839
843     void writeBox();
844
845     void writeBeamBlock();
846
850     void writeBathymetryFile();
851
855     void writeBeamPatternFile();
856
861     void writeAltimetryFile( int curr_run );
862
863
867     void removeAllCfgFiles();
868
874     void removeCfgFiles( double curr_frequency, int curr_run );
875
876
886     void checkBoundaries( double& frequency, double& tx_depth, double& rx_start_depth, double&
      rx_start_range, double& rx_end_depth, double& rx_end_range ) const;
887
891     void checkDepthOffsets();
892
901     void checkDepthOffsets( const CoordZ& coords, double& min_offset, double& max_offset, double
      min_depth_value, double max_depth_value );
902
906     void checkAngles();
907
911     void checkRangeOffsets();
912
```

```
913
914   };
915
916
917   //inline functions
919   inline bool BellhopWoss::isValidBhMode( const ::std::string& mode )const {
920     return (  mode == "a" || mode == "A" || mode == "C" || mode == "I" || mode == "S" );
921   }
922
923
924   inline void BellhopWoss::initBox( double depth, double range ) {
925           if(box_depth < 0) //box depth not set by user - default to depth + 10%
926                   box_depth = depth + depth / 10.0;
927
928           if(box_range < 0) //box range not set by user - default to range + 10%
929                   box_range = range + range / 10.0;
930
931   }
932
933
934   inline void BellhopWoss::writeSediment() {
935     f_out « "\'A*\' 0.0" « ::std::setw(30) « "!  BOTTOM TYPE" « ::std::endl
936           « max_normalized_ssp_depth « " " « sediment_value->getStringValues() «  "  /  !  "
937           « sediment_value->getType() « " BOTTOM TYPE " « ::std::endl;
938   }
939
940
941   inline void BellhopWoss::writeHeader( double curr_frequency, int curr_run ) {
942     f_out.precision(WOSS_DECIMAL_PRECISION);
943     f_out « "\'BELLHOP - woss id = " « woss_id « "; run = " « curr_run « "\'" « ::std::endl
944           « curr_frequency « ::std::setw(30) « "!  FREQUENCY [HZ]" « ::std::endl
945           « 1 « ::std::setw(30) « "!  NMEDIA" « ::std::endl;
946   }
947
948
949   inline void BellhopWoss::writeTransmitter() {
950     f_out « total_transmitters « ::std::setw(30) « "!  NUMBER OF SOURCES" « ::std::endl;
951     if (total_transmitters == 1) f_out « tx_coordz.getDepth() + tx_min_depth_offset « "  " « "/" «
    ::std::setw(30) « "!  SOURCE'S DEPTH" « ::std::endl;
952     else f_out « tx_coordz.getDepth() + tx_min_depth_offset « "  " « tx_coordz.getDepth() +
    tx_max_depth_offset
953             « "  " « "/" « ::std::setw(30) « "!  SOURCES' DEPTHS" « ::std::endl;
954   }
955
956
957   inline void BellhopWoss::writeReceiver() {
958     f_out « total_rx_depths « ::std::setw(30) « "!  NUMBER OF RX DEPTH(S)" « ::std::endl;
959     if (total_rx_depths == 1) f_out « rx_coordz.getDepth() + rx_min_depth_offset « "  " « "/" «
    ::std::setw(30) « "!  RX'S DEPTH" « ::std::endl;
960     else f_out « rx_coordz.getDepth() + rx_min_depth_offset « "  " « rx_coordz.getDepth() +
    rx_max_depth_offset « "  " « "/"
961             « ::std::setw(30) « "!  RX'S DEPTHS" « ::std::endl;
962
963     f_out « total_rx_ranges « ::std::setw(30) « "!NUMBER OF RX RANGE(S)" « ::std::endl;
964     if (total_rx_ranges == 1) f_out « ((total_great_circle_distance + rx_min_range_offset) / 1000.0) « "
    " « "/" « ::std::setw(30) « "!  RX'S RANGE" « ::std::endl;
965     else f_out « ((total_great_circle_distance + rx_min_range_offset) / 1000.0) « "  " «
    ((total_great_circle_distance + rx_max_range_offset)/1000.0) « "  " « "/"
966             « ::std::setw(30) « "!  RX'S RANGES" « ::std::endl;
967   }
968
969
970   inline void BellhopWoss::writeRayOptions() {
971     f_out « "\'"« bellhop_op_mode « beam_options;
972
973     if ( transducer->isValid() ) f_out « "*";
974
975     f_out « "\'" « ::std::setw(30) « "!  RAY OPTIONS" « ::std::endl
976           « total_rays « ::std::setw(30) « "!  NUMBER OF RAYS" « ::std::endl
977           « min_angle « "  " « max_angle « "  " « "/" « ::std::setw(30) « "!  START, END ANGLES" «
    ::std::endl;
978   }
979
980
981   inline void BellhopWoss::writeBox() {
982     f_out « "0.0" « " " « box_depth « " " « (box_range/1000.0)  « " ,  "
983           « "!  RAY-STEP , BOX DEPTH, BOX RANGE" « ::std::endl;
984   }
985
986
987   inline void BellhopWoss::writeBeamBlock() {
988     f_out « "\'MS\' 1.0 100.0 0," « ::std::endl « "3 5" « ::std::endl;
989   }
990
991
992 }
993
```

```
994
995 #endif /* WOSS_BELLHOP_H */
996
997
998
999
```

14.19 woss/res-reader.cpp File Reference

Provides the implementation of woss::ResReader class.

14.19.1 Detailed Description

Provides the implementation of woss::ResReader class.

Author

Federico Guerra

Provides the implementation of the woss::ResReader class

14.20 woss/res-reader.h File Reference

Provides the interface for woss::ResReader class.

Classes

- class woss::ResReader

  Abstract class for channel simulator result files processing.

14.20.1 Detailed Description

Provides the interface for woss::ResReader class.

Author

Federico Guerra

Provides the interface for the woss::ResReader class

14.21 res-reader.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_READER_DEFINITIONS_H
41 #define WOSS_RES_READER_DEFINITIONS_H
42
43
44 #include <string>
45
46
47 namespace woss {
48
49
50   class Woss;
51   class Pressure;
52   class TimeArr;
53
59   class ResReader {
60
61
62     public:
63
64
68     ResReader();
69
74     ResReader( const Woss* const woss );
75
76     virtual ~ResReader() { }
77
82     virtual bool initialize() = 0;
83
84
94     virtual Pressure* readAvgPressure( double tx_depth, double start_rx_depth, double start_rx_range,
      double end_rx_depth, double end_rx_range ) = 0;
95
103     virtual Pressure* readPressure( double tx_depth, double rx_depth, double rx_range ) const = 0;
104
105
113     virtual TimeArr* readTimeArr( double tx_depth, double rx_depth, double rx_range ) const = 0;
114
115
121     ResReader& setWossPtr( const Woss* const woss ) { woss_ptr = woss; return *this; }
122
127     const Woss* const getWossPtr() { return woss_ptr; }
128
129
135     ResReader& setFileName( const ::std::string& name ) { file_name = name; return *this; }
136
141     ::std::string getFileName() { return file_name; }
142
143
144     protected:
145
146
150     const Woss* woss_ptr;
151
152
```

```
156     ::std::string file_name;
157
158
159   };
160
161
162 }
163
164
165 #endif /* WOSS_RES_READER_DEFINITIONS_H */
166
167
```

14.22 woss/woss-controller.cpp File Reference

Provides the implementation for woss::WossController class.

14.22.1 Detailed Description

Provides the implementation for woss::WossController class.

Author

Federico Guerra

Provides the implementation for the woss::WossController class

14.23 woss/woss-controller.h File Reference

Provides the interface for woss::WossController class.

Classes

- class woss::WossController

  Class for managing all WOSS classes involved.

Typedefs

- typedef Singleton< WossController > woss::SWossController

  Singleton implementation of WossController class.

14.23.1 Detailed Description

Provides the interface for woss::WossController class.

Author

Federico Guerra

Provides the interface for the woss::WossController class

14.23.2 Typedef Documentation

14.23.2.1 SWossController `typedef Singleton< WossController > woss::SWossController`

Singleton implementation of WossController class.

Singleton implementation of WossController class

14.24 woss-controller.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_CONTROLLER_H
41 #define WOSS_CONTROLLER_H
42
43
44 #include <cassert>
45 #include <singleton-definitions.h>
46
47
48 namespace woss {
49
50
51   class WossDbCreator;
52   class WossCreator;
53   class WossDbManager;
54   class WossManager;
55   class TransducerHandler;
56
57
64   class WossController {
65
66
67     public:
68
69
73     WossController();
74
78     WossController( WossController& copy );
79
83     WossController& operator=( WossController& copy );
84
85
89     virtual ~WossController();
90
91
```

```
95     bool initialize();
96
97
98     WossController& setBathymetryDbCreator( WossDbCreator* const ptr ) { bathymetry_db_creator = ptr;
       return *this; }
99
100    WossController& setSedimentDbCreator( WossDbCreator* const ptr ) { sediment_db_creator = ptr; return
       *this; }
101
102    WossController& setSSPDbCreator( WossDbCreator* const ptr ) { ssp_db_creator = ptr; return *this; }
103
104    WossController& setPressureDbCreator( WossDbCreator* const ptr ) { pressure_result_db_creator = ptr;
       return *this; }
105
106    WossController& setTimeArrDbCreator( WossDbCreator* const ptr ) { timearr_result_db_creator = ptr;
       return *this; }
107
108    WossController& setWossCreator( WossCreator* const ptr ) { woss_creator = ptr; return *this; }
109
110    WossController& setWossDbManager( WossDbManager* const ptr ) { woss_db_manager = ptr; return *this;
       }
111
112    WossController& setWossManager( WossManager* const ptr ) { woss_manager = ptr; return *this; }
113
114    WossController& setTransducerHandler( TransducerHandler* const ptr ) { transducer_handler = ptr;
       return *this; }
115
116    void setDebug( bool flag) { debug = flag;}
117
118    const WossDbCreator* const getBathymetryDbCreator()const { assert(initialized); return
       bathymetry_db_creator; }
119
120    const WossDbCreator* const getSedimentDbCreator()const { assert(initialized); return
       sediment_db_creator; }
121
122    const WossDbCreator* const getSSPDbCreator()const { assert(initialized); return ssp_db_creator; }
123
124    const WossDbCreator* const getPressureDbCreator()const { assert(initialized); return
       pressure_result_db_creator; }
125
126    const WossDbCreator* const getTimeArrDbCreator()const { assert(initialized); return
       timearr_result_db_creator; }
127
128    const WossCreator* const getWossCreator()const { assert(initialized); return woss_creator; }
129
130    const WossDbManager* const getWossDbManager()const { assert(initialized); return woss_db_manager; }
131
132    WossManager* const getWossManager()const { assert(initialized); return woss_manager; }
133
134    TransducerHandler* const getTransducerHandler()const { assert(initialized); return
       transducer_handler; }
135
136    bool getDebug()const { return debug; }
137
138    protected:
139
140
144    double debug;
145
149    bool initialized;
150
151
152    WossDbCreator* bathymetry_db_creator;
153
154    WossDbCreator* sediment_db_creator;
155
156    WossDbCreator* ssp_db_creator;
157
158    WossDbCreator* pressure_result_db_creator;
159
160    WossDbCreator* timearr_result_db_creator;
161
162    WossCreator* woss_creator;
163
164    WossDbManager* woss_db_manager;
165
166    WossManager* woss_manager;
167
168    TransducerHandler* transducer_handler;
169
170
171  };
172
173
179  typedef Singleton< WossController > SWossController;
180
181
```

```
182 }
183
184
185 #endif /* WOSS_CONTROLLER_H */
186
```

14.25 woss/woss-creator-container.cpp File Reference

Provides the implementation of WossCreator< CustomTransducer> class.

14.25.1 Detailed Description

Provides the implementation of WossCreator< CustomTransducer> class.

Author

Federico Guerra

Provides the implementation of fully specialized WossCreator< CustomTransducer> class

14.26 woss/woss-creator-container.h File Reference

Provides the interface for woss::WossCreatorContainer class.

Classes

- struct woss::CustomTransducer

  Initial set up of a transducer.
- class woss::WossCreatorContainer< Data >

  Class that stores WossCreator parameters.
- class woss::WossCreatorContainer< Data * >

  Partial specialization for pointers.
- class woss::WossCreatorContainer< CustomTransducer >

  Full specialization for woss::CustomTransducer.

14.26.1 Detailed Description

Provides the interface for woss::WossCreatorContainer class.

Author

Federico Guerra

Provides the interface for woss::WossCreatorContainer class

14.27 woss-creator-container.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_CREATOR_CONTAINER_DEFINITIONS_H
41 #define WOSS_CREATOR_CONTAINER_DEFINITIONS_H
42
43
44 #include <map>
45 #include <location-definitions.h>
46 #include <iostream>
47
48
49 namespace woss {
50
57   struct CustomTransducer {
58
59
67     CustomTransducer( const ::std::string& name = "", double bearing = 0.0, double vert_rot = 0.0, double
      horiz_rot = 0.0, double mult = 1.0, double add = 0.0 )
68       :  type(name), initial_bearing(bearing), initial_vert_rotation(vert_rot),
      initial_horiz_rotation(horiz_rot), multiply_costant(mult), add_costant(add) { }
69
70
71     friend std::ostream& operator«( std::ostream& os, const CustomTransducer& instance ) {
72       os « "type name = " « instance.type «"; initial bearing = " « instance.initial_bearing
73          « "; initial vertical rotation = " « instance.initial_vert_rotation
74          « "; initial horizontal rotation = " « instance.initial_horiz_rotation
75          « "; mult costant = " « instance.multiply_costant « "; add costant = "
76          « instance.add_costant;
77       return os;
78     }
79
80
84     ::std::string type;
85
86
90     double initial_bearing;
91
95     double initial_vert_rotation;
96
100     double initial_horiz_rotation;
101
105     double multiply_costant;
106
110     double add_costant;
111
112
113   };
114
115
116
123   template< typename Data >
124   class WossCreatorContainer {
125
126
127     public:
128
```

```
129
133     static Location* const ALL_LOCATIONS;
134
138     static const CoordZ ALL_COORDZ;
139
140
144     WossCreatorContainer();
145
149     ~WossCreatorContainer();
150
151
156     bool isEmpty() const;
157
162     int size() const;
163
164
172     bool insert( const Data& data, Location* const tx, Location* const rx );
173
181     bool insert( const Data& data, const CoordZ& tx, const CoordZ& rx );
182
183
190     Data get( Location* const tx, Location* const rx  ) const;
191
198     Data get( const CoordZ& tx, const CoordZ& rx ) const;
199
205     Data& accessAllLocations();
206
207
213     void erase( Location* const tx, Location* const rx );
214
220     void erase( const CoordZ& tx, const CoordZ& rx );
221
222
230     void replace( const Data& data, Location* const tx, Location* const rx );
231
239     void replace( const Data& data, const CoordZ& tx, const CoordZ& rx );
240
241
245     void clear();
246
247
252     void setDebug( bool flag ) { debug = flag; }
253
258     bool isUsingDebug()const { return debug; }
259
260
261     protected:
262
263
267     typedef ::std::map< Location*, Data > InnerContainer;
268     typedef typename InnerContainer::iterator ICIter;
269     typedef typename InnerContainer::reverse_iterator ICRIter;
270     typedef typename InnerContainer::const_iterator ICCIter;
271     typedef typename InnerContainer::const_reverse_iterator ICCRIter;
272
273
277     typedef ::std::map< Location*, InnerContainer > DataContainer;
278     typedef typename DataContainer::iterator DCIter;
279     typedef typename DataContainer::const_iterator DCCIter;
280     typedef typename DataContainer::reverse_iterator DCRIter;
281     typedef typename DataContainer::const_reverse_iterator DCRCIter;
282
283
290     DCIter find( const CoordZ& coordinates );
291
299     ICIter find( const CoordZ& coordinates, const DCIter& iter );
300
301
307     Location* createLocation( const CoordZ& coordinates );
308
309
313     DataContainer data_container;
314
315
319     bool debug;
320
321
322   };
323
324
325   template< typename Data >
326   Location* const WossCreatorContainer< Data >::ALL_LOCATIONS = NULL;
327
328   template< typename Data >
329   const CoordZ WossCreatorContainer< Data >::ALL_COORDZ = CoordZ();
330
331
```

```
332   template< typename Data >
333   WossCreatorContainer< Data >::WossCreatorContainer()
334   :  data_container(),
335     debug(false)
336   {
337   }
338
339
340   template< typename Data >
341   WossCreatorContainer< Data >::~WossCreatorContainer() {
342     clear();
343   }
344
345
346   template< typename Data >
347   inline bool WossCreatorContainer< Data >::isEmpty()const {
348     return data_container.empty();
349   }
350
351
352   template< typename Data >
353   inline int WossCreatorContainer< Data >::size()const {
354     return data_container.size();
355   }
356
357
358   template< typename Data >
359   inline Location* WossCreatorContainer< Data >::createLocation( const CoordZ& coordinates ) {
360     if ( coordinates == ALL_COORDZ ) return NULL;
361     return new Location( coordinates );
362   }
363
364
365   template< typename Data >
366   typename WossCreatorContainer< Data >::DCIter WossCreatorContainer< Data >::find( const CoordZ&
      coordinates ) {
367     DCIter it_all_loc = data_container.end();
368
369     for ( DCIter it = data_container.begin(); it != data_container.end(); it++ ) {
370       if ( it->first == ALL_LOCATIONS ) {
371         it_all_loc = it;
372         if ( coordinates == ALL_COORDZ ) break;
373         else continue;
374       }
375       if ( it->first->isEquivalentTo( coordinates ) ) {
376         if (debug) ::std::cout « "WossCreatorContainer::find() tx coordinates found = "
377                                « coordinates « ::std::endl;
378         return it;
379       }
380     }
381
382     if ( coordinates == ALL_COORDZ && it_all_loc != data_container.end() ) {
383       if (debug) ::std::cout « "WossCreatorContainer::find() tx ALL_COORDZ found" « ::std::endl;
384       return it_all_loc;
385     }
386
387     if (debug && coordinates != ALL_COORDZ ) ::std::cout « "WossCreatorContainer::find() tx coordinates
      not found = "
388                                                          « coordinates « ::std::endl;
389
390     if (debug && coordinates == ALL_COORDZ ) ::std::cout « "WossCreatorContainer::find() tx ALL_COORDZ
      not found" « ::std::endl;
391     return data_container.end();
392   }
393
394
395   template< typename Data >
396   typename WossCreatorContainer< Data >::ICIter WossCreatorContainer< Data >::find( const CoordZ&
      coordinates, const DCIter& iter ) {
397     ICIter it_all_loc = iter->second.end();
398
399     for ( ICIter it = iter->second.begin(); it != iter->second.end(); it++ ) {
400       if ( it->first == ALL_LOCATIONS ) {
401         it_all_loc = it;
402         if ( coordinates == ALL_COORDZ ) break;
403         else continue;
404       }
405       if ( it->first->isEquivalentTo( coordinates ) ) {
406         if (debug) ::std::cout « "WossCreatorContainer::find() rx coordinates found = "
407                                « coordinates « ::std::endl;
408         return it;
409       }
410     }
411
412     if ( coordinates == ALL_COORDZ && it_all_loc != iter->second.end() ) {
413       if (debug) ::std::cout « "WossCreatorContainer::find() rx ALL_COORDZ found" « ::std::endl;
414       return it_all_loc;
```

```
415     }
416
417     if (debug && coordinates != ALL_COORDZ ) ::std::cout « "WossCreatorContainer::find() rx coordinates
    not found = "
418                                                          « coordinates « ::std::endl;
419
420     if (debug && coordinates == ALL_COORDZ ) ::std::cout « "WossCreatorContainer::find() rx ALL_COORDZ
    not found" « ::std::endl;
421     return iter->second.end();
422   }
423
424
425   template< typename Data >
426   inline bool WossCreatorContainer< Data >::insert( const Data& data, Location* const tx, Location*
    const rx ) {
427     DCIter it = data_container.find(tx);
428     if ( it == data_container.end() ) {
429       data_container[tx][rx] = data;
430       return true;
431     }
432     ICIter it2 = it->second.find(rx);
433     if ( it2 == it->second.end() ) {
434       (it->second)[rx] = data;
435       return true;
436     }
437     return false;
438   }
439
440
441   template< typename Data >
442   inline bool WossCreatorContainer< Data >::insert( const Data& data, const CoordZ& tx, const CoordZ& rx
    ) {
443     DCIter it = find( tx );
444     if ( it == data_container.end() ) {
445       data_container[ createLocation(tx) ][ createLocation(rx) ] = data;
446       return true;
447     }
448     ICIter it2 = find( rx, it );
449     if ( it2 == it->second.end() ) {
450       (it->second)[ createLocation(rx) ] = data;
451       return true;
452     }
453     return false;
454   }
455
456
457   template< typename Data >
458   inline Data WossCreatorContainer< Data >::get( Location* const tx, Location* const rx )const {
459     DCIter it = const_cast< typename WossCreatorContainer< Data >::DataContainer&
    >(data_container).find( tx );
460
461     if ( tx != ALL_LOCATIONS ) {
462       if ( it == data_container.end() ) {
463         if ( debug ) ::std::cout « "WossCreatorContainer::get() no tx location found = " « *tx
464                                  « "; trying ALL_LOCATIONS"« ::std::endl;
465
466         it = const_cast< typename WossCreatorContainer< Data >::DataContainer& >(data_container).find(
    ALL_LOCATIONS );
467       }
468
469       if ( debug ) ::std::cout « "WossCreatorContainer::get() tx location found = " « *tx « ::std::endl;
470
471     }
472
473     if ( it != data_container.end() ) {
474       ICIter it2 = it->second.find( rx );
475
476       if ( rx != ALL_LOCATIONS ) {
477
478         if ( it2 == it->second.end() ) {
479           if ( debug ) ::std::cout « "WossCreatorContainer::get() no rx location found = " « *rx
480                                    « "; trying ALL_LOCATIONS"« ::std::endl;
481
482           it2 = it->second.find( ALL_LOCATIONS );
483         }
484
485         if ( debug ) ::std::cout « "WossCreatorContainer::get() rx location found = " « *rx «
    ::std::endl;
486
487       }
488
489       if ( it2 != it->second.end() ) return it2->second;
490     }
491
492     ::std::cerr « "WARNING: WossCreatorContainer::get() no tx nor rx location found, returning default
    constructor!!" « ::std::endl;
493
```

```
494     return Data();
495   }
496
497
498   template< typename Data >
499   inline Data& WossCreatorContainer< Data >::accessAllLocations() {
500     return data_container[ALL_LOCATIONS][ALL_LOCATIONS];
501   }
502
503
504   template< typename Data >
505   inline Data WossCreatorContainer< Data >::get( const CoordZ& tx, const CoordZ& rx )const {
506     DCIter it = const_cast< WossCreatorContainer< Data >& >(*this).find( tx );
507
508     if ( tx != ALL_COORDZ ) {
509       if ( it == data_container.end() ) it = const_cast< WossCreatorContainer< Data >& >(*this).find(
    ALL_COORDZ );
510     }
511
512     if ( it != data_container.end() ) {
513       ICIter it2 = const_cast< WossCreatorContainer< Data >& >(*this).find( rx, it );
514
515       if ( rx != ALL_COORDZ ) {
516         if ( it2 == it->second.end() ) it2 = const_cast< WossCreatorContainer< Data >& >(*this).find(
    ALL_COORDZ, it );
517       }
518
519       if ( it2 != it->second.end() ) {
520         if ( debug ) ::std::cout « "WossCreatorContainer::get() value found = " « it2->second «
    ::std::endl;
521
522         return it2->second;
523       }
524     }
525     ::std::cerr « "WARNING: WossCreatorContainer::get() no tx nor rx coordinates found, returning
    default constructor!!" « ::std::endl;
526
527     return Data();
528   }
529
530
531   template< typename Data >
532   inline void WossCreatorContainer< Data >::erase( Location* const tx, Location* const rx ) {
533     DCIter it = data_container.find(tx);
534     if ( it != data_container.end() ) it->second.erase( rx );
535     if ( it->second.empty() ) data_container.erase(it);
536   }
537
538
539   template< typename Data >
540   inline void WossCreatorContainer< Data >::erase( const CoordZ& tx, const CoordZ& rx ) {
541     for ( DCIter it = data_container.begin(); it != data_container.end();  ) {
542
543       if ( it->first->isEquivalentTo( tx ) ) {
544
545         for ( ICIter it2 = it->second.begin(); it2 != it->second.end(); ) {
546           if ( it2->first->isEquivalentTo( rx ) ) it->second.erase(it2++);
547           else ++it2;
548         }
549
550       }
551
552       if ( it->second.empty() ) data_container.erase(it++);
553       else ++it;
554     }
555   }
556
557
558   template< typename Data >
559   inline void WossCreatorContainer< Data >::replace( const Data& data, Location* const tx, Location*
    const rx ) {
560     data_container[tx][rx] = data;
561   }
562
563
564   template< typename Data >
565   inline void WossCreatorContainer< Data >::replace( const Data& data, const CoordZ& tx, const CoordZ&
    rx ) {
566     data_container[ createLocation(tx) ][ createLocation(rx) ] = data;
567   }
568
569
570   template< typename Data >
571   inline void WossCreatorContainer< Data >::clear() {
572 //     for( DCIter it = data_container.begin(); it != data_container.end(); it++ ) {
573
574 //       for( ICIter it2 = it->second.begin(); it2 != it->second.end(); it2++ ) {
```

```
575 //
576 //       }
577
578 //       it->second.clear();
579 //     }
580     data_container.clear();
581   }
582
583
585
592    template< typename Data >
593   class WossCreatorContainer< Data* > {
594
595
596     public:
597
598
599     static Location* const ALL_LOCATIONS;
600
601     static const CoordZ ALL_COORDZ;
602
603
604     WossCreatorContainer();
605
606     ~WossCreatorContainer();
607
608
609     bool isEmpty() const;
610
611     int size() const;
612
613
622     bool insert( Data* data, Location* const tx, Location* const rx );
623
632     bool insert( Data* data, const CoordZ& tx, const CoordZ& rx );
633
634
641     Data* get( Location* const tx, Location* const rx ) const;
642
649     Data* get( const CoordZ& tx, const CoordZ& rx ) const;
650
651
658     Data*& accessAllLocations();
659
660
661     void erase( Location* const tx, Location* const rx );
662
663     void erase( const CoordZ& tx, const CoordZ& rx );
664
665
673     void replace( Data* const data, Location* const tx, Location* const rx );
674
682     void replace( Data* const data, const CoordZ& tx, const CoordZ& rx );
683
684
685     void clear();
686
687
688     void setDebug( bool flag ) { debug = flag; }
689
690     bool isUsingDebug()const { return debug; }
691
692
693     protected:
694
695
696     typedef ::std::map< Location*, Data* > InnerContainer;
697     typedef typename InnerContainer::iterator ICIter;
698     typedef typename InnerContainer::reverse_iterator ICRIter;
699     typedef typename InnerContainer::const_iterator ICCIter;
700     typedef typename InnerContainer::const_reverse_iterator ICCRIter;
701
702
703     typedef ::std::map< Location*, InnerContainer > DataContainer;
704
705     typedef typename DataContainer::iterator DCIter;
706     typedef typename DataContainer::const_iterator DCCIter;
707     typedef typename DataContainer::reverse_iterator DCRIter;
708     typedef typename DataContainer::const_reverse_iterator DCRCIter;
709
710
711     DCIter find( const CoordZ& coordinates );
712
713     ICIter find( const CoordZ& coordinates, const DCIter& iter );
714
715
716     Location* createLocation( const CoordZ& coordinates );
```

```
717
718
719     DataContainer data_container;
720
721
722     bool debug;
723
724
725   };
726
727
728   template< typename Data >
729   Location* const WossCreatorContainer< Data* >::ALL_LOCATIONS = NULL;
730
731   template< typename Data >
732   const CoordZ WossCreatorContainer< Data* >::ALL_COORDZ = CoordZ();
733
734
735   template< typename Data >
736   WossCreatorContainer< Data* >::WossCreatorContainer()
737   :  data_container(),
738     debug(false)
739   {
740   }
741
742
743   template< typename Data >
744   WossCreatorContainer< Data* >::~WossCreatorContainer() {
745     clear();
746   }
747
748
749   template< typename Data >
750   inline bool WossCreatorContainer< Data* >::isEmpty()const {
751     return data_container.empty();
752   }
753
754
755   template< typename Data >
756   inline int WossCreatorContainer< Data* >::size()const {
757     return data_container.size();
758   }
759
760
761   template< typename Data >
762   inline Location* WossCreatorContainer< Data* >::createLocation( const CoordZ& coordinates ) {
763     if ( coordinates == ALL_COORDZ ) return NULL;
764     return new Location( coordinates );
765   }
766
767
768   template< typename Data >
769   typename WossCreatorContainer< Data* >::DCIter WossCreatorContainer< Data* >::find( const CoordZ&
      coordinates ) {
770     DCIter it_all_loc = data_container.end();
771
772     for ( DCIter it = data_container.begin(); it != data_container.end(); it++ ) {
773       if ( it->first == ALL_LOCATIONS ) {
774         it_all_loc = it;
775         if ( coordinates == ALL_COORDZ ) break;
776         else continue;
777       }
778       if ( it->first->isEquivalentTo( coordinates ) ) {
779         if (debug) ::std::cout « "WossCreatorContainer*::find() tx coordinates found = "
780                                « coordinates « ::std::endl;
781         return it;
782       }
783     }
784
785     if ( coordinates == ALL_COORDZ && it_all_loc != data_container.end() ) {
786       if (debug) ::std::cout « "WossCreatorContainer*::find() tx ALL_COORDZ found" « ::std::endl;
787       return it_all_loc;
788     }
789
790     if (debug && coordinates != ALL_COORDZ ) ::std::cout « "WossCreatorContainer*::find() tx coordinates
      not found = "
791                                                          « coordinates « ::std::endl;
792
793     if (debug && coordinates == ALL_COORDZ ) ::std::cout « "WossCreatorContainer*::find() tx ALL_COORDZ
      not found" « ::std::endl;
794     return data_container.end();
795   }
796
797
798   template< typename Data >
799   typename WossCreatorContainer< Data* >::ICIter WossCreatorContainer< Data* >::find( const CoordZ&
      coordinates, const DCIter& iter ) {
```

```
800     ICIter it_all_loc = iter->second.end();
801
802     for ( ICIter it = iter->second.begin(); it != iter->second.end(); it++ ) {
803       if ( it->first == ALL_LOCATIONS ) {
804         it_all_loc = it;
805         if ( coordinates == ALL_COORDZ ) break;
806         else continue;
807       }
808       if ( it->first->isEquivalentTo( coordinates ) ) {
809         if (debug) ::std::cout « "WossCreatorContainer*::find() rx coordinates found = "
810                                « coordinates « ::std::endl;
811         return it;
812       }
813     }
814
815     if ( coordinates == ALL_COORDZ && it_all_loc != iter->second.end() ) {
816       if (debug) ::std::cout « "WossCreatorContainer*::find() rx ALL_COORDZ found" « ::std::endl;
817       return it_all_loc;
818     }
819
820     if (debug && coordinates != ALL_COORDZ ) ::std::cout « "WossCreatorContainer*::find() rx coordinates
    not found = "
821                                                          « coordinates « ::std::endl;
822
823     if (debug && coordinates == ALL_COORDZ ) ::std::cout « "WossCreatorContainer*::find() rx ALL_COORDZ
    not found" « ::std::endl;
824     return iter->second.end();
825   }
826
827
828   template< typename Data >
829   inline bool WossCreatorContainer< Data* >::insert( Data* data, Location* const tx, Location* const rx
    ) {
830     DCIter it = data_container.find(tx);
831     if ( it == data_container.end() ) {
832       data_container[tx][rx] = data;
833       return true;
834     }
835     ICIter it2 = it->second.find(rx);
836     if ( it2 == it->second.end() ) {
837       (it->second)[rx] = data;
838       return true;
839     }
840
841     delete data;
842     data = NULL;
843     return false;
844   }
845
846
847   template< typename Data >
848   inline bool WossCreatorContainer< Data* >::insert( Data* data, const CoordZ& tx, const CoordZ& rx ) {
849     DCIter it = find( tx );
850     if ( it == data_container.end() ) {
851       data_container[ createLocation(tx) ][ createLocation(rx) ] = data;
852       return true;
853     }
854     ICIter it2 = find( rx, it );
855     if ( it2 == it->second.end() ) {
856       (it->second)[ createLocation(rx) ] = data;
857       return true;
858     }
859
860     delete data;
861     data = NULL;
862     return false;
863   }
864
865
866   template< typename Data >
867   inline Data* WossCreatorContainer< Data* >::get( Location* const tx, Location* const rx )const {
868     DCIter it = const_cast< typename WossCreatorContainer< Data* >::DataContainer&
    >(data_container).find( tx );
869
870     if ( tx != ALL_LOCATIONS ) {
871       if ( it == data_container.end() ) {
872         if ( debug ) ::std::cout « "WossCreatorContainer*::get() no tx location found = " « *tx
873                                  « "; trying ALL_LOCATIONS"« ::std::endl;
874
875         it = const_cast< typename WossCreatorContainer< Data* >::DataContainer& >(data_container).find(
    ALL_LOCATIONS );
876       }
877
878       if ( debug ) ::std::cout « "WossCreatorContainer*::get() tx location found = " « *tx «
    ::std::endl;
879
880     }
```

```
881
882     if ( it != data_container.end() ) {
883       ICIter it2 = it->second.find( rx );
884
885       if ( rx != ALL_LOCATIONS ) {
886
887         if ( it2 == it->second.end() ) {
888           if ( debug ) ::std::cout « "WossCreatorContainer*::get() no rx location found = " « *rx
889                                    « "; trying ALL_LOCATIONS"« ::std::endl;
890
891           it2 = it->second.find( ALL_LOCATIONS );
892         }
893
894         if ( debug ) ::std::cout « "WossCreatorContainer*::get() rx location found = " « *rx «
    ::std::endl;
895       }
896
897       if ( it2 != it->second.end() ) return it2->second->clone();
898     }
899
900     if ( debug ) ::std::cerr « "WARNING: WossCreatorContainer*::get() no tx nor rx location found,
    returning default constructor!!" « ::std::endl;
901
902     return new Data();
903   }
904
905
906   template< typename Data >
907   inline Data*& WossCreatorContainer< Data* >::accessAllLocations() {
908     return data_container[ALL_LOCATIONS][ALL_LOCATIONS];
909   }
910
911
912   template< typename Data >
913   inline Data* WossCreatorContainer< Data* >::get( const CoordZ& tx, const CoordZ& rx )const {
914     DCIter it = const_cast< WossCreatorContainer< Data* >& >(*this).find( tx );
915
916     if ( tx != ALL_COORDZ ) {
917       if ( it == data_container.end() ) it = const_cast< WossCreatorContainer< Data* >& >(*this).find(
    ALL_COORDZ );
918     }
919
920     if ( it != data_container.end() ) {
921       ICIter it2 = const_cast< WossCreatorContainer< Data* >& >(*this).find( rx, it );
922
923       if ( rx != ALL_COORDZ ) {
924         if ( it2 == it->second.end() ) it2 = const_cast< WossCreatorContainer< Data* >& >(*this).find(
    ALL_COORDZ, it );
925       }
926
927       if ( it2 != it->second.end() ) return it2->second->clone();
928     }
929     if ( debug ) ::std::cerr « "WARNING: WossCreatorContainer*::get() no tx nor rx coordinates found,
    returning default constructor!!" « ::std::endl;
930
931     return new Data();
932   }
933
934
935   template< typename Data >
936   inline void WossCreatorContainer< Data* >::erase( Location* const tx, Location* const rx ) {
937     DCIter it = data_container.find(tx);
938     if ( it != data_container.end() ) {
939
940       ICIter it2 = it->second.find( rx );
941
942       if ( it2 != it->second.end() ) {
943         delete it2->second;
944 //         delete it2->first;
945         it->second.erase( it2 );
946       }
947
948     }
949     if ( it->second.empty() ) {
950 //       delete it->first;
951       data_container.erase(it);
952     }
953   }
954
955
956   template< typename Data >
957   inline void WossCreatorContainer< Data* >::erase( const CoordZ& tx, const CoordZ& rx ) {
958     for ( DCIter it = data_container.begin(); it != data_container.end();  ) {
959
960       if ( it->first->isEquivalentTo( tx ) ) {
961
962         for ( ICIter it2 = it->second.begin(); it2 != it->second.end(); ) {
```

```
963           if ( it2->first->isEquivalentTo( rx ) ) {
964             delete it2->second;
965 //              delete it2->first;
966             it->second.erase(it2++);
967           }
968           else ++it2;
969         }
970
971       }
972
973       if ( it->second.empty() ) {
974 //          delete it->first;
975         data_container.erase(it++);
976       }
977       else ++it;
978     }
979   }
980
981
982   template< typename Data >
983   inline void WossCreatorContainer< Data* >::replace( Data* const data, Location* const tx, Location*
      const rx ) {
984     if ( data_container[tx][rx] != NULL ) delete data_container[tx][rx];
985     data_container[tx][rx] = data;
986   }
987
988
989   template< typename Data >
990   inline void WossCreatorContainer< Data* >::replace( Data* const data, const CoordZ& tx, const CoordZ&
      rx ) {
991     Location* tx_loc = createLocation(tx);
992     Location* rx_loc = createLocation(rx);
993
994     if ( data_container[tx_loc][rx_loc] != NULL ) delete data_container[tx_loc][rx_loc];
995     data_container[tx_loc][rx_loc] = data;
996   }
997
998
999   template< typename Data >
1000   inline void WossCreatorContainer< Data* >::clear() {
1001     for( DCIter it = data_container.begin(); it != data_container.end(); it++ ) {
1002
1003       for( ICIter it2 = it->second.begin(); it2 != it->second.end(); it2++ ) {
1004         delete it2->second;
1005       }
1006
1007       it->second.clear();
1008 //        delete it->first;
1009     }
1010     data_container.clear();
1011   }
1012
1013
1015
1016
1022   template<>
1023   class WossCreatorContainer< CustomTransducer > {
1024
1025
1026     public:
1027
1028
1029     static Location* const ALL_LOCATIONS;
1030
1031     static const CoordZ ALL_COORDZ;
1032
1033
1034     WossCreatorContainer();
1035
1036     ~WossCreatorContainer();
1037
1038
1039     bool isEmpty() const;
1040
1041     int size() const;
1042
1043
1052     bool insert( const CustomTransducer& data, Location* const tx, Location* const rx );
1053
1062     bool insert( const CustomTransducer& data, const CoordZ& tx, const CoordZ& rx );
1063
1064
1071     CustomTransducer get( Location* const tx, Location* const rx ) const;
1072
1079     CustomTransducer get( const CoordZ& tx, const CoordZ& rx ) const;
1080
1081
```

```
1088     CustomTransducer& accessAllLocations();
1089
1090
1091     void erase( Location* const tx, Location* const rx );
1092
1093     void erase( const CoordZ& tx, const CoordZ& rx );
1094
1095
1103     void replace( const CustomTransducer& data, Location* const tx, Location* const rx );
1104
1112     void replace( const CustomTransducer& data, const CoordZ& tx, const CoordZ& rx );
1113
1114
1115     void clear();
1116
1117
1118     void setDebug( bool flag ) { debug = flag; }
1119
1120     bool isUsingDebug()const { return debug; }
1121
1122
1123     protected:
1124
1125
1126     typedef ::std::map< Location*, CustomTransducer > InnerContainer;
1127     typedef InnerContainer::iterator ICIter;
1128     typedef InnerContainer::reverse_iterator ICRIter;
1129     typedef InnerContainer::const_iterator ICCIter;
1130     typedef InnerContainer::const_reverse_iterator ICCRIter;
1131
1132
1133     typedef ::std::map< Location*, InnerContainer > DataContainer;
1134
1135     typedef DataContainer::iterator DCIter;
1136     typedef DataContainer::const_iterator DCCIter;
1137     typedef DataContainer::reverse_iterator DCRIter;
1138     typedef DataContainer::const_reverse_iterator DCRCIter;
1139
1140
1141     DCIter find( const CoordZ& coordinates );
1142
1143     ICIter find( const CoordZ& coordinates, const DCIter& iter );
1144
1145
1146     Location* createLocation( const CoordZ& coordinates );
1147
1148
1149     DataContainer data_container;
1150
1151
1152     bool debug;
1153
1154
1155   };
1156
1157
1158   inline bool WossCreatorContainer< CustomTransducer >::isEmpty()const {
1159     return data_container.empty();
1160   }
1161
1162
1163   inline int WossCreatorContainer< CustomTransducer >::size()const {
1164     return data_container.size();
1165   }
1166
1167
1168   inline Location* WossCreatorContainer< CustomTransducer >::createLocation( const CoordZ& coordinates
     ) {
1169     if ( coordinates == ALL_COORDZ ) return NULL;
1170     return new Location( coordinates );
1171   }
1172
1173
1174   inline bool WossCreatorContainer< CustomTransducer >::insert( const CustomTransducer& data, Location*
     const tx, Location* const rx ) {
1175     DCIter it = data_container.find(tx);
1176     if ( it == data_container.end() ) {
1177       data_container[tx][rx] = data;
1178       return true;
1179     }
1180     ICIter it2 = it->second.find(rx);
1181     if ( it2 == it->second.end() ) {
1182       (it->second)[rx] = data;
1183       return true;
1184     }
1185     return false;
1186   }
```

```
1187
1188
1189   inline bool WossCreatorContainer< CustomTransducer >::insert( const CustomTransducer& data, const
       CoordZ& tx, const CoordZ& rx ) {
1190     DCIter it = find( tx );
1191     if ( it == data_container.end() ) {
1192       data_container[ createLocation(tx) ][ createLocation(rx) ] = data;
1193       return true;
1194     }
1195     ICIter it2 = find( rx, it );
1196     if ( it2 == it->second.end() ) {
1197       (it->second)[ createLocation(rx) ] = data;
1198       return true;
1199     }
1200     return false;
1201   }
1202
1203
1204   inline CustomTransducer& WossCreatorContainer< CustomTransducer >::accessAllLocations() {
1205     return data_container[ALL_LOCATIONS][ALL_LOCATIONS];
1206   }
1207
1208
1209   inline void WossCreatorContainer< CustomTransducer >::erase( Location* const tx, Location* const rx )
       {
1210     DCIter it = data_container.find(tx);
1211     if ( it != data_container.end() ) {
1212
1213       ICIter it2 = it->second.find( rx );
1214
1215       if ( it2 != it->second.end() ) {
1216         it->second.erase( it2 );
1217       }
1218
1219     }
1220     if ( it->second.empty() ) {
1221       data_container.erase(it);
1222     }
1223   }
1224
1225
1226   inline void WossCreatorContainer< CustomTransducer >::erase( const CoordZ& tx, const CoordZ& rx ) {
1227     for ( DCIter it = data_container.begin(); it != data_container.end();  ) {
1228
1229       if ( it->first->isEquivalentTo( tx ) ) {
1230
1231         for ( ICIter it2 = it->second.begin(); it2 != it->second.end(); ) {
1232           if ( it2->first->isEquivalentTo( rx ) ) {
1233             it->second.erase(it2++);
1234           }
1235           else ++it2;
1236         }
1237
1238       }
1239
1240       if ( it->second.empty() ) {
1241         data_container.erase(it++);
1242       }
1243       else ++it;
1244     }
1245   }
1246
1247
1248   inline void WossCreatorContainer< CustomTransducer >::replace( const CustomTransducer& data,
       Location* const tx, Location* const rx ) {
1249     data_container[tx][rx] = data;
1250   }
1251
1252
1253   inline void WossCreatorContainer< CustomTransducer >::replace( const CustomTransducer& data, const
       CoordZ& tx, const CoordZ& rx ) {
1254     Location* tx_loc = createLocation(tx);
1255     Location* rx_loc = createLocation(rx);
1256
1257     data_container[tx_loc][rx_loc] = data;
1258   }
1259
1260
1261   inline void WossCreatorContainer< CustomTransducer >::clear() {
1262 //     for( DCIter it = data_container.begin(); it != data_container.end(); it++ ) {
1263 //       for( ICIter it2 = it->second.begin(); it2 != it->second.end(); it2++ ) {
1264 //
1265 //       }
1266 //     }
1267     data_container.clear();
1268   }
1269
```

```
1270 }
1271
1272
1273 #endif // WOSS_CREATOR_CONTAINER_DEFINITIONS_H
```

14.28 woss/woss-creator.cpp File Reference

Provides the implementation of woss::WossCreator class.

14.28.1 Detailed Description

Provides the implementation of woss::WossCreator class.

Author

Federico Guerra

Provides the implementation of woss::WossCreator class

14.29 woss/woss-creator.h File Reference

Provides the interface for woss::WossCreator class.

Classes

- class woss::WossCreator

  Abstract class that provides correctly initialized Woss objects.

14.29.1 Detailed Description

Provides the interface for woss::WossCreator class.

Author

Federico Guerra

Provides the interface for woss::WossCreator class

14.30 woss-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_CREATOR_DEFINITIONS_H
41 #define WOSS_CREATOR_DEFINITIONS_H
42
43
44 #define WOSS_CREATOR_MAX_FREQ_STEP (1.0e20)
45 #define WOSS_CREATOR_ALL_COORDZ CoordZ()
46
47
48 #include "woss.h"
49 #include "woss-creator-container.h"
50
51
52 namespace woss {
53
54
55
56   class WossDbManager;
57   class TransducerHandler;
58
59
65   class WossCreator {
66
67
68     protected:
69
70
74     typedef WossCreatorContainer< SimTime > CCSimTime;
75
79     typedef WossCreatorContainer< double > CCDouble;
80
84     typedef WossCreatorContainer< int > CCInt;
85
86
87     public:
88
89
93     WossCreator();
94
95     virtual ~WossCreator() { }
96
97
106     virtual Woss* const createWoss( const CoordZ& tx, const CoordZ& rx, double start_freq, double
      end_freq ) const = 0;
107
108
114     WossCreator& setWossDebug( bool flag ) { woss_debug = flag; return *this; }
115
121     WossCreator& setDebug( bool flag ) { debug = flag; updateDebugFlag(); return *this; }
122
127     bool usingDebug()const { return debug; }
128
133     bool usingWossDebug()const { return woss_debug; }
134
135
```

```
141     WossCreator& setWrkDirPath( const ::std::string& path ) { work_dir_path = path; return *this; }
142
148      WossCreator& setCleanWorkDir( bool flag ) { woss_clean_workdir = flag; return *this; }
149
154     ::std::string getWrkDirPath()const { return work_dir_path; }
155
156
164     WossCreator& setFrequencyStep( double f_step, const CoordZ& tx, const CoordZ& rx ) {
165       if ( f_step <= 0.0 ) f_step = WOSS_CREATOR_MAX_FREQ_STEP;
166       ccfrequency_step.replace(f_step, tx, rx);
167       return *this; }
168
176     WossCreator& setFrequencyStep( double f_step, Location* const tx = CCDouble::ALL_LOCATIONS,
    Location* const rx = CCDouble::ALL_LOCATIONS ) {
177       if ( f_step <= 0.0 ) f_step = WOSS_CREATOR_MAX_FREQ_STEP;
178       ccfrequency_step.replace(f_step, tx, rx);
179       return *this; }
180
187     double getFrequencyStep( const CoordZ& tx, const CoordZ& rx )const {
188       return ccfrequency_step.get(tx, rx); }
189
196     double getFrequencyStep( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS )const {
197       return ccfrequency_step.get(tx, rx); }
198
205     WossCreator& eraseFrequencyStep( const CoordZ& tx, const CoordZ& rx ) {
206       ccfrequency_step.erase(tx, rx); return *this; }
207
214     WossCreator& eraseFrequencyStep( Location* const tx = CCDouble::ALL_LOCATIONS, Location* const rx =
    CCDouble::ALL_LOCATIONS ) {
215       ccfrequency_step.erase(tx, rx); return *this; }
216
217
225     WossCreator& setEvolutionTimeQuantum( double value, const CoordZ& tx, const CoordZ& rx ) {
226       ccevolution_time_quantum.replace( value, tx, rx ); return *this; }
227
235     WossCreator& setEvolutionTimeQuantum( double value, Location* const tx = CCInt::ALL_LOCATIONS,
    Location* const rx = CCInt::ALL_LOCATIONS ) {
236       ccevolution_time_quantum.replace( value, tx, rx ); return *this; }
237
244     double getEvolutionTimeQuantum( const CoordZ& tx, const CoordZ& rx )const {
245       return ccevolution_time_quantum.get(tx, rx); }
246
253     double getEvolutionTimeQuantum( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS )const {
254       return ccevolution_time_quantum.get(tx, rx); }
255
262     WossCreator& eraseEvolutionTimeQuantum( const CoordZ& tx, const CoordZ& rx ) {
263       ccevolution_time_quantum.erase(tx, rx); return *this; }
264
271     WossCreator& eraseEvolutionTimeQuantum( Location* const tx = CCInt::ALL_LOCATIONS, Location* const
    rx = CCInt::ALL_LOCATIONS ) {
272       ccevolution_time_quantum.erase(tx, rx); return *this; }
273
274
282     WossCreator& setTotalRuns( int runs, const CoordZ& tx, const CoordZ& rx ) {
283       cctotal_runs.replace( runs, tx, rx ); return *this; }
284
292     WossCreator& setTotalRuns( int runs, Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
293       cctotal_runs.replace( runs, tx, rx ); return *this; }
294
301     int getTotalRuns( const CoordZ& tx, const CoordZ& rx )const {
302       return cctotal_runs.get(tx, rx); }
303
310     int getTotalRuns( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS )const {
311       return cctotal_runs.get(tx, rx); }
312
319     WossCreator& eraseTotalRuns( const CoordZ& tx, const CoordZ& rx ) {
320       cctotal_runs.erase( tx, rx ); return *this; }
321
328     WossCreator& eraseTotalRuns( Location* const tx = CCInt::ALL_LOCATIONS, Location* const rx =
    CCInt::ALL_LOCATIONS ) {
329       cctotal_runs.erase( tx, rx ); return *this; }
330
331
339     WossCreator& setSimTime( const SimTime& simtime, const CoordZ& tx, const CoordZ& rx ) {
340       ccsimtime_map.replace(simtime, tx, rx); return *this; }
341
349     WossCreator& setSimTime( const SimTime& simtime, Location* const tx = CCSimTime::ALL_LOCATIONS,
    Location* const rx = CCSimTime::ALL_LOCATIONS ) {
350       ccsimtime_map.replace(simtime, tx, rx); return *this; }
351
358     SimTime getSimTime( const CoordZ& tx, const CoordZ& rx )const {
359       return ccsimtime_map.get(tx, rx); }
360
```

```
367     SimTime getSimTime( Location* const tx = CCSimTime::ALL_LOCATIONS, Location* const rx =
    CCSimTime::ALL_LOCATIONS )const {
368       return ccsimtime_map.get(tx, rx); }
369
376     WossCreator& eraseSimTime( const CoordZ& tx, const CoordZ& rx ) { ccsimtime_map.erase(tx, rx);
    return *this; }
377
384     WossCreator& eraseSimTime( Location* const tx = CCSimTime::ALL_LOCATIONS, Location* const rx =
    CCSimTime::ALL_LOCATIONS ) {
385       ccsimtime_map.erase(tx, rx); return *this; }
386
387
393     WossCreator& setWossDbManager( const WossDbManager* const ptr ) { woss_db_manager = ptr; return
    *this; }
394
395
401     WossCreator& setTransducerHandler( const TransducerHandler* const ptr ) { transducer_handler = ptr;
    return *this; }
402
403
404     const Woss& getWossNotValid() const;
405
406
407     protected:
408
409
410     static const Woss* woss_not_valid;
411
412
413     const WossDbManager* woss_db_manager;
414
415
416     const TransducerHandler* transducer_handler;
417
418
422     ::std::string work_dir_path;
423
424
428     CCDouble ccevolution_time_quantum;
429
433     CCInt cctotal_runs;
434
438     CCDouble ccfrequency_step;
439
440
444     CCSimTime ccsimtime_map;
445
446
450     bool debug;
451
455     bool woss_debug;
456
460     bool woss_clean_workdir;
461
462
468     virtual bool initializeWoss( Woss* const woss_ptr ) const = 0;
469
470     virtual const Woss* createNotValidWoss() const = 0;
471
475     virtual void updateDebugFlag();
476
477   };
478
479
480 }
481
482
483 #endif /* WOSS_CREATOR_DEFINITIONS_H */
484
```

14.31 woss/woss-manager-simple.h File Reference

Provides the interface for woss::WossManagerSimple class.

Classes

- class woss::WossManagerSimple< WMResDb >

  simple template extension of WossManagerResDb or WossManagerResDbMT

14.31.1 Detailed Description

Provides the interface for woss::WossManagerSimple class.

Author

Federico Guerra

Provides the interface for the woss::WossManagerSimple class

14.32 woss-manager-simple.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_MANAGER_SIMPLE_DEFINITIONS_H
41 #define WOSS_MANAGER_SIMPLE_DEFINITIONS_H
42
43
44 #include <time-arrival-definitions.h>
45 #include <definitions-handler.h>
46 #include "woss-manager.h"
47
48
49 namespace woss {
50
51
60   template< typename WMResDb = WossManagerResDb >
61   class WossManagerSimple :  public WMResDb {
62
63
64     public:
65
66
70     WossManagerSimple();
71
72     virtual ~WossManagerSimple() { reset(); }
73
74
75     virtual WossManagerSimple& eraseActiveWoss( const CoordZ& tx, const CoordZ& rx, double
      start_frequency, double end_frequency );
76
81     virtual bool reset();
82
83
89     virtual bool timeEvolve( const Time& time_value );
90
91
96     static void setSpaceSampling( double radius ) { space_sampling = radius; }
```

```
97
102     static double getSpaceSampling() { return space_sampling; }
103
104
105     protected:
106
107
111     typedef typename ::std::map< CoordZ, Woss*, CoordComparator< WossManagerSimple, CoordZ > >
    WossCoordZMap;
112     typedef typename WossCoordZMap::iterator WCZIter;
113     typedef typename WossCoordZMap::reverse_iterator WCZRIter;
114
115
119     typedef typename ::std::map< CoordZ, WossCoordZMap, CoordComparator< WossManagerSimple, CoordZ > >
    WossContainer;
120     typedef typename WossContainer::iterator WCIter;
121     typedef typename WossContainer::reverse_iterator WCRIter;
122
123
128     static double space_sampling;
129
133     WossContainer woss_map;
134
135
144     virtual Woss* const getWoss( const CoordZ& tx, const CoordZ& rx, double start_frequency, double
    end_frequency );
145
146
147   };
148
149
150   template< typename WMResDb >
151   double WossManagerSimple< WMResDb >::space_sampling = 0.0;
152
153
154   template< typename WMResDb >
155   WossManagerSimple< WMResDb >::WossManagerSimple()
156   :  woss_map()
157   {
158
159
160   }
161
162
163   template< typename WMResDb >
164   bool WossManagerSimple< WMResDb >::reset() {
165     for (WCIter it1 = woss_map.begin(); it1 != woss_map.end(); it1++) {
166         for (WCZIter it2 = (it1->second).begin(); it2 != (it1->second).end(); it2++) {
167           delete it2->second;
168           it2->second = NULL;
169         }
170     }
171     woss_map.clear();
172     return true;
173   }
174
175
176   template< typename WMResDb >
177   bool WossManagerSimple< WMResDb >::timeEvolve( const Time& time_value ) {
178     for (WCIter it1 = woss_map.begin(); it1 != woss_map.end(); it1++) {
179         for (WCZIter it2 = (it1->second).begin(); it2 != (it1->second).end(); it2++) {
180           it2->second->timeEvolve(time_value);
181         }
182     }
183     return true;
184   }
185
186
187   template< typename WMResDb >
188   Woss* const WossManagerSimple< WMResDb >::getWoss( const CoordZ& tx_coordz, const CoordZ& rx_coordz,
    double start_frequency, double end_frequency ) {
189     if (WMResDb::debug) ::std::cout « "WossManagerSimple::getWoss() tx coords " « tx_coordz « "; rx
    coords "
190                                     « rx_coordz « "; start freq " « start_frequency « "; end freq "
191                                     « end_frequency « ::std::endl;
192
193     WCIter it1 = woss_map.find( tx_coordz );
194
195     if (it1 == woss_map.end() ) { // no tx CoordZ found
196
197       if (WMResDb::debug) ::std::cout « "WossManagerSimple::getWoss() no tx CoordZ found" « ::std::endl;
198
199       Woss* const curr_woss = this->woss_creator->createWoss( tx_coordz, rx_coordz, start_frequency,
    end_frequency );
200
201       woss_map[tx_coordz][rx_coordz] = curr_woss;
202       return( curr_woss );
```

```
203     }
204     else { // start CoordZ found
205       WCZIter it2 = (it1->second).find( rx_coordz );
206
207       if ( it2 == it1->second.end() ) { // no rx CoordZ foundZ
208
209         if (WMResDb::debug) ::std::cout << "WossManagerSimple::getWoss() no rx CoordZ found" <<
    ::std::endl;
210
211         Woss* const curr_woss = WMResDb::woss_creator->createWoss( tx_coordz, rx_coordz,
    start_frequency, end_frequency );
212
213         woss_map[tx_coordz][rx_coordz] = curr_woss;
214         return( curr_woss );
215       }
216       else return( it2->second );
217     }
218   }
219
220
221   template< typename WMResDb >
222   WossManagerSimple< WMResDb >& WossManagerSimple< WMResDb >::eraseActiveWoss( const CoordZ& tx_coordz,
    const CoordZ& rx_coordz, double start_frequency, double end_frequency ) {
223     if (WMResDb::debug) ::std::cout << "WossManagerSimple::eraseActiveWoss() tx coords " << tx_coordz << ";
    rx coords "
224                                     << rx_coordz << "; start freq " << start_frequency << "; end freq "
225                                     << end_frequency << ::std::endl;
226
227     WCIter it1 = woss_map.find( tx_coordz );
228
229     if (it1 == woss_map.end() ) return *this;
230     else { // start CoordZ found
231       WCZIter it2 = (it1->second).find( rx_coordz );
232
233       if ( it2 == it1->second.end() ) return *this;
234       else {
235         delete it2->second;
236         it1->second.erase(it2);
237         if ( it1->second.empty() ) woss_map.erase(it1);
238       }
239     }
240     return *this;
241   }
242
243
244 }
245
246
247
248 #endif /* WOSS_MANAGER_SIMPLE_DEFINITIONS_H */
249
250
```

14.33 woss/woss-manager.cpp File Reference

Provides the implementation of woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes.

14.33.1 Detailed Description

Provides the implementation of woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes.

Author

Federico Guerra

Provides the implementation of woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes

14.34 woss/woss-manager.h File Reference

Provides the interface for woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes.

Classes

- class woss::WossManager

  Abstract class that interfaces Pressure or TimeArr requests from user layer.
- class woss::WossManagerResDb

  Abstract class that implements WossManager. It adds computed results dbs control.
- class woss::WossManagerResDbMT

  Multi-threaded extension of WossManagerResDb.
- struct woss::WossManagerResDbMT::ThreadQuery
- struct woss::WossManagerResDbMT::ThreadParam
- struct woss::WossManagerResDbMT::ThreadCondSignal

Typedefs

- typedef ::std::pair< CoordZ, CoordZ > woss::CoordZPair
- typedef ::std::vector< CoordZPair > woss::CoordZPairVect
- typedef ::std::pair< double, double > woss::SimFreq
- typedef ::std::vector< SimFreq > woss::SimFreqVector
- typedef ::std::vector< Pressure * > woss::PressureVector
- typedef ::std::vector< TimeArr * > woss::TimeArrVector

Functions

- void * woss::WMSMTcreateThreadTimeArr (void *ptr)
- void * woss::WMSMTcreateThreadPressure (void *ptr)

14.34.1 Detailed Description

Provides the interface for woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes.

Author

Federico Guerra

Provides the interface for woss::WossManager, woss::WossManagerResDb and woss::WossManagerResDbMT classes

14.34.2 Typedef Documentation

14.34.2.1 CoordZPair `typedef ::std::pair< CoordZ, CoordZ > woss::CoordZPair`

A pair of CoordZ (tx, rx)

14.34.2.2 CoordZPairVect `typedef ::std::vector< CoordZPair > woss::CoordZPairVect`

A vector of CoordZPair

14.34.2.3 PressureVector `typedef ::std::vector< Pressure* > woss::PressureVector`

A vector of heap-created Pressure objects

14.34.2.4 SimFreq `typedef ::std::pair< double, double > woss::SimFreq`

A pair of frequency (start, end)

14.34.2.5 SimFreqVector `typedef ::std::vector< SimFreq > woss::SimFreqVector`

A vector of SimFreq

14.34.2.6 TimeArrVector `typedef ::std::vector< TimeArr* > woss::TimeArrVector`

A vector of heap-created TimeArr objects

14.34.3 Function Documentation

14.34.3.1 WMSMTcreateThreadPressure() `void * woss::WMSMTcreateThreadPressure ( void * ptr )`

Function used for Pressure thread creation

Parameters

ptr	void pointer

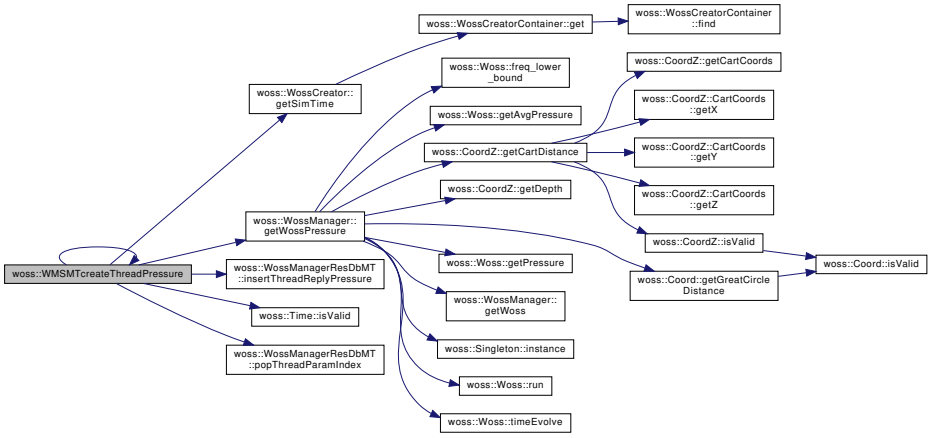
Returns

void pointer

References woss::WossManager::debug, woss::WossCreator::getSimTime(), woss::WossManager::getWossPressure(), woss::WossManagerResDbMT::insertThreadReplyPressure(), woss::Time::isValid(), woss::WossManagerResDbMT::popThreadPara and woss::WMSMTcreateThreadPressure().

Referenced by woss::WMSMTcreateThreadPressure().

Here is the call graph for this function:



14.34.3.2 WMSMTcreateThreadTimeArr() `void * woss::WMSMTcreateThreadTimeArr ( void * ptr )`

Function used for TimeArr thread creation

Parameters

ptr	void pointer

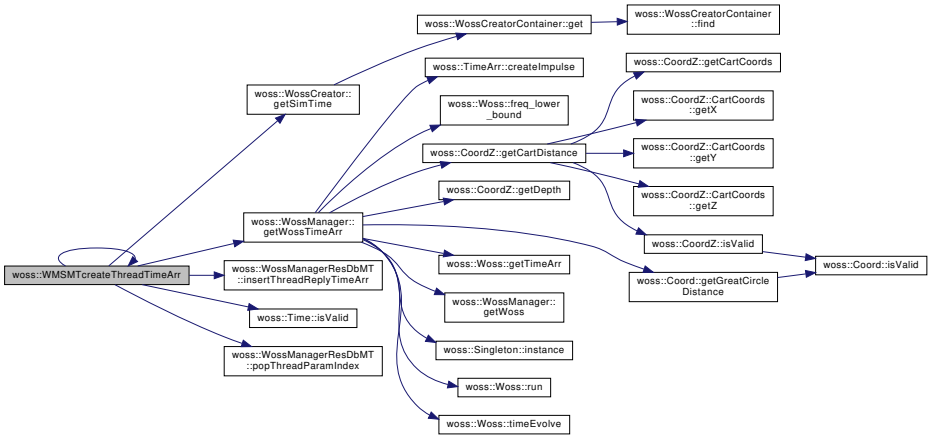
Returns

void pointer

References woss::WossManager::debug, woss::WossCreator::getSimTime(), woss::WossManager::getWossTimeArr(), woss::WossManagerResDbMT::insertThreadReplyTimeArr(), woss::Time::isValid(), woss::WossManagerResDbMT::popThreadParar and woss::WMSMTcreateThreadTimeArr().

Referenced by woss::WMSMTcreateThreadTimeArr().

Here is the call graph for this function:



14.35 woss-manager.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_MANAGER_DEFINITIONS_H
41 #define WOSS_MANAGER_DEFINITIONS_H
42
43
44 #include <definitions-handler.h>
45 #include <time-arrival-definitions.h>
46 #include "woss-creator.h"
47 #include <woss-db-manager.h>
48
49
50 namespace woss {
51
52
56   typedef ::std::pair< CoordZ, CoordZ > CoordZPair;
57
61   typedef ::std::vector< CoordZPair > CoordZPairVect;
62
63
67   typedef ::std::pair< double, double > SimFreq;
68
72   typedef ::std::vector< SimFreq > SimFreqVector;
73
74
78   typedef ::std::vector< Pressure* > PressureVector;
79
83   typedef ::std::vector< TimeArr* > TimeArrVector;
84
85
93   class WossManager {
94
95
96     public:
97
98
102     WossManager();
103
104     virtual ~WossManager();
105
106
115     virtual const Woss& getActiveWoss( const CoordZ& tx, const CoordZ& rx, double start_frequency,
      double end_frequency ) const;
116
125     virtual WossManager& eraseActiveWoss( const CoordZ& tx, const CoordZ& rx, double start_frequency,
      double end_frequency ) = 0;
126
127
137     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, const Time& time_value ) = 0;
138
148     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, double time_value = 0.0 );
149
158     virtual PressureVector getWossPressure( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, const Time& time_value );
```

```
159
168     virtual PressureVector getWossPressure( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, double time_value = 0.0 );
169
179     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, const Time& time_value ) = 0;
180
190     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, double time_value = 0.0 );
191
200     virtual TimeArrVector getWossTimeArr( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, const Time& time_value );
201
210     virtual TimeArrVector getWossTimeArr( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, double time_value = 0.0 );
211
212
217     virtual bool reset() = 0;
218
219
225     virtual bool timeEvolve( const Time& time_value ) = 0;
226
227
233     WossManager& setWossCreator( const WossCreator* const ptr ) { woss_creator = ptr; return *this; }
234
235     void setTimeEvolutionActiveFlag( bool flag ) { is_time_evolution_active = flag; }
236
237     void setDebugFlag( bool flag ) { debug = flag; }
238
239
240     const WossCreator* const getWossCreator() { return woss_creator; }
241
242     bool getTimeEvolutionActiveFlag() { return (bool)is_time_evolution_active; }
243
244     bool getDebugFlag() { return (bool)debug; }
245
246
247     protected:
248
249
250     static const Time NO_EVOLUTION_TIME;
251
252
256     const WossCreator* woss_creator;
257
258
262     bool debug;
263
264
265     bool is_time_evolution_active;
266
267
276     virtual Woss* const getWoss( const CoordZ& tx, const CoordZ& rx, double start_frequency, double
      end_frequency ) = 0;
277
278
279   };
280
281
288   class WossManagerResDb :  public WossManager {
289
290
291     public:
292
293
294     WossManagerResDb();
295
296     virtual ~WossManagerResDb() {}
297
298
308     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, const Time& time_value );
309
319 //     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, double time_value );
320
330     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, const Time& time_value );
331
341 //     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
      start_frequency, double end_frequency, double time_value );
342
343
349     WossManagerResDb& setWossDbManager( const WossDbManager* const ptr ) { woss_db_manager = ptr; return
      *this; }
350
351
```

```
352     protected:
353
354
358     const WossDbManager* woss_db_manager;
359
360
370     TimeArr* dbGetTimeArr( const CoordZ& tx, const CoordZ& rx, double frequency, const Time& time_value
    ) const;
371
380     void dbInsertTimeArr( const CoordZ& tx, const CoordZ& rx, double frequency, const Time& time_value,
    const TimeArr& channel ) const;
381
382
392     Pressure* dbGetPressure( const CoordZ& tx, const CoordZ& rx, double frequency, const Time&
    time_value ) const;
393
402     void dbInsertPressure( const CoordZ& tx, const CoordZ& rx, double frequency, const Time& time_value,
    const Pressure& press ) const;
403
404
405   };
406
407
408   //inline functions
410   inline void WossManagerResDb::dbInsertTimeArr( const CoordZ& tx, const CoordZ& rx, double frequency,
    const Time& time_value, const TimeArr& channel )const {
411     if ( woss_db_manager ) woss_db_manager->insertTimeArr( tx, rx, frequency, time_value, channel ) ;
412   }
413
414
415   inline TimeArr* WossManagerResDb::dbGetTimeArr( const CoordZ& tx, const CoordZ& rx, double frequency,
    const Time& time_value )const {
416     if ( woss_db_manager ) return( woss_db_manager->getTimeArr( tx, rx, frequency, time_value ) );
417     return( SDefHandler::instance()->getTimeArr()->create( TimeArr::createNotValid() ) );
418   }
419
420
421   inline void WossManagerResDb::dbInsertPressure( const CoordZ& tx, const CoordZ& rx, double frequency,
    const Time& time_value, const Pressure& press )const {
422     if ( woss_db_manager ) woss_db_manager->insertPressure( tx, rx, frequency, time_value, press );
423   }
424
425
426   inline Pressure* WossManagerResDb::dbGetPressure( const CoordZ& tx, const CoordZ& rx, double
    frequency, const Time& time_value )const {
427     if ( woss_db_manager ) return( woss_db_manager->getPressure( tx, rx, frequency, time_value ) );
428     return( SDefHandler::instance()->getPressure()->create( Pressure::createNotValid() ) );
429   }
430
431
432 #ifdef WOSS_MULTITHREAD
433
434
435 #include <set>
436
437
441 #define MAX_TOTAL_PTHREAD 32
442
443
453   class WossManagerResDbMT : public WossManagerResDb {
454
455
456     public:
457
458
462     WossManagerResDbMT();
463
464     virtual ~WossManagerResDbMT();
465
466
476     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
    start_frequency, double end_frequency, const Time& time_value );
477
487     virtual Pressure* getWossPressure( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
    start_frequency, double end_frequency, double time_value );
488
489
499     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
    start_frequency, double end_frequency, const Time& time_value );
500
510     virtual TimeArr* getWossTimeArr( const CoordZ& tx_coordz, const CoordZ& rx_coordz, double
    start_frequency, double end_frequency, double time_value );
511
512
521     virtual PressureVector getWossPressure( const CoordZPairVect& coordinates, double start_frequency,
    double end_frequency, const Time& time_value );
522
```

```
531     virtual PressureVector getWossPressure( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, double time_value );
532
541     virtual TimeArrVector getWossTimeArr( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, const Time& time_value );
542
551     virtual TimeArrVector getWossTimeArr( const CoordZPairVect& coordinates, double start_frequency,
      double end_frequency, double time_value );
552
558     void setConcurrentThreads( int number ) { concurrent_threads = number; checkConcurrentThreads(); }
559
564     int getConcurrentThreads() { return concurrent_threads; }
565
566
567     friend void* WMSMTcreateThreadTimeArr( void* ptr );
568
569     friend void* WMSMTcreateThreadPressure( void* ptr );
570
571
572     protected:
573
574
578     //typedef ::std::pair< CoordZPairVect, SimFreq > ThreadQuery;
579     struct ThreadQuery {
580
581
582       CoordZPairVect coordz_pair_vect;
583
584       SimFreq sim_freq;
585
586       int curr_index;
587
588       struct {
589           bool is_time_object;
590
591           Time time_cal;
592
593           double time_double;
594
595       } time_data;
596
597     };
598
599
603 //     typedef ::std::pair< CoordZPair, SimFreq > ThreadParam;
604     struct ThreadParam {
605
606
607       CoordZPair coordz_pair;
608
609       SimFreq sim_freq;
610
611       struct {
612           bool is_time_object;
613
614           Time time_cal;
615
616           double time_double;
617
618       } time_data;
619
620
621     };
622
623
627     typedef ::std::pair< int, ThreadParam > ThreadParamIndex;
628
629
630     struct ThreadCondSignal {
631
632
633       ThreadCondSignal() { pthread_mutex_init( &mutex, NULL ); pthread_cond_init( &condition, NULL );  }
634
635       ~ThreadCondSignal() { pthread_mutex_destroy( &mutex ); pthread_cond_destroy( &condition ); }
636
637
638       pthread_mutex_t mutex;
639
640       pthread_cond_t condition;
641
642
643     };
644
645
646     typedef ::std::map< Woss*, ThreadCondSignal* > ActiveWoss;
647     typedef ActiveWoss::iterator AWIter;
648     typedef ActiveWoss::reverse_iterator AWRIter;
```

```
649     typedef ActiveWoss::const_iterator AWCIter;
650     typedef ActiveWoss::const_reverse_iterator AWCRIter;
651
652
656     int max_thread_number;
657
661     int total_queries;
662
666     int concurrent_threads;
667
668
672     volatile int total_thread_created;
673
677     volatile int total_thread_ended;
678
679
683     pthread_t thread_controller;
684
688     pthread_t thread_arr[MAX_TOTAL_PTHREAD];
689
690
694     pthread_spinlock_t mutex;
695
699     pthread_spinlock_t request_mutex;
700
701
705     ThreadQuery thread_query;
706
707
711     TimeArrVector thread_time_arr_reply;
712
716     PressureVector thread_pressure_reply;
717
721     ActiveWoss active_woss;
722
726     void checkConcurrentThreads();
727
731     void initThreadVars();
732
733
738     ThreadParamIndex popThreadParamIndex();
739
740
746     void insertThreadReplyTimeArr( int index, woss::TimeArr* time_arr );
747
753     void insertThreadReplyPressure( int index, woss::Pressure* pressure );
754
755
756   };
757
758
764   void* WMSMTcreateThreadTimeArr( void* ptr );
765
771   void* WMSMTcreateThreadPressure( void* ptr );
772
773
774 #endif // WOSS_MULTITHREAD
775
776
777 }
778
779
780 #endif /* WOSS_MANAGER_DEFINITIONS_H */
781
```

14.36 woss/woss.cpp File Reference

Provides the implementation of woss::Woss and woss::WossResReader class.

Functions

- void woss::destroyWossSpinlock ()

14.36.1 Detailed Description

Provides the implementation of woss::Woss and woss::WossResReader class.

Author

Federico Guerra

Provides the implementation of the woss::Woss and woss::WossResReader classes

14.36.2 Function Documentation

14.36.2.1 destroyWossSpinlock() `void woss::destroyWossSpinlock ( )`

Function used to destroy the static pthread_spin_t woss_mutex

14.37 woss/woss.h File Reference

Provides the interface for woss::Woss and woss::WossResReader classes.

Classes

- class woss::Woss

  Abstract class that provides the interface for initializing and running a channel simulator.
- class woss::WossResReader

  Woss class with ResReader objects for reading simulated results.

Typedefs

- typedef ::std::vector< double > woss::RangeVector
- typedef ::std::set< double > woss::FreqSet
- typedef FreqSet::iterator **woss::FreqSIt**
- typedef FreqSet::const_iterator **woss::FreqSCIt**
- typedef FreqSet::reverse_iterator **woss::FreqSRIt**
- typedef FreqSet::const_reverse_iterator **woss::FreqSCRIt**
- typedef ::std::map< double, ResReader * > woss::ResReaderMap
- typedef ResReaderMap::iterator **woss::RRMIter**
- typedef ResReaderMap::reverse_iterator **woss::RRMRIter**
- typedef ResReaderMap::const_iterator **woss::RRMCIter**
- typedef ResReaderMap::const_reverse_iterator **woss::RRMCRIter**
- typedef ::std::pair< RRMIter, bool > **woss::RRMPair**

Functions

- void woss::destroyWossSpinlock ()

14.37.1 Detailed Description

Provides the interface for woss::Woss and woss::WossResReader classes.

Author

Federico Guerra

Provides the interface for the woss::Woss and woss::WossResReader classes

14.37.2 Typedef Documentation

14.37.2.1 FreqSet `typedef ::std::set< double > woss::FreqSet`

Set of frequencies [Hz]

14.37.2.2 RangeVector `typedef ::std::vector< double > woss::RangeVector`

Vector of range values [m]

14.37.2.3 ResReaderMap `typedef ::std::map< double, ResReader* > woss::ResReaderMap`

Map that links a frequency [Hz] to a ResReader instance

14.37.3 Function Documentation

14.37.3.1 destroyWossSpinlock() `void woss::destroyWossSpinlock ( )`

Function used to destroy the static pthread_spin_t woss_mutex

14.38 woss.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_PROGRAM_DEFINITIONS_H
41 #define WOSS_PROGRAM_DEFINITIONS_H
42
43
44 #include <set>
45 #include <vector>
46 #include <map>
47 #include <climits>
48 #include <coordinates-definitions.h>
49 #include <time-definitions.h>
50 #include "res-reader.h"
51
52
53 #ifdef WOSS_MULTITHREAD
54 #include <pthread.h>
55 #endif // WOSS_MULTITHREAD
56
57
58 namespace woss {
59
60
61   class WossDbManager;
62
66   typedef ::std::vector< double > RangeVector;
67
68
72   typedef ::std::set< double > FreqSet;
73   typedef FreqSet::iterator FreqSIt;
74   typedef FreqSet::const_iterator FreqSCIt;
75   typedef FreqSet::reverse_iterator FreqSRIt;
76   typedef FreqSet::const_reverse_iterator FreqSCRIt;
77
78
79   static const int WOSS_MIN_DEPTH = 0;
81   static const int WOSS_MAX_DEPTH = INT_MAX;
83   static const int WOSS_MIN_RANGE = -INT_MAX;
85   static const int WOSS_MAX_RANGE = INT_MAX;
94   class Woss {
95
96
97     public:
98
99
103     Woss();
104
115     Woss( const Coord& tx, const Coord& rx, const Time& start_t, const Time& end_t, double start_freq,
      double end_freq, double freq_step ) ;
116
117     virtual ~Woss();
118
119
124     virtual bool initialize() = 0;
125
126
```

```
132     virtual bool run() = 0;
133
134
140     virtual bool timeEvolve( const Time& time_value ) = 0;
141
142
147     virtual bool isValid() const = 0;
148
149
160     virtual Pressure* getAvgPressure( double frequency, double tx_depth, double start_rx_depth =
        WOSS_MIN_DEPTH, double start_rx_range = WOSS_MIN_RANGE, double end_rx_depth = WOSS_MAX_DEPTH , double
        end_rx_range = WOSS_MAX_RANGE ) const = 0;
161
170     virtual Pressure* getPressure( double frequency, double tx_depth, double rx_depth, double rx_range )
        const = 0;
171
172
181     virtual TimeArr* getTimeArr( double frequency, double tx_depth, double rx_depth, double rx_range )
        const = 0;
182
183
189     Woss& setDebug( bool flag ) { debug = flag; return *this; }
190
196     Woss& setCleanWorkDir( bool flag ) { clean_workdir = flag; return *this; }
197
198
204     Woss& setWorkDirPath( const ::std::string& path ) { work_dir_path = path; return *this; }
205
211     Woss& setWossDbManager( const WossDbManager* const ptr ) { db_manager = ptr; return *this; }
212
213
219     Woss& insertFrequency( double freq ) { frequencies.insert(freq); return *this; }
220
228     Woss& insertFrequencies( double freq_start, double freq_end, double freq_step );
229
235     Woss& setFrequencies( const FreqSet& freq_set ) { frequencies = freq_set; return *this; }
236
242     Woss& eraseFrequency( double freq ) { frequencies.erase(freq); return *this; }
243
248     Woss& clearFrequencies() { frequencies.clear(); return *this; }
249
250
256     Woss& setTotalRuns( int runs ) { total_runs = runs; return *this; }
257
263     Woss& setTxCoordZ( const CoordZ& coordz ) { tx_coordz = coordz; return *this; }
264
270     Woss& setRxCoordZ( const CoordZ& coordz ) { rx_coordz = coordz; return *this; }
271
272
278     Woss& setStartTime( const Time& start_t ) { start_time = start_t; return *this; }
279
285     Woss& setEndTime( const Time& end_t ) { end_time = end_t; return *this; }
286
292     Woss& setEvolutionTimeQuantum( double value ) { evolution_time_quantum = value; return *this; }
293
294
299     int getWossId()const { return woss_id; }
300
305     ::std::string getWorkDirPath()const { return work_dir_path; }
306
307
312     const FreqSet& getFrequencies()const { return frequencies; }
313
318     double getMinFrequency()const { return( *( frequencies.begin() ) ); }
319
324     double getMaxFrequency()const { return( *( frequencies.rbegin() ) ); }
325
326
331     FreqSCIt freq_begin()const { return( frequencies.begin() ); }
332
337     FreqSCIt freq_end()const { return( frequencies.end() ); }
338
343     FreqSCRIt freq_rbegin()const { return( frequencies.rbegin() ); }
344
349     FreqSCRIt freq_rend()const { return( frequencies.rend() ); }
350
356     FreqSCIt freq_lower_bound( double frequency )const { return( frequencies.lower_bound( frequency ) );
        }
357
363     FreqSCIt freq_upper_bound( double frequency )const { return( frequencies.upper_bound( frequency ) );
        }
364
365
370     int getTotalRuns()const { return total_runs; }
371
372
377     CoordZ getTxCoordZ()const { return tx_coordz; }
```

```
378
383     CoordZ getRxCoordZ()const { return rx_coordz; }
384
385
390     Time getStartTime()const { return start_time; }
391
396     Time getCurrentTime()const { return current_time; }
397
402     Time getEndTime()const { return end_time; }
403
408     double getEvolutionTimeQuantum()const { return evolution_time_quantum; }
409
415     double getGreatCircleDistance()const { return total_great_circle_distance; }
416
421     double getDistance()const { return total_distance; }
422
427     double getBearing()const { return bearing; }
428
432     bool usingDebug()const { return debug; }
433
437     virtual bool isRunning() const;
438
439
440     protected:
441
442
443 #ifdef WOSS_MULTITHREAD
447     static pthread_spinlock_t woss_mutex;
448
449     friend void destroyWossSpinlock();
450
451 #endif // WOSS_MULTITHREAD
452
456     static int woss_counter;
457
461     int woss_id;
462
463
467     ::std::string work_dir_path;
468
469
473     const WossDbManager* db_manager;
474
475
479     Time start_time;
480
484     Time current_time;
485
489     Time end_time;
490
496     double evolution_time_quantum;
497
501     CoordZ tx_coordz;
502
506     CoordZ rx_coordz;
507
508
512     FreqSet frequencies;
513
514
519     double bearing;
520
525     double total_great_circle_distance;
526
531     double total_distance;
532
536     int total_runs;
537
538
542     bool debug;
543
544
545     bool has_run_once;
546
547
551     volatile bool is_running;
552
556     bool clean_workdir;
557
564     virtual bool mkWorkDir( double curr_frequency, int curr_run = 0 );
565
572     virtual bool rmWorkDir( double curr_frequency, int curr_run = 0 );
573
574     virtual bool rmWorkDir();
575
576
577   };
```

```
578
579
580 #ifdef WOSS_MULTITHREAD
581
585   void destroyWossSpinlock();
586
587 #endif // WOSS_MULTITHREAD
588
589
593   typedef ::std::map< double, ResReader* > ResReaderMap;
594   typedef ResReaderMap::iterator RRMIter;
595   typedef ResReaderMap::reverse_iterator RRMRIter;
596   typedef ResReaderMap::const_iterator RRMCIter;
597   typedef ResReaderMap::const_reverse_iterator RRMCRIter;
598   typedef ::std::pair< RRMIter, bool > RRMPair;
599
600
607   class WossResReader :  public Woss {
608
609
610     public:
611
612
616     WossResReader() :  Woss() { }
617
628     WossResReader( const CoordZ& tx, const CoordZ& rx, const Time& start_t, const Time& end_t,
629                    double start_freq, double end_freq, double freq_step )
630                    :  Woss( tx, rx, start_t, end_t, start_freq, end_freq, freq_step ) { }
631
632
633     virtual ~WossResReader() { clearResReaderMap(); }
634
635
641     virtual bool initResReader( double curr_frequency ) = 0;
642
643
644     protected:
645
646
650     ResReaderMap res_reader_map;
651
652
656     void clearResReaderMap();
657
658
659   };
660
661
662 }
663
664
665 #endif /* WOSS_PROGRAM_DEFINITIONS_H */
666
667
```

14.39 woss/woss_db/bathymetry-gebco-db-creator.cpp File Reference

Provides the implementation of woss::BathyGebcoDbCreator class.

14.39.1 Detailed Description

Provides the implementation of woss::BathyGebcoDbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::BathyGebcoDbCreator class

14.40 woss/woss_db/bathymetry-gebco-db-creator.h File Reference

Provides the interface for woss::BathyGebcoDbCreator class.

Classes

- class woss::BathyGebcoDbCreator

  WossDbCreator for the GEBCO bathymetry database.

14.40.1 Detailed Description

Provides the interface for woss::BathyGebcoDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::BathyGebcoDbCreator class

14.41 bathymetry-gebco-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_BATHYMETRY_GEBCO_DB_CREATOR_H
41 #define WOSS_BATHYMETRY_GEBCO_DB_CREATOR_H
42
43 #ifdef WOSS_NETCDF_SUPPORT
44
45
46 #include "woss-db-creator.h"
47 #include "bathymetry-gebco-db.h"
48
49
50 namespace woss {
51
52
61   class BathyGebcoDbCreator :  public WossDbCreator {
62
63
64     public:
65
66
70     BathyGebcoDbCreator();
71
72     virtual ~BathyGebcoDbCreator();
73
74
79     virtual WossDb* const createWossDb();
```

```
80
86      BathyGebcoDbCreator& setGebcoBathyType( GEBCO_BATHY_TYPE bathy_type ) {
87        gebco_type = bathy_type; return *this; }
88
93      GEBCO_BATHY_TYPE getGebcoBathyType() { return gebco_type; }
94
95      protected:
96
97
102      GEBCO_BATHY_TYPE gebco_type;
103
104
110      virtual bool initializeDb( WossDb* const woss_db );
111
112
113    };
114
115 }
116
117 #endif // WOSS_NETCDF_SUPPORT
118
119 #endif /* WOSS_BATHYMETRY_GEBCO_DB_CREATOR_H */
120
```

14.42 woss/woss_db/bathymetry-gebco-db.cpp File Reference

Provides the implementation of woss::BathyGebcoDb class.

14.42.1 Detailed Description

Provides the implementation of woss::BathyGebcoDb class.

Author

Federico Guerra

Provides the implementation of the woss::BathyGebcoDb class

14.43 woss/woss_db/bathymetry-gebco-db.h File Reference

Provides the interface for woss::BathyGebcoDb class.

Classes

- class woss::BathyGebcoDb
  NetCDF specialization of WossNetcdfDb for GEBCO database.

Typedefs

- typedef std::pair< long, long > woss::Gebco2DIndexes

Enumerations

- enum woss::GEBCO_BATHY_TYPE {
  woss::GEBCO_1D_1_MINUTE_BATHY_TYPE = 0 , woss::GEBCO_1D_30_SECONDS_BATHY_TYPE = 1 ,
  woss::GEBCO_2D_1_MINUTE_BATHY_TYPE = 2 , woss::GEBCO_2D_30_SECONDS_BATHY_TYPE = 3 ,
  woss::GEBCO_2D_15_SECONDS_BATHY_TYPE = 4 , woss::GEBCO_INVALID_BATHY_TYPE }

14.43.1 Detailed Description

Provides the interface for woss::BathyGebcoDb class.

Author

Federico Guerra

Provides the interface for the woss::BathyGebcoDb class

14.43.2 Typedef Documentation

14.43.2.1 Gebco2DIndexes `typedef std::pair< long, long > woss::Gebco2DIndexes`

GEBCO 2D netcdf indexes

14.43.3 Enumeration Type Documentation

14.43.3.1 GEBCO_BATHY_TYPE `enum woss::GEBCO_BATHY_TYPE`

GEBCO version in use

Enumerator

GEBCO_1D_1_MINUTE_BATHY_TYPE	GEBCO 1D, one minute of arc netcf format.
GEBCO_1D_30_SECONDS_BATHY_TYPE	GEBCO 1D, thirty seconds of arc netcf format.
GEBCO_2D_1_MINUTE_BATHY_TYPE	GEBCO 2D, one minute of arc netcf format.
GEBCO_2D_30_SECONDS_BATHY_TYPE	GEBCO 2D, thirty seconds of arc netcf format.
GEBCO_2D_15_SECONDS_BATHY_TYPE	GEBCO 2D, fifteen seconds of arc netcf format.
GEBCO_INVALID_BATHY_TYPE	INVALID, must be last.

14.44 bathymetry-gebco-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
```

```
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
39 #ifndef WOSS_BATHYMETRY_GEBCO_DB_H
40 #define WOSS_BATHYMETRY_GEBCO_DB_H
41
42 #ifdef WOSS_NETCDF_SUPPORT
43
44 #include "woss-db.h"
45 #include <utility>
46 #if defined (WOSS_NETCDF4_SUPPORT)
47 #include <ncVar.h>
48 #endif // defined (WOSS_NETCDF4_SUPPORT)
49
50 namespace woss {
51
52
53   static const int GEBCO_1_MINUTE_BATHY_NLAT = 10801;
55   static const int GEBCO_1_MINUTE_BATHY_NLON = 21601;
57   static const double GEBCO_1_MINUTE_BATHY_SPACING = 0.0166666666666667;
60   static const int GEBCO_30_SECONDS_BATHY_NLAT = 21600;
62   static const int GEBCO_30_SECONDS_BATHY_NLON = 43200;
64   static const double GEBCO_30_SECONDS_BATHY_SPACING = 0.00833333333333333;
67   static const int GEBCO_15_SECONDS_BATHY_NLAT = 43200;
69   static const int GEBCO_15_SECONDS_BATHY_NLON = 86400;
71   static const double GEBCO_15_SECONDS_BATHY_SPACING = 0.004166666666666665;
74   static const double GEBCO_1_MINUTE_BATHY_MIN_LAT = -90.0;
76   static const double GEBCO_1_MINUTE_BATHY_MAX_LAT = 90.0;
78   static const double GEBCO_1_MINUTE_BATHY_MIN_LONG = -180.0;
80   static const double GEBCO_1_MINUTE_BATHY_MAX_LONG = 180.0;
83   static const double GEBCO_30_SECONDS_BATHY_MIN_LAT = -89.995833333333333335;
85   static const double GEBCO_30_SECONDS_BATHY_MAX_LAT = 89.995833333333333335;
87   static const double GEBCO_30_SECONDS_BATHY_MIN_LONG = -179.995833333333333335;
89   static const double GEBCO_30_SECONDS_BATHY_MAX_LONG = 179.995833333333333335;
92   static const double GEBCO_15_SECONDS_BATHY_MIN_LAT = -89.997916666666666675;
94   static const double GEBCO_15_SECONDS_BATHY_MAX_LAT = 89.997916666666666675;
96   static const double GEBCO_15_SECONDS_BATHY_MIN_LONG = -179.997916666666666675;
98   static const double GEBCO_15_SECONDS_BATHY_MAX_LONG = 179.997916666666666675;
101   static const double GEBCO_1D_1_MINUTE_BATHY_START_LAT = 90.0;
103   static const double GEBCO_1D_1_MINUTE_BATHY_START_LONG = -180.0;
106   static const double GEBCO_1D_30_SECONDS_BATHY_START_LAT = 89.995833333333333335;
108   static const double GEBCO_1D_30_SECONDS_BATHY_START_LONG = -179.995833333333333335;
111   static const double GEBCO_2D_1_MINUTE_BATHY_START_LAT = -90.0;
113   static const double GEBCO_2D_1_MINUTE_BATHY_START_LONG = -180.0;
116   static const double GEBCO_2D_30_SECONDS_BATHY_START_LAT = -89.995833333333333335;
118   static const double GEBCO_2D_30_SECONDS_BATHY_START_LONG = -179.995833333333333335;
121   static const double GEBCO_2D_15_SECONDS_BATHY_START_LAT = -89.997916666666666675;
123   static const double GEBCO_2D_15_SECONDS_BATHY_START_LONG = -179.997916666666666675;
127   typedef std::pair< long, long > Gebco2DIndexes;
132   enum GEBCO_BATHY_TYPE {
133     GEBCO_1D_1_MINUTE_BATHY_TYPE    = 0,
134     GEBCO_1D_30_SECONDS_BATHY_TYPE = 1,
135     GEBCO_2D_1_MINUTE_BATHY_TYPE    = 2,
136     GEBCO_2D_30_SECONDS_BATHY_TYPE = 3,
137     GEBCO_2D_15_SECONDS_BATHY_TYPE = 4,
138     GEBCO_INVALID_BATHY_TYPE
139   };
140
141
148   class BathyGebcoDb :  public WossNetcdfDb, public WossBathymetryDb {
149
150
151     public:
152
153
158     BathyGebcoDb( const ::std::string& name );
159
160     virtual ~BathyGebcoDb() { }
161
162
169     virtual bool insertValue( const Coord& coordinates, const Bathymetry& bathymetry_value );
170
171
178     virtual double getValue( const Coord& coords ) const ;
```

```
179
180
185     virtual bool finalizeConnection();
186
187
192     void setGebcoType( const GEBCO_BATHY_TYPE& type ) { gebco_type = type; }
193
198     GEBCO_BATHY_TYPE getGebcoType() { return gebco_type; }
199
200
201     protected:
202
203
207     GEBCO_BATHY_TYPE gebco_type;
208
209
213 #if defined(WOSS_NETCDF4_SUPPORT)
214     netCDF::NcVar bathy_var;
215 #else
216     NcVar* bathy_var;
217 #endif // #if defined(WOSS_NETCDF4_SUPPORT)
218
219
223 #if defined(WOSS_NETCDF4_SUPPORT)
224     netCDF::NcVar lat_var;
225 #else
226     NcVar* lat_var;
227 #endif // defined(WOSS_NETCDF4_SUPPORT)
228
232 #if defined(WOSS_NETCDF4_SUPPORT)
233     netCDF::NcVar lon_var;
234 #else
235     NcVar* lon_var;
236 #endif // defined(WOSS_NETCDF4_SUPPORT)
237
238
245     long get1DBathyIndex( const Coord& coords ) const ;
246
253     Gebco2DIndexes get2DBathyIndexes( const Coord& coords ) const ;
254   };
255
256 }
257
258 #endif // WOSS_NETCDF_SUPPORT
259
260 #endif /* WOSS_BATHYMETRY_GEBCO_DB_H */
261
262
263
```

14.45 bathymetry-utm-csv-db-creator.h

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_BATHYMETRY_UMT_CSV_DB_CREATOR_H
41 #define WOSS_BATHYMETRY_UMT_CSV_DB_CREATOR_H
42
```

```
43 #include <utility>
44 #include "woss-db-creator.h"
45
46
47 namespace woss {
48
49
56   class BathyUtmCsvDbCreator :  public WossDbCreator {
57
58
59     public:
60
61
65     BathyUtmCsvDbCreator();
66
67     virtual ~BathyUtmCsvDbCreator();
68
69
74     virtual WossDb* const createWossDb();
75
82     BathyUtmCsvDbCreator& setCSVSeparator( const char new_separator ) { separator = new_separator; return
     *this; }
83
89     const char getCSVSeparator()const { return separator; }
90
97     BathyUtmCsvDbCreator& setDbSpacing( double spacing ) { db_spacing = spacing; return *this; }
98
104     double getDbSpacing()const { return db_spacing; }
105
113     BathyUtmCsvDbCreator& setDbTotalValues(int nnorth, int neast) {
114       total_northing_values = nnorth;
115       total_easting_values = neast;
116       return *this;
117     }
118
124     std::pair<int, int> getDbTotalValues()const {
125       return std::make_pair(total_northing_values, total_easting_values);
126     }
127
135     BathyUtmCsvDbCreator& setDbRangeEasting(double start, double end) {
136       range_easting_start = start;
137       range_easting_end = end;
138       return *this;
139     }
140
146     std::pair<double, double> getDbRangeEasting()const {
147       return std::make_pair(range_easting_start, range_easting_end);
148     }
149
157     BathyUtmCsvDbCreator& setDbRangeNorthing(double start, double end) {
158       range_northing_start = start;
159       range_northing_end = end;
160       return *this;
161     }
162
168     std::pair<double, double> getDbRangeNorthing()const {
169       return std::make_pair( range_northing_start, range_northing_end);
170     }
171
177     BathyUtmCsvDbCreator& setLandApproximationFlag(bool flag) {
178       approx_land_to_sea_surface = flag;
179       return *this;
180     }
181
187     bool getLandApproximationFlag() {
188       return approx_land_to_sea_surface;
189     }
190
191     protected:
192
193     char separator;
195     double db_spacing;
197     int total_northing_values;
199     int total_easting_values;
201     double range_easting_start;
203     double range_easting_end;
205     double range_northing_start;
207     double range_northing_end;
209     bool approx_land_to_sea_surface;
216     virtual bool initializeDb( WossDb* const woss_db );
217
218
219   };
220
221 }
222
223
```

```
224 #endif /* WOSS_BATHYMETRY_UMT_CSV_DB_CREATOR_H */
225
```

14.46 bathymetry-utm-csv-db.h

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
39 #ifndef WOSS_BATHYMETRY_UTM_CSV_DB_H
40 #define WOSS_BATHYMETRY_UTM_CSV_DB_H
41
42 #include "woss-db.h"
43 #include <vector>
44 #include <utility>
45
46 namespace woss {
47
48
55   class BathyUtmCsvDb :  public WossTextualDb, public WossBathymetryDb {
56     public:
57
63     BathyUtmCsvDb( const ::std::string& name );
64
65     virtual ~BathyUtmCsvDb() { }
66
67
75     virtual bool insertValue( const Coord& coordinates, const Bathymetry& bathymetry_value );
76
84     virtual double getValue( const Coord& coords ) const ;
85
86
92     virtual bool finalizeConnection();
93
99     void setCSVSeparator( const char new_separator ) { separator = new_separator; }
100
106     const char getCSVSeparator()const { return separator; }
107
113     void setDbSpacing( double spacing ) { db_spacing = spacing; }
114
120     double getDbSpacing()const { return db_spacing; }
121
128     void setDbTotalValues(int nnorth, int neast) {
129       total_northing_values = nnorth;
130       total_easting_values = neast;
131     }
132
138     std::pair<int, int> getDbTotalValues()const {
139       return std::make_pair(total_northing_values, total_easting_values);
140     }
141
148     void setDbRangeEasting(double start, double end) {
149       range_easting_start = start;
150       range_easting_end = end;
151     }
152
158     std::pair<double, double> getDbRangeEasting()const {
159       return std::make_pair(range_easting_start, range_easting_end);
160     }
161
```

```
168     void setDbRangeNorthing(double start, double end) {
169       range_northing_start = start;
170       range_northing_end = end;
171     }
172
178     std::pair<double, double> getDbRangeNorthing()const {
179       return std::make_pair( range_northing_start, range_northing_end);
180     }
181
186     void setLandApproximationFlag(bool flag) {
187       approx_land_to_sea_surface = flag;
188     }
189
195     bool getLandApproximationFlag() {
196       return approx_land_to_sea_surface;
197     }
198
199     protected:
200
208     int getBathyIndex( const Coord& coords ) const ;
209
215     virtual bool importData();
216
217
218     std::vector<double> bathy_vec;
220     char separator;
222     double db_spacing;
224     int total_northing_values;
226     int total_easting_values;
228     double range_easting_start;
230     double range_easting_end;
232     double range_northing_start;
234     double range_northing_end;
236     bool approx_land_to_sea_surface;
238     static const double land_approximation_depth;
239   };
240
241 }
242
243 #endif /* WOSS_BATHYMETRY_UTM_CSV_DB_H */
244
245
246
```

14.47 woss/woss_db/res-pressure-bin-db-creator.cpp File Reference

Provides the implementation of woss::ResPressureBinDbCreator class.

14.47.1 Detailed Description

Provides the implementation of woss::ResPressureBinDbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::ResPressureBinDbCreator class

14.48 woss/woss_db/res-pressure-bin-db-creator.h File Reference

Provides the interface for woss::ResPressureBinDbCreator class.

Classes

- class woss::ResPressureBinDbCreator

  DbCreator for binary Pressure database.

14.48.1 Detailed Description

Provides the interface for woss::ResPressureBinDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::ResPressureBinDbCreator class

14.49 res-pressure-bin-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_PRESSURE_BIN_DB_CREATOR_H
41 #define WOSS_RES_PRESSURE_BIN_DB_CREATOR_H
42
43
44 #include "woss-db-creator.h"
45
46
47 namespace woss {
48
49
55   class ResPressureBinDbCreator :  public WossDbCreator {
56
57
58     public:
59
60
64     ResPressureBinDbCreator();
65
66     virtual ~ResPressureBinDbCreator();
67
68
73     virtual WossDb* const createWossDb();
74
75
76     void setSpaceSampling( double value ) { space_sampling = value; }
77
78     double getSpaceSampling() { return space_sampling; }
79
80
81     protected:
82
83
84     double space_sampling;
85
86
```

```
92     virtual bool initializeDb( WossDb* const woss_db );
93
94
95   };
96
97
98 }
99
100
101 #endif /* WOSS_RES_PRESSURE_BIN_DB_CREATOR_H */
102
```

14.50 woss/woss_db/res-pressure-bin-db.cpp File Reference

Provides the implementation of woss::ResPressureBinDb class.

14.50.1 Detailed Description

Provides the implementation of woss::ResPressureBinDb class.

Author

Federico Guerra

Provides the implementation of the woss::ResPressureBinDb class

14.51 woss/woss_db/res-pressure-bin-db.h File Reference

Provides the interface for woss::ResPressureBinDb class.

Classes

- class woss::ResPressureBinDb

  Binary WossDb for Pressure.

14.51.1 Detailed Description

Provides the interface for woss::ResPressureBinDb class.

Author

Federico Guerra

Provides the interface for the woss::ResPressureBinDb class

14.52 res-pressure-bin-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_PRESSURE_BIN_DB_H
41 #define WOSS_RES_PRESSURE_BIN_DB_H
42
43
44 #include "res-pressure-txt-db.h"
45
46
47 namespace woss {
48
49
56   class ResPressureBinDb :  public ResPressureTxtDb {
57
58
59     public:
60
61
66     ResPressureBinDb( const ::std::string& name ) :  ResPressureTxtDb(name) { }
67
68     virtual ~ResPressureBinDb() { }
69
70
71     protected:
72
73
79     virtual bool writeMap();
80
86     virtual bool importMap();
87
88
89   };
90
91
92 }
93 #endif /* WOSS_RES_PRESSURE_BIN_DB_H */
94
95
96
```

14.53 woss/woss_db/res-pressure-txt-db-creator.cpp File Reference

Provides the implementation of woss::ResPressureTxtDbCreator class.

14.53.1 Detailed Description

Provides the implementation of woss::ResPressureTxtDbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::ResPressureTxtDbCreator class

14.54 woss/woss_db/res-pressure-txt-db-creator.h File Reference

Provides the interface for woss::ResPressureTxtDbCreator class.

Classes

- class woss::ResPressureTxtDbCreator

  DbCreator for textual Pressure database.

14.54.1 Detailed Description

Provides the interface for woss::ResPressureTxtDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::ResPressureTxtDbCreator class

14.55 res-pressure-txt-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_PRESSURE_TXT_DB_CREATOR_H
41 #define WOSS_RES_PRESSURE_TXT_DB_CREATOR_H
42
43
44 #include "woss-db-creator.h"
```

```
45
46
47 namespace woss {
48
49
55   class ResPressureTxtDbCreator :  public virtual WossDbCreator {
56
57
58     public:
59
60
64     ResPressureTxtDbCreator();
65
66     virtual ~ResPressureTxtDbCreator();
67
68
73     virtual WossDb* const createWossDb();
74
75
76     void setSpaceSampling( double value ) { space_sampling = value; }
77
78     double getSpaceSampling() { return space_sampling; }
79
80
81     protected:
82
83
84     double space_sampling;
85
86
92     virtual bool initializeDb( WossDb* const woss_db );
93
94
95   };
96
97
98 }
99
100
101 #endif /* WOSS_RES_PRESSURE_TXT_DB_CREATOR_H */
102
```

14.56 woss/woss_db/res-pressure-txt-db.cpp File Reference

Provides the implementation of woss::ResPressureTxtDb class.

14.56.1 Detailed Description

Provides the implementation of woss::ResPressureTxtDb class.

Author

Federico Guerra

Provides the implementation of the woss::ResPressureTxtDb class

14.57 woss/woss_db/res-pressure-txt-db.h File Reference

Provides the interface for woss::ResPressureTxtDb class.

Classes

- class woss::ResPressureTxtDb

  Textual WossDb for Pressure.

14.57.1 Detailed Description

Provides the interface for woss::ResPressureTxtDb class.

Author

Federico Guerra

Provides the interface for the woss::ResPressureTxtDb class

14.58 res-pressure-txt-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_PRESSURE_TXT_DB_H
41 #define WOSS_RES_PRESSURE_TXT_DB_H
42
43
44 #include <map>
45 #include <complex>
46 #include <custom-precision-double.h>
47 #include "woss-db.h"
48
49
50 namespace woss {
51
52
59   class ResPressureTxtDb :  public WossTextualDb, public WossResPressDb {
60
61
62     public:
63
64
69     ResPressureTxtDb( const ::std::string& name );
70
71     virtual ~ResPressureTxtDb() { }
72
73
78     virtual bool closeConnection();
79
80
85     virtual bool finalizeConnection();
86
87
98     virtual Pressure* getValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value ) const;
99
100
```

```
109     virtual bool insertValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
    const Time& time_value, const Pressure& pressure );
110
111
112     static void setSpaceSampling( double value ) { space_sampling = value; }
113
114     static double getSpaceSampling() { return space_sampling; }
115
116
117     protected:
118
119
120     typedef ::std::map< time_t, ::std::complex< double > > TimeMap;
121     typedef TimeMap::iterator TMIter;
122     typedef TimeMap::const_iterator TMCIter;
123     typedef TimeMap::reverse_iterator TMRIter;
124
125     typedef ::std::map< PDouble, TimeMap > FreqMap;
126     typedef FreqMap::iterator FMIter;
127     typedef FreqMap::reverse_iterator FMRIter;
128
129     typedef ::std::map< CoordZ, FreqMap, CoordComparator< ResPressureTxtDb, CoordZ > > RxMap;
130     typedef RxMap::iterator RxMIter;
131     typedef RxMap::reverse_iterator RxMRIter;
132
137     typedef ::std::map< CoordZ, RxMap, CoordComparator< ResPressureTxtDb, CoordZ > > PressureMatrix;
138     typedef PressureMatrix::iterator PMIter;
139     typedef PressureMatrix::const_iterator PMCIter;
140     typedef PressureMatrix::reverse_iterator PMRIter;
141     typedef PressureMatrix::const_reverse_iterator PMCRIter;
142
143
144     static double space_sampling;
145
146
150     PressureMatrix pressure_map;
151
155     int initial_pressmap_size;
156
157     bool has_been_modified;
158
159
165     void printScreenMap();
166
172     virtual bool writeMap();
173
179     virtual bool importMap();
180
181
190     ::std::complex<double> readMap( const CoordZ& tx, const CoordZ& rx, const double frequency, const
    Time& time_value ) const;
191
192
193   };
194
195
196 }
197 #endif /* WOSS_RES_PRESSURE_TXT_DB_H */
198
199
200
```

14.59 woss/woss_db/res-time-arr-bin-db-creator.cpp File Reference

Provides the implementation of ResTimeArrBinDbCreator class.

14.59.1 Detailed Description

Provides the implementation of ResTimeArrBinDbCreator class.

Author

Federico Guerra

Provides the implementation of the ResTimeArrBinDbCreator class

14.60 woss/woss_db/res-time-arr-bin-db-creator.h File Reference

Provides the interface for woss::ResTimeArrBinDbCreator class.

Classes

- class woss::ResTimeArrBinDbCreator
  DbCreator for textual TimeArr database.

14.60.1 Detailed Description

Provides the interface for woss::ResTimeArrBinDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrBinDbCreator class

14.61 res-time-arr-bin-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_TIME_ARR_BIN_DB_CREATOR_H
41 #define WOSS_RES_TIME_ARR_BIN_DB_CREATOR_H
42
43
44 #include "woss-db-creator.h"
45
46
47 namespace woss {
48
49
55   class ResTimeArrBinDbCreator :  public WossDbCreator {
56
57
58     public:
59
60
```

```
64      ResTimeArrBinDbCreator();
65
66      virtual ~ResTimeArrBinDbCreator();
67
68
73      virtual WossDb* const createWossDb();
74
75
76      void setSpaceSampling( double value ) { space_sampling = value; }
77
78      double getSpaceSampling() { return space_sampling; }
79
80
81      protected:
82
83
84      double space_sampling;
85
86
92      virtual bool initializeDb( WossDb* const woss_db );
93
94
95
96    };
97
98  }
99
100
101 #endif /* WOSS_RES_TIME_ARR_BIN_DB_CREATOR_H */
102
```

14.62 woss/woss_db/res-time-arr-bin-db.cpp File Reference

Provides the interface for woss::ResTimeArrBinDb class.

14.62.1 Detailed Description

Provides the interface for woss::ResTimeArrBinDb class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrBinDb class

14.63 woss/woss_db/res-time-arr-bin-db.h File Reference

Provides the interface for woss::ResTimeArrBinDb class.

Classes

- class woss::ResTimeArrBinDb

  Binary WossDb for TimeArr.

14.63.1 Detailed Description

Provides the interface for woss::ResTimeArrBinDb class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrBinDb class

14.64 res-time-arr-bin-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_TIME_ARR_BIN_DB_H
41 #define WOSS_RES_TIME_ARR_BIN_DB_H
42
43
44 #include "res-time-arr-txt-db.h"
45
46
47 namespace woss {
48
49
56   class ResTimeArrBinDb :  public ResTimeArrTxtDb {
57
58
59     public:
60
61
66     ResTimeArrBinDb( const ::std::string& name ) :  ResTimeArrTxtDb(name) { }
67
68     virtual ~ResTimeArrBinDb() { }
69
70
71     static void setSpaceSampling( double value ) { space_sampling = value; }
72
73     static double getSpaceSampling() { return space_sampling; }
74
75
76     protected:
77
78
85     virtual bool writeMap();
86
87
94     virtual bool importMap();
95
96
97   };
98
99
100 }
101
102
103 #endif /* WOSS_RES_TIME_ARR_BIN_DB_H */
104
105
```

14.65 woss/woss_db/res-time-arr-txt-db-creator.cpp File Reference

Provides the implementation of ResTimeArrTxtDbCreator class.

14.65.1 Detailed Description

Provides the implementation of ResTimeArrTxtDbCreator class.

Author

Federico Guerra

Provides the implementation of the ResTimeArrTxtDbCreator class

14.66 woss/woss_db/res-time-arr-txt-db-creator.h File Reference

Provides the interface for woss::ResTimeArrTxtDbCreator class.

Classes

- class woss::ResTimeArrTxtDbCreator

  DbCreator for textual TimeArr database.

14.66.1 Detailed Description

Provides the interface for woss::ResTimeArrTxtDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrTxtDbCreator class

14.67 res-time-arr-txt-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
```

```
27 */
28
29
40 #ifndef WOSS_RES_TIME_ARR_TXT_DB_CREATOR_H
41 #define WOSS_RES_TIME_ARR_TXT_DB_CREATOR_H
42
43
44 #include "woss-db-creator.h"
45
46
47 namespace woss {
48
49
55   class ResTimeArrTxtDbCreator :  public WossDbCreator {
56
57
58     public:
59
60
64     ResTimeArrTxtDbCreator();
65
66     virtual ~ResTimeArrTxtDbCreator();
67
68
73     virtual WossDb* const createWossDb();
74
75
76     void setSpaceSampling( double value ) { space_sampling = value; }
77
78     double getSpaceSampling() { return space_sampling; }
79
80
81     protected:
82
83
84     double space_sampling;
85
86
92     virtual bool initializeDb( WossDb* const woss_db );
93
94
95
96   };
97
98 }
99
100
101 #endif /* WOSS_RES_TIME_ARR_TXT_DB_CREATOR_H */
102
```

14.68 woss/woss_db/res-time-arr-txt-db.cpp File Reference

Provides the interface for woss::ResTimeArrTxtDb class.

14.68.1 Detailed Description

Provides the interface for woss::ResTimeArrTxtDb class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrTxtDb class

14.69 woss/woss_db/res-time-arr-txt-db.h File Reference

Provides the interface for woss::ResTimeArrTxtDb class.

Classes

- class woss::ResTimeArrTxtDb

  Textual WossDb for TimeArr.

14.69.1 Detailed Description

Provides the interface for woss::ResTimeArrTxtDb class.

Author

Federico Guerra

Provides the interface for the woss::ResTimeArrTxtDb class

14.70 res-time-arr-txt-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_RES_TIME_ARR_TXT_DB_H
41 #define WOSS_RES_TIME_ARR_TXT_DB_H
42
43
44 #include <coordinates-definitions.h>
45 #include <time-arrival-definitions.h>
46 #include "woss-db.h"
47
48
49 namespace woss {
50
51
58   class ResTimeArrTxtDb :  public WossTextualDb, public WossResTimeArrDb {
59
60
61     public:
62
63
68     ResTimeArrTxtDb( const ::std::string& name );
69
70     virtual ~ResTimeArrTxtDb() { }
71
72
77     virtual bool finalizeConnection();
78
```

```
79
84      virtual bool closeConnection();
85
86
97      virtual TimeArr* getValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value ) const ;
98
107       virtual bool insertValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double
      frequency,const Time& time_value, const TimeArr& channel ) ;
108
109
110     static void setSpaceSampling( double value ) { space_sampling = value; }
111
112     static double getSpaceSampling() { return space_sampling; }
113
114
115     protected:
116
117
118     typedef ::std::map< time_t, TimeArr > TimeMap;
119     typedef TimeMap::iterator TMIter;
120     typedef TimeMap::const_iterator TMCIter;
121     typedef TimeMap::reverse_iterator TMRIter;
122
123     typedef ::std::map< PDouble, TimeMap > FreqMap;
124     typedef FreqMap::iterator FMIter;
125     typedef FreqMap::const_iterator FMCIter;
126     typedef FreqMap::reverse_iterator FMRIter;
127
128     typedef ::std::map< CoordZ, FreqMap, CoordComparator< ResTimeArrTxtDb, CoordZ > > RxMap;
129 //      typedef ::std::map< CoordZ, FreqMap > RxMap;
130     typedef RxMap::iterator RxMIter;
131     typedef RxMap::const_iterator RxMCIter;
132     typedef RxMap::reverse_iterator RxMRIter;
133
138     typedef ::std::map< CoordZ, RxMap, CoordComparator< ResTimeArrTxtDb, CoordZ > > ArrMatrix;
139 //      typedef ::std::map< CoordZ, RxMap > ArrMatrix;
140     typedef ArrMatrix::iterator AMXIter;
141     typedef ArrMatrix::const_iterator AMXCIter;
142     typedef ArrMatrix::reverse_iterator AMXRIter;
143     typedef ArrMatrix::const_reverse_iterator AMXCRIter;
144
145
146     static double space_sampling;
147
148
152     ArrMatrix arrivals_map;
153
157     int initial_arrmap_size;
158
159
160     bool has_been_modified;
161
162
169     void printScreenMap();
170
171
178     virtual bool writeMap();
179
180
187     virtual bool importMap();
188
189
198     const TimeArr* readMap( const CoordZ& tx, const CoordZ& rx, const double frequency, const Time&
      time_value ) const;
199
200
201   };
202
203
204 }
205
206
207 #endif /* WOSS_RES_TIME_ARR_TXT_DB_H */
208
209
```

14.71 woss/woss_db/sediment-deck41-coord-db.cpp File Reference

Provides the implementation of woss::SedimDeck41CoordDb class.

14.71.1 Detailed Description

Provides the implementation of woss::SedimDeck41CoordDb class.

Author

Federico Guerra

Provides the implementation of the woss::SedimDeck41CoordDb class

14.72 woss/woss_db/sediment-deck41-coord-db.h File Reference

Provides the interface for woss::SedimDeck41CoordDb class.

Classes

- class woss::SedimDeck41CoordDb

  WossDb for custom made NetCDF DECK41 Sediment database.

Enumerations

- enum woss::DECK41DbType { woss::DECK41_DB_V1_TYPE = 0 , woss::DECK41_DB_V2_TYPE = 1 , woss::DECK41_DB_INVALID_TYPE }

14.72.1 Detailed Description

Provides the interface for woss::SedimDeck41CoordDb class.

Author

Federico Guerra

Provides the interface for the woss::SedimDeck41CoordDb class

14.72.2 Enumeration Type Documentation

14.72.2.1 DECK41DbType `enum woss::DECK41DbType`

DECK41 db version in use

Enumerator

DECK41_DB_V1_TYPE	DECK41 V1 NetCDF Legacy db.
DECK41_DB_V2_TYPE	DECK41 V2 NetCDF4 db.
DECK41_DB_INVALID_TYPE	INVALID, must be last.

14.73 sediment-deck41-coord-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DECK41_COORD_DB_H
41 #define WOSS_SEDIMENT_DECK41_COORD_DB_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46 #include <sediment-definitions.h>
47 #include "woss-db.h"
48 #if defined (WOSS_NETCDF4_SUPPORT)
49 #include <ncVar.h>
50 #endif // defined (WOSS_NETCDF4_SUPPORT)
51
52 namespace woss {
53
54
55   static const int DECK41_MINUTES_SEDIM_NLAT = 10801;
57   static const int DECK41_MINUTES_SEDIM_NLON = 21601;
59   static const double DECK41_MINUTES_SEDIM_SPACING = 0.0166666666666667;
61   static const int DECK41_V2_SEDIM_NLAT = 10800;
63   static const int DECK41_V2_SEDIM_NLON = 21600;
65   static const double DECK41_V2_SEDIM_MIN_LAT = -89.9833333333333333;
67   static const double DECK41_V2_SEDIM_MAX_LAT = 89.9833333333333333;
69   static const double DECK41_V2_SEDIM_MIN_LON = -179.9833333333333333;
71   static const double DECK41_V2_SEDIM_MAX_LON = 179.9833333333333333;
73   static const double DECK41_V2_SEDIM_START_LAT = -89.9833333333333333;
75   static const double DECK41_V2_SEDIM_START_LON = -179.9833333333333333;
77   static const double DECK41_V2_SEDIM_SPACING = 0.0166666666666667;
82   enum DECK41DbType {
83     DECK41_DB_V1_TYPE       = 0,
84     DECK41_DB_V2_TYPE       = 1,
85     DECK41_DB_INVALID_TYPE
86   };
87
88
94   class SedimDeck41CoordDb :  public WossNetcdfDb {
95
96
97     public:
98
103     SedimDeck41CoordDb( const ::std::string& name = DB_NAME_NOT_SET );
104
105 #if defined (WOSS_NETCDF4_SUPPORT)
111     SedimDeck41CoordDb( const ::std::string& name, DECK41DbType db_type );
112 #endif // defined (WOSS_NETCDF4_SUPPORT)
113
114     virtual ~SedimDeck41CoordDb() { }
115
121     Deck41Types getSeaFloorType( const Coord& coordinates ) const ;
122
123
128     virtual bool finalizeConnection();
129
130
135     virtual bool isValid() { return( WossNetcdfDb::isValid()
```

```
136                                     && db_name != DB_NAME_NOT_SET && deck41_db_type !=
      DECK41_DB_INVALID_TYPE ); }
137
138 #if defined (WOSS_NETCDF4_SUPPORT)
143     void setDeck41DbType( DECK41DbType db_type ) { deck41_db_type = db_type; }
144 #endif // defined (WOSS_NETCDF4_SUPPORT)
145
150     DECK41DbType getDeck41DbType()const { return deck41_db_type; }
151
152
153     protected:
154
155
159 #if defined (WOSS_NETCDF4_SUPPORT)
160     netCDF::NcVar main_sedim_var_coord;
161 #else
162     NcVar* main_sedim_var_coord;
163 #endif //
164
168 #if defined (WOSS_NETCDF4_SUPPORT)
169     netCDF::NcVar sec_sedim_var_coord;
170 #else
171     NcVar* sec_sedim_var_coord;
172 #endif // defined (WOSS_NETCDF4_SUPPORT)
173
177 #if defined(WOSS_NETCDF4_SUPPORT)
178     netCDF::NcVar lat_var;
179 #else
180     NcVar* lat_var;
181 #endif // defined(WOSS_NETCDF4_SUPPORT)
182
186 #if defined(WOSS_NETCDF4_SUPPORT)
187     netCDF::NcVar lon_var;
188 #else
189     NcVar* lon_var;
190 #endif // defined(WOSS_NETCDF4_SUPPORT)
191
195     DECK41DbType deck41_db_type;
196
202     int getSedimIndex( const Coord& coords ) const ;
203
204 #if defined (WOSS_NETCDF4_SUPPORT)
210     ::std::pair< int, int > getSedimIndexes( const Coord& coordinates ) const;
211 #endif // defined (WOSS_NETCDF4_SUPPORT)
212   };
213
214
215 }
216
217 #endif // WOSS_NETCDF_SUPPORT
218
219 #endif /* WOSS_SEDIMENT_DECK41_COORD_DB_H */
220
221
222
```

14.74 woss/woss_db/sediment-deck41-db-creator.cpp File Reference

Provides the implementation of woss::SedimDeck41DbCreator class.

14.74.1 Detailed Description

Provides the implementation of woss::SedimDeck41DbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::SedimDeck41DbCreator class

14.75 woss/woss_db/sediment-deck41-db-creator.h File Reference

Provides the interface for woss::SedimDeck41DbCreator class.

Classes

- class woss::SedimDeck41DbCreator

  DbCreator for NetCDF Deck41 Sediment database.

14.75.1 Detailed Description

Provides the interface for woss::SedimDeck41DbCreator class.

Author

Federico Guerra

Provides the interface for the woss::SedimDeck41DbCreator class

14.76 sediment-deck41-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DECK41_DB_CREATOR_H
41 #define WOSS_SEDIMENT_DECK41_DB_CREATOR_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46 #include "woss-db-creator.h"
47 #include "sediment-deck41-coord-db.h"
48
49
50 namespace woss {
51
52
53   class SedimDeck41Db;
54
55
```

```
61   class SedimDeck41DbCreator :  public WossDbCreator {
62
63
64     public:
65
66
70     SedimDeck41DbCreator();
71
72     virtual ~SedimDeck41DbCreator();
73
74
79     virtual WossDb* const createWossDb();
80
81
82     void setDeck41CoordPathName( const ::std::string& name ) { db_coord_name = name; }
83
84     void setDeck41MarsdenPathName( const ::std::string& name ) { db_marsden_name = name; }
85
86     void setDeck41MarsdenOnePathName( const ::std::string& name ) { db_marsden_one_name = name; }
87
88 #if defined (WOSS_NETCDF4_SUPPORT)
89     void setDeck41DbType( DECK41DbType db_type ) { deck41_db_type = db_type; }
90 #endif // defined (WOSS_NETCDF4_SUPPORT)
91
92     ::std::string getDeck41CoordPathName()const { return db_coord_name; }
93
94     ::std::string getDeck41MarsdenPathName()const { return db_marsden_name; }
95
96     ::std::string getDeck41MarsdenOnePathName()const { return db_marsden_one_name; }
97
98     DECK41DbType getDeck41DbType()const { return deck41_db_type; }
99
100
101     protected:
102
103     ::std::string db_coord_name;
104
105     ::std::string db_marsden_name;
106
107     ::std::string db_marsden_one_name;
108
112     DECK41DbType deck41_db_type;
113
119     virtual bool initializeDb( WossDb* woss_db );
120
126     bool initializeSedimDb( SedimDeck41Db* const woss_db );
127   };
128
129
130 }
131
132 #endif // WOSS_NETCDF_SUPPORT
133
134 #endif /* WOSS_SEDIMENT_DECK41_DB_CREATOR_H */
135
```

14.77 woss/woss_db/sediment-deck41-db-logic-control.cpp File Reference

Provides the implementation of woss::Deck41TypeTests class.

14.77.1 Detailed Description

Provides the implementation of woss::Deck41TypeTests class.

Author

Federico Guerra

Provides the implementation of the woss::Deck41TypeTests class

14.78 woss/woss_db/sediment-deck41-db-logic-control.h File Reference

Provides the interface for woss::Deck41TypeTests class.

Classes

- class woss::Deck41TypeTests

  Abstraction layer for database and data manipulation.

14.78.1 Detailed Description

Provides the interface for woss::Deck41TypeTests class.

Author

Federico Guerra

Provides the interface for the woss::Deck41TypeTests class

14.79 sediment-deck41-db-logic-control.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef FLOOR_CONDITIONS_H
41 #define FLOOR_CONDITIONS_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46 #include <sediment-definitions.h>
47
48
49 namespace woss {
50
51
52   static const int DECK41_FLOORTYPE_GRAVEL = 0;
54   static const int DECK41_FLOORTYPE_SAND = 1;
56   static const int DECK41_FLOORTYPE_SILT = 2;
58   static const int DECK41_FLOORTYPE_CLAY = 3;
```

```
60   static const int DECK41_FLOORTYPE_OOZE = 4;
62   static const int DECK41_FLOORTYPE_MUD = 5;
64   static const int DECK41_FLOORTYPE_ROCKS = 6;
66   static const int DECK41_FLOORTYPE_ORGANIC = 7;
68   static const int DECK41_FLOORTYPE_NODULES = 8;
70   static const int DECK41_FLOORTYPE_HARDBOTTOM = 9;
72   static const int DECK41_FLOORTYPE_NODATA = 11;
81   class Deck41TypeTests {
82
83
84     public:
85
86
90     Deck41TypeTests();
91
92     virtual ~Deck41TypeTests();
93
94
100     bool conditionFloorA( const Deck41Types& types ) const ;
101
107     bool conditionFloorB( const Deck41Types& types ) const ;
108
115     bool conditionFloorC( const Deck41Types& types ) const ;
116
123     bool conditionFloorD( const Deck41Types& types ) const ;
124
131     bool conditionFloorE( const Deck41Types& types ) const ;
132
139     bool conditionFloorF( const Deck41Types& types ) const ;
140
147     bool conditionFloorG( const Deck41Types& types ) const ;
148
149
154     bool getConditionA()const { return condition_a; }
155
160     bool getConditionB()const { return condition_b; }
161
166     bool getConditionC()const { return condition_c; }
167
172     bool getConditionD()const { return condition_d; }
173
178     bool getConditionE()const { return condition_e; }
179
184     bool getConditionF()const { return condition_f; }
185
190     bool getConditionG()const { return condition_g; }
191
192
197     void updateAllConditions( const Deck41Types& types );
198
199
200     private:
201
202
203     bool condition_a;
204
205     bool condition_b;
206
207     bool condition_c;
208
209     bool condition_d;
210
211     bool condition_e;
212
213     bool condition_f;
214
215     bool condition_g;
216
217
218   };
219
220
221   //inline functions
223   inline bool Deck41TypeTests::conditionFloorA(const Deck41Types& types)const { // first = return main
      type
224     bool equal = (types.first == types.second);
225     bool first = (types.first == DECK41_FLOORTYPE_GRAVEL) || (types.first == DECK41_FLOORTYPE_SAND) ||
     (types.first == DECK41_FLOORTYPE_SILT)
226                 || (types.first == DECK41_FLOORTYPE_MUD) || (types.first ==
     DECK41_FLOORTYPE_HARDBOTTOM);
227     bool second = (types.second == DECK41_FLOORTYPE_ROCKS) || (types.second == DECK41_FLOORTYPE_ORGANIC)
     ||
228                   (types.second == DECK41_FLOORTYPE_NODULES) || (types.second ==
     DECK41_FLOORTYPE_NODATA) ||
229                   (types.second == DECK41_FLOORTYPE_HARDBOTTOM);
230
231     return ( (equal && first ) || (first && second));
```

```
232   }
233
234
235   inline bool Deck41TypeTests::conditionFloorB(const Deck41Types& types)const { // second = return
      secondary type
236     bool first = (types.first == DECK41_FLOORTYPE_NODATA);
237     bool second = (types.second == DECK41_FLOORTYPE_GRAVEL) || (types.second == DECK41_FLOORTYPE_SAND)
      ||
238                   (types.second == DECK41_FLOORTYPE_SILT) || (types.second == DECK41_FLOORTYPE_MUD);
239
240     return (first && second);
241   }
242
243
244   inline bool Deck41TypeTests::conditionFloorC(const Deck41Types& types)const { // first* = try other
      coords, if fails get main type
245     bool first = (types.first == DECK41_FLOORTYPE_CLAY) || (types.first == DECK41_FLOORTYPE_OOZE) ||
      (types.first == DECK41_FLOORTYPE_ORGANIC)
246               || (types.first == DECK41_FLOORTYPE_ROCKS) || (types.first == DECK41_FLOORTYPE_NODULES);
247     bool second = (types.second == DECK41_FLOORTYPE_ROCKS) || (types.second == DECK41_FLOORTYPE_ORGANIC)
      ||
248                   (types.second == DECK41_FLOORTYPE_NODULES) || (types.second ==
      DECK41_FLOORTYPE_NODATA) ||
249                   (types.second == DECK41_FLOORTYPE_HARDBOTTOM);
250
251     return (first && second);
252   }
253
254
255   inline bool Deck41TypeTests::conditionFloorD(const Deck41Types& types)const { // second* = try other
      coords, if fails get secondary type
256     if ( types.first == DECK41_FLOORTYPE_ORGANIC ) return true;
257     else {
258         bool first = (types.first == DECK41_FLOORTYPE_ROCKS) || (types.first ==
      DECK41_FLOORTYPE_NODULES) ||
259                      (types.first == DECK41_FLOORTYPE_NODATA) ;
260         bool second = (types.second == DECK41_FLOORTYPE_GRAVEL) || (types.second ==
      DECK41_FLOORTYPE_SAND) ||
261                       (types.second == DECK41_FLOORTYPE_SILT) || (types.second == DECK41_FLOORTYPE_MUD);
262
263     return (first && second);
264     }
265   }
266
267   inline bool Deck41TypeTests::conditionFloorE(const Deck41Types& types)const { // 65%/35% = weighted
      avg
268     bool nequal = types.first != types.second;
269     bool first = (types.first == DECK41_FLOORTYPE_GRAVEL) || (types.first == DECK41_FLOORTYPE_SAND) ||
      (types.first == DECK41_FLOORTYPE_SILT)
270                 || (types.first == DECK41_FLOORTYPE_MUD);
271     bool second = first || (types.second == DECK41_FLOORTYPE_OOZE) || (types.second ==
      DECK41_FLOORTYPE_CLAY);
272     bool special = (types.first == DECK41_FLOORTYPE_OOZE) && (types.second == DECK41_FLOORTYPE_CLAY);
273
274     return ( (first && second && nequal) || special );
275   }
276
277
278   inline bool Deck41TypeTests::conditionFloorF(const Deck41Types& types)const { // 40%/60% = weighted
      avg
279     bool nequal = types.first != types.second;
280     bool first = (types.first == DECK41_FLOORTYPE_CLAY) || (types.first == DECK41_FLOORTYPE_OOZE);
281     bool second = (types.second == DECK41_FLOORTYPE_GRAVEL) || (types.second == DECK41_FLOORTYPE_SAND)
      || (types.second == DECK41_FLOORTYPE_SILT)
282                 || (types.second == DECK41_FLOORTYPE_MUD) || (types.second == DECK41_FLOORTYPE_OOZE);
283
284     return (first && second && nequal);
285   }
286
287
288   inline bool Deck41TypeTests::conditionFloorG(const Deck41Types& types)const { // exit* = try other
      coords, if fails exit program
289     return ( (types.first == DECK41_FLOORTYPE_NODATA) && (types.second == DECK41_FLOORTYPE_NODATA));
290   }
291
292
293   inline void Deck41TypeTests::updateAllConditions(const Deck41Types& types) {
294     condition_a = conditionFloorA(types); // first = return main type
295     condition_b = conditionFloorB(types); // second = return secondary type
296     condition_c = conditionFloorC(types); // first* = try other coords, if fails get main type
297     condition_d = conditionFloorD(types); // second* = try other coords, if fails get secondary type
298     condition_e = conditionFloorE(types); // 65%/35% = weighted avg
299     condition_f = conditionFloorF(types); // 40%/60% = weighted avg
300     condition_g = conditionFloorG(types);
301   }
302
303
```

```
304 }
305
306 #endif // WOSS_NETCDF_SUPPORT
307
308 #endif /* FLOOR_CONDITIONS_H */
309
```

14.80 woss/woss_db/sediment-deck41-db.cpp File Reference

Provides the implementation of woss::SedimDeck41Db class.

14.80.1 Detailed Description

Provides the implementation of woss::SedimDeck41Db class.

Author

Federico Guerra

Provides the implementation of the woss::SedimDeck41Db class

14.81 woss/woss_db/sediment-deck41-db.h File Reference

Provides the interface for woss::SedimDeck41Db class.

Classes

- class woss::SedimDeck41Db
  WossDb for NetCDF DECK41 Sediment database.

Typedefs

- typedef ::std::map< int, int > woss::FrequencyMap
- typedef FrequencyMap::const_iterator **woss::FMCIter**
- typedef FrequencyMap::iterator **woss::FMIter**
- typedef FrequencyMap::reverse_iterator **woss::FMRIter**
- typedef FrequencyMap::const_reverse_iterator **woss::FMCRIter**
- typedef ::std::map< char, double > woss::SedimWeightMap
- typedef SedimWeightMap::iterator **woss::SWIter**
- typedef SedimWeightMap::reverse_iterator **woss::SWRIter**

14.81.1 Detailed Description

Provides the interface for woss::SedimDeck41Db class.

Author

Federico Guerra

Provides the interface for the woss::SedimDeck41Db class

14.81.2 Typedef Documentation

14.81.2.1 FrequencyMap `typedef ::std::map<int, int> woss::FrequencyMap`

Map that links a DECK41 integer type to the number of times it has appeared in a database query result

See also

deck41-db-state-logic-constrol.h

14.81.2.2 SedimWeightMap `typedef ::std::map< char, double > woss::SedimWeightMap`

Map that links a condition of Deck41TypeTests to its weight for calculating weighted average value

See also

Deck41TypeTests

14.82 sediment-deck41-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DECK41_DB_H
41 #define WOSS_SEDIMENT_DECK41_DB_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46 #include <map>
47 #include "sediment-deck41-db-logic-control.h"
48 #include "sediment-deck41-coord-db.h"
49 #include "sediment-deck41-marsden-db.h"
50 #include "sediment-deck41-marsden-one-db.h"
51
```

```
52
53 namespace woss {
54
55
60   typedef ::std::map<int, int> FrequencyMap;
61   typedef FrequencyMap::const_iterator FMCIter;
62   typedef FrequencyMap::iterator FMIter;
63   typedef FrequencyMap::reverse_iterator FMRIter;
64   typedef FrequencyMap::const_reverse_iterator FMCRIter;
65
66
71   typedef ::std::map< char, double > SedimWeightMap;
72   typedef SedimWeightMap::iterator SWIter;
73   typedef SedimWeightMap::reverse_iterator SWRIter;
74
75
82   class SedimDeck41Db :  public WossDb, public WossSedimentDb {
83
84
85     friend class SedimDeck41DbCreator;
86
87
88     public:
89
90
95     SedimDeck41Db( const ::std::string& name );
96
97     virtual ~SedimDeck41Db() { }
98
99
103     virtual bool openConnection() { return false; }
104
108     virtual bool finalizeConnection() { return false; }
109
110
115     virtual bool closeConnection();
116
117
124     virtual bool insertValue( const Coord& coordinates, const Sediment& sediment_value );
125
126
134     virtual Sediment* getValue( const CoordZ& coordz ) const ;
135
143     virtual Sediment* getValue( const CoordZVector& coordz_vector ) const ;
144
145
146     protected:
147
148
152     SedimDeck41CoordDb sediment_coord_db;
153
157     SedimDeck41MarsdenDb sediment_marsden_db;
158
162     SedimDeck41MarsdenOneDb sediment_marsden_one_db;
163
164
168     mutable Deck41TypeTests curr_tests_state;
169
173     mutable Deck41TypeTests prev_tests_state;
174
175
179     static SedimWeightMap sediment_weight_map;
180
181
186     static SedimWeightMap initSedimWeightMap();
187
188
194     double calculateAvgDepth( const CoordZVector& coordz_vector ) const;
195
196
202     Deck41Types getDeck41TypesFromCoords( const CoordZVector& coordz_vector ) const;
203
209     Deck41Types getDeck41TypesFromMarsdenCoords( const CoordZVector& coordz_vector ) const ;
210
216     Deck41Types getDeck41TypesFromMarsdenSquare( const CoordZVector& coordz_vector ) const ;
217
218
226     Deck41Types calculateDeck41Types( const CoordZVector& coordz_vector ) const;
227
228
235     Sediment* calculateSediment( const Deck41Types& floor_types, double avg_depth ) const;
236
244     Sediment* createSediment( int deck41_type, double depth ) const;
245
246
252     int getMaxAppereanceFrequencyValue( const FrequencyMap& frequency_map ) const;
253
```

```
254
261     bool doTestA( const Deck41TypeTests& test ) const ;
262
269     bool doTestB( const Deck41TypeTests& test ) const ;
270
277     bool doTestC( const Deck41TypeTests& test ) const ;
278
279
280   };
281
282
283   // inline functions
285
286   inline bool SedimDeck41Db::doTestA(const Deck41TypeTests& floor)const {
287     return  ((floor.getConditionA() == true) || (floor.getConditionB() == true) ||
      (floor.getConditionE() == true) ||
288              (floor.getConditionF() == true));
289   }
290
291
292   inline bool SedimDeck41Db::doTestB(const Deck41TypeTests& floor)const {
293     return  ((floor.getConditionC() == true) || (floor.getConditionD() == true) ||
      (floor.getConditionG() == true) );
294   }
295
296
297   inline bool SedimDeck41Db::doTestC(const Deck41TypeTests& floor)const {
298     return ((floor.getConditionC() == true) || (floor.getConditionD() == true));
299   }
300
301
302 }
303
304 #endif // WOSS_NETCDF_SUPPORT
305
306 #endif /* WOSS_SEDIMENT_DECK41_DB_H */
307
308
309
```

14.83 woss/woss_db/sediment-deck41-marsden-db.cpp File Reference

Provides the implementation of woss::SedimDeck41MarsdenDb class.

14.83.1 Detailed Description

Provides the implementation of woss::SedimDeck41MarsdenDb class.

Author

Federico Guerra

Provides the implementation of the woss::SedimDeck41MarsdenDb class

14.84 woss/woss_db/sediment-deck41-marsden-db.h File Reference

Provides the interface for woss::SedimDeck41MarsdenDb class.

Classes

- class woss::SedimDeck41MarsdenDb

  WossDb for custom made NetCDF marsden square DECK41 Sediment database.

14.84.1 Detailed Description

Provides the interface for woss::SedimDeck41MarsdenDb class.

Author

Federico Guerra

Provides the interface for the woss::SedimDeck41MarsdenDb class

14.85 sediment-deck41-marsden-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DECK41_MARSDEN_DB_H
41 #define WOSS_SEDIMENT_DECK41_MARSDEN_DB_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46
47 #include <sediment-definitions.h>
48 #include "woss-db.h"
49 #include "sediment-deck41-coord-db.h"
50 #if defined (WOSS_NETCDF4_SUPPORT)
51 #include <ncVar.h>
52 #endif // defined (WOSS_NETCDF4_SUPPORT)
53
54 namespace woss {
55
56
62   class SedimDeck41MarsdenDb :  public WossNetcdfDb {
63
64
65     public:
66
71     SedimDeck41MarsdenDb( const ::std::string& name = DB_NAME_NOT_SET );
72
73 #if defined (WOSS_NETCDF4_SUPPORT)
79     SedimDeck41MarsdenDb( const ::std::string& name, DECK41DbType db_type );
80 #endif // defined (WOSS_NETCDF4_SUPPORT)
81
82     virtual ~SedimDeck41MarsdenDb() { }
83
84
90     Deck41Types getSeaFloorType( const Marsden& marsden_square ) const;
91
92
97     virtual bool isValid() { return( WossNetcdfDb::isValid()
```

```
98                                      && db_name != DB_NAME_NOT_SET
99                                      && deck41_db_type != DECK41_DB_INVALID_TYPE ); }
100
101
106     virtual bool finalizeConnection();
107
108 #if defined (WOSS_NETCDF4_SUPPORT)
113     void setDeck41DbType( DECK41DbType db_type ) { deck41_db_type = db_type; }
114 #endif // defined (WOSS_NETCDF4_SUPPORT)
115
120     DECK41DbType getDeck41DbType()const { return deck41_db_type; }
121
122
123     protected:
124
125
129 #if defined (WOSS_NETCDF4_SUPPORT)
130     netCDF::NcVar main_sedim_var_marsden;
131 #else
132     NcVar* main_sedim_var_marsden;
133 #endif // defined (WOSS_NETCDF4_SUPPORT)
134
138 #if defined (WOSS_NETCDF4_SUPPORT)
139     netCDF::NcVar sec_sedim_var_marsden;
140 #else
141     NcVar* sec_sedim_var_marsden;
142 #endif // defined (WOSS_NETCDF4_SUPPORT)
146 #if defined(WOSS_NETCDF4_SUPPORT)
147     netCDF::NcVar marsden_square_var;
148 #else
149     NcVar* marsden_square_var;
150 #endif // defined(WOSS_NETCDF4_SUPPORT)
151
155     DECK41DbType deck41_db_type;
156
157   };
158
159
160 }
161
162 #endif // WOSS_NETCDF_SUPPORT
163
164 #endif /* WOSS_SEDIMENT_DECK41_MARSDEN_DB_H */
165
166
167
```

14.86 woss/woss_db/sediment-deck41-marsden-one-db.cpp File Reference

Provides the implementation of woss::SedimDeck41MarsdenOneDb class.

14.86.1 Detailed Description

Provides the implementation of woss::SedimDeck41MarsdenOneDb class.

Author

Federico Guerra

Provides the implementation of the woss::SedimDeck41MarsdenOneDb class

14.87 woss/woss_db/sediment-deck41-marsden-one-db.h File Reference

Provides the interface for woss::SedimDeck41MarsdenOneDb class.

Classes

- class woss::SedimDeck41MarsdenOneDb

  WossDb for custom made NetCDF marsden coordinates DECK41 Sediment database.

14.87.1 Detailed Description

Provides the interface for woss::SedimDeck41MarsdenOneDb class.

Author

Federico Guerra

Provides the interface for the woss::SedimDeck41MarsdenOneDb class

14.88 sediment-deck41-marsden-one-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DECK41_MARSDEN_ONE_DB_H
41 #define WOSS_SEDIMENT_DECK41_MARSDEN_ONE_DB_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46
47 #include <sediment-definitions.h>
48 #include "woss-db.h"
49 #include "sediment-deck41-coord-db.h"
50 #if defined (WOSS_NETCDF4_SUPPORT)
51 #include <ncVar.h>
52 #endif // defined (WOSS_NETCDF4_SUPPORT)
53
54 namespace woss {
55
56
62   class SedimDeck41MarsdenOneDb :  public WossNetcdfDb {
63
64
65     public:
66
72     SedimDeck41MarsdenOneDb( const ::std::string& name = DB_NAME_NOT_SET );
73
74 #if defined (WOSS_NETCDF4_SUPPORT)
```

```
80      SedimDeck41MarsdenOneDb( const ::std::string& name, DECK41DbType db_type );
81  #endif // defined (WOSS_NETCDF4_SUPPORT)
82
83      virtual ~SedimDeck41MarsdenOneDb() { }
84
85
91      Deck41Types getSeaFloorType( const MarsdenCoord& marsden_coord ) const;
92
93
98      virtual bool isValid() { return( WossNetcdfDb::isValid()
99                                       && db_name != DB_NAME_NOT_SET
100                                      && deck41_db_type != DECK41_DB_INVALID_TYPE ); }
101
102
107      virtual bool finalizeConnection();
108
109 #if defined (WOSS_NETCDF4_SUPPORT)
114     void setDeck41DbType( DECK41DbType db_type ) { deck41_db_type = db_type; }
115 #endif // defined (WOSS_NETCDF4_SUPPORT)
116
121     DECK41DbType getDeck41DbType()const { return deck41_db_type; }
122
123
124     protected:
125
126
130 #if defined (WOSS_NETCDF4_SUPPORT)
131     netCDF::NcVar main_sedim_var_marsden_one;
132 #else
133     NcVar* main_sedim_var_marsden_one;
134 #endif // defined (WOSS_NETCDF4_SUPPORT)
135
139 #if defined (WOSS_NETCDF4_SUPPORT)
140     netCDF::NcVar sec_sedim_var_marsden_one;
141 #else
142     NcVar* sec_sedim_var_marsden_one;
143 #endif // defined (WOSS_NETCDF4_SUPPORT)
147 #if defined(WOSS_NETCDF4_SUPPORT)
148     netCDF::NcVar marsden_square_var;
149 #else
150     NcVar* marsden_square_var;
151 #endif // defined(WOSS_NETCDF4_SUPPORT)
155 #if defined(WOSS_NETCDF4_SUPPORT)
156     netCDF::NcVar marsden_one_square_var;
157 #else
158     NcVar* marsden_one_square_var;
159 #endif // defined(WOSS_NETCDF4_SUPPORT)
160
164     DECK41DbType deck41_db_type;
165
166   };
167
168
169 }
170
171 #endif // WOSS_NETCDF_SUPPORT
172
173 #endif /* WOSS_SEDIMENT_DECK41_MARSDEN_ONE_DB_H */
174
175
176
```

14.89 woss/woss_db/ssp-woa2005-db-creator.cpp File Reference

Provides the implementation of woss::SspWoa2005DbCreator class.

14.89.1 Detailed Description

Provides the implementation of woss::SspWoa2005DbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::SspWoa2005DbCreator class

14.90 woss/woss_db/ssp-woa2005-db-creator.h File Reference

Provides the interface for woss::SspWoa2005DbCreator class.

Classes

- class woss::SspWoa2005DbCreator

  WossDbCreator for the custom made NetCDF WOA2005 SSP database.

14.90.1 Detailed Description

Provides the interface for woss::SspWoa2005DbCreator class.

Author

Federico Guerra

Provides the interface for the woss::SspWoa2005DbCreator class

14.91 ssp-woa2005-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SSP_NETCDF_DB_CREATOR_H
41 #define WOSS_SSP_NETCDF_DB_CREATOR_H
42
43
44 #ifdef WOSS_NETCDF_SUPPORT
45
46
47 #include "woss-db-creator.h"
48 #include "ssp-woa2005-db.h"
49
50 namespace woss {
51
52
59   class SspWoa2005DbCreator :  public WossDbCreator {
60
61
```

```
62     public:
63
64
68     SspWoa2005DbCreator();
69
70 #if defined (WOSS_NETCDF4_SUPPORT)
75     SspWoa2005DbCreator( WOADbType db_type );
76 #endif // defined (WOSS_NETCDF4_SUPPORT)
77
78     virtual ~SspWoa2005DbCreator();
79
80
85     virtual WossDb* const createWossDb();
86
87
92     WOADbType getWoaDbType()const { return woa_db_type; }
93
94 #if defined (WOSS_NETCDF4_SUPPORT)
99     SspWoa2005DbCreator& setWoaDbType(WOADbType type) { woa_db_type = type; return *this; }
100 #endif // defined (WOSS_NETCDF4_SUPPORT)
101
102     protected:
103
104
110     virtual bool initializeDb( WossDb* const woss_db );
111
112     WOADbType woa_db_type;
113   };
114
115 }
116
117 #endif // WOSS_NETCDF_SUPPORT
118
119
120 #endif /* WOSS_SSP_NETCDF_DB_CREATOR_H */
121
```

14.92 woss/woss_db/ssp-woa2005-db.cpp File Reference

Provides the implementation of woss::SspWoa2005Db class.

14.92.1 Detailed Description

Provides the implementation of woss::SspWoa2005Db class.

Author

Federico Guerra

Provides the implementation of the woss::SspWoa2005Db class

14.93 woss/woss_db/ssp-woa2005-db.h File Reference

Provides the interface for woss::SspWoa2005Db class.

Classes

- class woss::SspWoa2005Db

  WossDb for the custom made NetCDF WOA2005 SSP database.

Typedefs

- typedef ::std::pair< int, int > woss::SSPIndexes

Enumerations

- enum woss::WOADbType { woss::WOA_DB_TYPE_2005 = 0 , woss::WOA_DB_TYPE_2013 = 1 , woss::WOA_DB_TYPE_INVALID }

14.93.1 Detailed Description

Provides the interface for woss::SspWoa2005Db class.

Author

Federico Guerra

Provides the interface for the woss::SspWoa2005Db class

14.93.2 Typedef Documentation

14.93.2.1 SSPIndexes `typedef ::std::pair< int, int > woss::SSPIndexes`

Pair representing latitute and longitude index for NetCDF value access

14.93.3 Enumeration Type Documentation

14.93.3.1 WOADbType `enum woss::WOADbType`

Enumerator

WOA_DB_TYPE_2005	2005 and 2009 NetCDF Db type, 1 degree resolution
WOA_DB_TYPE_2013	2013, 2001 and 2018 NetCDF4 Db type, 0.25 degree resolution
WOA_DB_TYPE_INVALID	Must always be the last.

14.94 ssp-woa2005-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
```

```
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
30 #ifndef WOSS_SSP_NETCDF_DB_H
31 #define WOSS_SSP_NETCDF_DB_H
32
33
44 #ifdef WOSS_NETCDF_SUPPORT
45
46
47 #include <ssp-definitions.h>
48 #include "woss-db.h"
49 #if defined (WOSS_NETCDF4_SUPPORT)
50 #include <ncVar.h>
51 #endif // defined (WOSS_NETCDF4_SUPPORT)
52
53 namespace woss {
54
55   static const int SSP_WOA2005_STD_NLAT = 180;
57   static const int SSP_WOA2005_STD_NLON = 360;
59   static const double SSP_WOA2005_STD_SPACING = 1.0;
62   static const double SSP_WOA2005_STD_MIN_LAT = -89.5;
64   static const double SSP_WOA2005_STD_MAX_LAT = 89.5;
67   static const double SSP_WOA2005_STD_MIN_LON = -179.5;
69   static const double SSP_WOA2005_STD_MAX_LON = 179.5;
72   static const double SSP_WOA2005_STD_START_LAT = 89.5;
74   static const double SSP_WOA2005_STD_START_LON = -179.5;
77   static const int SSP_WOA2013_STD_NLAT = 720;
79   static const int SSP_WOA2013_STD_NLON = 1440;
81   static const double SSP_WOA2013_STD_SPACING = 0.25;
84   static const double SSP_WOA2013_STD_MIN_LAT = -89.875;
86   static const double SSP_WOA2013_STD_MAX_LAT = 89.875;
89   static const double SSP_WOA2013_STD_MIN_LON = -179.875;
91   static const double SSP_WOA2013_STD_MAX_LON = 179.875;
94   static const double SSP_WOA2013_STD_START_LAT = -89.875;
96   static const double SSP_WOA2013_STD_START_LON = -179.875;
98   static const int SSP_STD_NDEPTH = 33;
103   static const short ssp_std_depths[SSP_STD_NDEPTH] = { 0, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250,
      300,
104                                                         400, 500, 600, 700, 800, 900, 1000, 1100, 1200,
      1300, 1400, 1500,
105                                                         1750, 2000, 2500, 3000, 3500, 4000, 4500, 5000,
      5500 };
106
110   typedef ::std::pair< int, int > SSPIndexes;
111
112   enum WOADbType {
113     WOA_DB_TYPE_2005 = 0,
114     WOA_DB_TYPE_2013 = 1,
115     WOA_DB_TYPE_INVALID
116   };
117
123   class SspWoa2005Db :  public WossNetcdfDb, public WossSSPDb {
124
125
126     public:
127
128
133     SspWoa2005Db( const ::std::string& name );
134
135 #if defined (WOSS_NETCDF4_SUPPORT)
141     SspWoa2005Db( const ::std::string& name, WOADbType db_type );
142 #endif // defined (WOSS_NETCDF4_SUPPORT)
```

```
143
144     virtual ~SspWoa2005Db() { }
145
146
151     virtual bool finalizeConnection();
152
153
161     virtual bool insertValue( const Coord& coordinates, const Time& time_value, const SSP& ssp_value );
162
163
172     virtual SSP* getValue( const Coord& coordinates, const Time& time, long double ssp_depth_precision )
        const ;
173
178     WOADbType getWoaDbType()const { return woa_db_type; }
179
180
181     protected:
182
186     WOADbType woa_db_type;
187
191 #if defined (WOSS_NETCDF4_SUPPORT)
192     netCDF::NcVar ssp_var;
193 #else
194     NcVar* ssp_var;
195 #endif // defined (WOSS_NETCDF4_SUPPORT)
196
197 #if defined (WOSS_NETCDF4_SUPPORT)
201     netCDF::NcVar lat_var;
205     netCDF::NcVar lon_var;
206 #endif // defined (WOSS_NETCDF4_SUPPORT)
207
213     SSPIndexes getSSPIndexes( const Coord& coordinates ) const;
214
215 #if defined (WOSS_NETCDF4_SUPPORT)
222     void getSSPValue( const Coord& coordinates, const SSPIndexes& indexes, double ssp_values[] ) const;
223 #else
229     void getSSPValue( const SSPIndexes& indexes, double ssp_values[] ) const;
230 #endif // defined (WOSS_NETCDF4_SUPPORT)
231
232
233   };
234
235
236 }
237
238
239 #endif // WOSS_NETCDF_SUPPORT
240
241 #endif /* WOSS_SSP_NETCDF_DB_H */
242
243
244
```

14.95 woss/woss_db/woss-db-creator.cpp File Reference

Provides the implementation of woss::WossDbCreator class.

14.95.1 Detailed Description

Provides the implementation of woss::WossDbCreator class.

Author

Federico Guerra

Provides the implementation of the woss::WossDbCreator class

14.96 woss/woss_db/woss-db-creator.h File Reference

Provides the interface for woss::WossDbCreator class.

Classes

- class woss::WossDbCreator

  Abstract class that provides the interface of database creator (Factory object)

14.96.1 Detailed Description

Provides the interface for woss::WossDbCreator class.

Author

Federico Guerra

Provides the interface for the woss::WossDbCreator class

14.97 woss-db-creator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DB_CREATOR_H
41 #define WOSS_DB_CREATOR_H
42
43
44 #include <string>
45
46
47 namespace woss {
48
49
50   class WossDb;
51
52
64   class WossDbCreator {
65
66
67     public:
68
69
73     WossDbCreator();
74
75     virtual ~WossDbCreator();
76
77
83     virtual WossDb* const createWossDb() = 0;
```

```
84
85
86      WossDbCreator& setDebug( bool flag ) { debug = flag; return *this; }
87
88      WossDbCreator& setWossDebug( bool flag ) { woss_db_debug = flag; return *this; }
89
90
91      WossDbCreator& setDbPathName( const ::std::string& name ) { pathname = name; return *this; }
92
93
94      bool isUsingDebug()const { return debug; }
95
96      bool isUsingWossDbDebug()const { return woss_db_debug; }
97
98
99      ::std::string getDbPathName()const { return pathname; }
100
101
102      protected:
103
104
108      bool debug;
109
113      bool woss_db_debug;
114
115
119      ::std::string pathname;
120
121
127      virtual bool initializeDb( WossDb* const woss_db ) = 0;
128
129
130    };
131
132
133 }
134
135
136 #endif /* WOSS_DB_CREATOR_H */
137
138
```

14.98 woss/woss_db/woss-db-custom-data-container.h File Reference

Provides the interface for woss::CustomDataContainer class.

Classes

- class woss::CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >
  Class for managing custom db data.
- class woss::CustomDataContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >
  CustomDataContainer template partial specialization for pointers.
- class woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >
  Class for managing custom db data.
- class woss::CustomDataTimeContainer< T, MidFunctor, InFunctor, Data *, OutComp, MidComp, InComp >
  Class for managing custom db data.

14.98.1 Detailed Description

Provides the interface for woss::CustomDataContainer class.

Author

Federico Guerra

Provides the interface for the woss::CustomDataContainer template class

14.99 woss-db-custom-data-container.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DB_CUSTOM_DATA_CONTAINER_H
41 #define WOSS_DB_CUSTOM_DATA_CONTAINER_H
42
43
44 #include <map>
45 #include <iostream>
46 #include <time-definitions.h>
47 #include <complex>
48
49 namespace woss {
50
51
59   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp = ::std::less<T>,
       class MidComp = ::std::less<double>, class InComp = ::std::less<double> >
60   class CustomDataContainer {
61
62
63     protected:
64
65
69     typedef typename ::std::map< double, Data, InComp > InnerData;
70     typedef typename InnerData::iterator CDCInnerIt;
71     typedef typename InnerData::reverse_iterator CDCInnerRIt;
72     typedef typename InnerData::const_iterator CDCInnerCIt;
73     typedef typename InnerData::const_reverse_iterator CDCInnerCRIt;
74
78     typedef typename ::std::map< double, InnerData, MidComp > MediumData;
79     typedef typename MediumData::iterator CDCMediumIt;
80     typedef typename MediumData::const_iterator CDCMediumCIt;
81     typedef typename MediumData::reverse_iterator CDCMediumRIt;
82     typedef typename MediumData::const_reverse_iterator CDCMediumCRIt;
83
87     typedef ::std::map< T, MediumData, OutComp > CustomContainer;
88     typedef typename CustomContainer::iterator CDCIt;
89     typedef typename CustomContainer::reverse_iterator CDCRIt;
90     typedef typename CustomContainer::const_iterator CDCCIt;
91     typedef typename CustomContainer::const_reverse_iterator CDCCRIt;
92
93
94     public:
95
96 #if __cplusplus >= 201103L // C++11 or later
97     static constexpr double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
98
99     static constexpr double DB_CDATA_ALL_INNER_KEYS = -10.0;
100 #else
101     static const double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
102
103     static const double DB_CDATA_ALL_INNER_KEYS = -10.0;
104 #endif // __cplusplus == 201103L
105
106     static const T DB_CDATA_ALL_OUTER_KEYS;
107
108
```

```
112     CustomDataContainer() :  debug(false), data_map() { }
113
117     ~CustomDataContainer() { clear(); }
118
119
125     MediumData& operator[] ( const T& key ) { return data_map[key]; }
126
127
132     bool empty()const { return data_map.empty(); }
133
134
139     int size()const { return data_map.size(); }
140
141
152     const Data* get( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r
    = DB_CDATA_ALL_INNER_KEYS ) const;
153
160     const Data* get( const T& tx, const T& rx ) const;
161
162
174     bool insert( const Data& data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b =
    DB_CDATA_ALL_MEDIUM_KEYS, double r = DB_CDATA_ALL_INNER_KEYS );
175
187     void replace( const Data& data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b =
    DB_CDATA_ALL_MEDIUM_KEYS, double r = DB_CDATA_ALL_INNER_KEYS );
188
189
200     void erase( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
    DB_CDATA_ALL_INNER_KEYS );
201
202
203 //      void normalizeMap( const T& t );
204
205
209     void clear();
210
211
216     void setDebug( bool flag ) { debug = flag; }
217
218
223     bool usingDebug() { return debug; }
224
225
226     protected:
227
228
232     bool debug;
233
234
238     CustomContainer data_map;
239
240
251     const Data* find( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double
    r = DB_CDATA_ALL_INNER_KEYS ) const;
252
253
254   };
255
256
257   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
258   const T CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
    >::DB_CDATA_ALL_OUTER_KEYS = T();
259
260
261   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
262   const Data* CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get(
    const T& t, double b, double r )const {
263     if ( data_map.empty() ) return NULL;
264
265     if ( debug ) ::std::cout << "CustomDataContainer::get() t = " << t << "; b = " << b << "; r = " << r <<
    ::std::endl;
266
267     const Data* ptr = find();
268     if ( ptr != NULL ) return ptr;
269
270     CDCCIt it = data_map.find( t );
271     if ( it == data_map.end() ) return NULL;
272
273     CDCMediumCIt it2 = it->second.lower_bound( b );
274     CDCMediumCRIt rit2 = it->second.rbegin();
275
276     CDCInnerCIt it3;
277     CDCInnerCRIt rit3;
278
279     if ( it2 == it->second.end() ) {
```

```
280       if( rit2 == it->second.rend() ) return NULL;
281       it3 = rit2->second.lower_bound( r );
282       rit3 = rit2->second.rbegin();
283       if ( it3 != rit2->second.end() ) return &it3->second;
284       if ( rit3 != rit2->second.rend() ) return &rit3->second;
285       return NULL;
286     }
287     else {
288       it3 = it2->second.lower_bound( r );
289       rit3 = it2->second.rbegin();
290       if ( it3 != it2->second.end() ) return &it3->second;
291       if ( rit3 != it2->second.rend() ) return &rit3->second;
292       return NULL;
293     }
294     if ( it3 == it2->second.end() ) return NULL;
295     return &(it3->second);
296   }
297
298
299   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
300   const Data* CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get(
      const T& tx, const T& rx )const {
301     MidFunctor mid_funct;
302     InFunctor in_funct;
303
304     double curr_b;
305     double curr_r;
306     double delta_b;
307     //double delta_r;
308     const Data* ret_val = NULL;
309
310     double curr_dist;
311     double min_dist = INFINITY;
312
313     if ( data_map.empty() == true ) {
314       if ( debug ) ::std::cout << "CustomDataContainer::get() data_map is empty " << ::std::endl;
315
316       return ret_val;
317     }
318
319     for ( CDCCIt it = data_map.begin(); it != data_map.end(); it++ ) {
320
321       if ( debug ) ::std::cout << "CustomDataContainer::get() start T = " << it->first << "; end T = " << rx
      << ::std::endl;
322
323       if ( it->first == DB_CDATA_ALL_OUTER_KEYS ) {
324         if ( debug ) ::std::cout << "CustomDataContainer::get() overriding start T = " << tx << "; end T =
      " << rx << ::std::endl;
325
326         curr_b = mid_funct(tx,rx);
327         curr_r = in_funct(tx,rx);
328       }
329       else {
330         curr_b = mid_funct(it->first,rx);
331         curr_r = in_funct(it->first,rx);
332       }
333
334       if ( debug ) ::std::cout << "CustomDataContainer::get() curr bearing = " << curr_b * 180.0 / M_PI
335                                << "; curr range = " << curr_r << ::std::endl;
336
337       CDCMediumCIt itb = it->second.begin();
338       if ( itb->first == DB_CDATA_ALL_MEDIUM_KEYS ) delta_b = 0;
339       else {
340         itb = it->second.lower_bound( curr_b );
341         if ( itb == it->second.end() ) itb = (++(it->second.rbegin())).base();
342
343         delta_b = curr_b - itb->first;
344         if (delta_b < 0.0) delta_b = -delta_b;
345         if (delta_b > M_PI) delta_b = 2.0*M_PI - delta_b ;
346       }
347
348       double ort_dist = curr_r * sin(delta_b);
349       double ort_projection = ::std::sqrt( curr_r*curr_r - ort_dist*ort_dist );
350
351       if ( debug ) ::std::cout << "CustomDataContainer::get() nearest bearing = " << itb->first * 180.0 /
      M_PI
352                                << "; diff bearing = " << delta_b * 180.0 / M_PI << "; orthog distance = " <<
      ort_dist
353                                << "; orthog range projection = " << ort_projection << ::std::endl;
354
355       CDCInnerCIt itr = itb->second.begin();
356       if ( itr->first == DB_CDATA_ALL_INNER_KEYS ) curr_dist = ort_dist;
357       else {
358         itr = itb->second.lower_bound( ort_projection );
359         if ( itr == itb->second.begin() || itr == itb->second.end() || itr->first == ort_projection ) {
360           if ( itr == itb->second.end() ) itr = (++(itb->second.rbegin())).base();
```

```
361           double adj_distance = ::std::abs( ort_projection - itr->first );
362           curr_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
363         }
364         else {
365           double adj_distance = ::std::abs( ort_projection - itr->first );
366           double first_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
367
368           if(debug) ::std::cout << "CustomDataContainer::get() first try, range = " << itr->first
369                                 << "; dist = " << first_dist << ::std::endl;
370
371           itr--;
372           adj_distance = ::std::abs( ort_projection - itr->first );
373           double before_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
374
375           if (debug) ::std::cout << "CustomDataContainer::get() second try, range = " << itr->first
376                                  << "; dist = " << before_dist << ::std::endl;
377
378           curr_dist = ::std::min( first_dist, before_dist );
379           if ( curr_dist == first_dist ) itr++;
380         }
381       }
382       if ( debug ) ::std::cout << "CustomDataContainer::get() nearest range = " << itr->first << ";
    distance = " << curr_dist
383                                << "; min distance = " << min_dist << ::std::endl;
384
385       if ( curr_dist < min_dist ) {
386         min_dist = curr_dist;
387         ret_val = &(itr->second);
388         if ( curr_dist == 0 ) break;
389       }
390
391     }
392
393    if ( debug && ret_val != NULL ) ::std::cout << "CustomDataContainer::get() ret value " << *ret_val <<
    ::std::endl;
394
395     return ret_val;
396   }
397
398
399   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
400   const Data* CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find(
    const T& t, double b, double r )const {
401
402     if (this->debug) ::std::cout << "CustomDataContainer::find() " << t = " << t << "; b = " << b << "; r = " <<
    r << ::std::endl;
403
404     CDCCIt it = data_map.find( t );
405     if ( it == data_map.end() ) {
406
407       if ( debug ) ::std::cout << "CustomDataContainer::find() t not found" << ::std::endl;
408
409       return NULL;
410     }
411
412     if ( debug ) ::std::cout << "CustomDataContainer::find() t found" << ::std::endl;
413
414     CDCMediumCIt it2 = it->second.find( b );
415     if ( it2 == it->second.end() ) {
416
417       if ( debug ) ::std::cout << "CustomDataContainer::find() b not found" << ::std::endl;
418
419       return NULL;
420     }
421
422     if ( debug ) ::std::cout << "CustomDataContainer::find() b found" << ::std::endl;
423
424     CDCInnerCIt it3 = it2->second.find( r );
425     if ( it3 == it2->second.end() ) {
426
427       if ( debug ) ::std::cout << "CustomDataContainer::find() r not found" << ::std::endl;
428
429       return NULL;
430     }
431     if ( debug ) ::std::cout << "CustomDataContainer::find() r found, data = " << it3->second <<
    ::std::endl;
432     return &(it3->second);
433   }
434
435
436   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
437   bool CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::insert( const
    Data& d, const T& t, double b, double r ) {
438     const Data* ptr = find( t, b, r );
439     if ( ptr != NULL ) return false;
```

```
440     data_map[t][b][r] = d;
441     return true;
442   }
443
444
445 //   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
446 //   void CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::normalizeMap(
      const T& t ) {
447 //     MidFunctor mid_funct;
448 //     InFunctor in_funct;
449 //
450 //     double curr_b;
451 //     double curr_r;
452 //     double delta_b;
453 //     double delta_r;
454 //     const Data* ret_val = NULL;
455 //
456 //     double curr_dist;
457 //     double min_dist = INFINITY;
458 //
459 //     for ( CDCCIt it = data_map.begin(); it != data_map.end(); it++ ) {
460 //
461 //       if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() start T = " << it->first << ";
      end T = " << rx << ::std::endl;
462 //
463 //       if ( it->first == DB_CDATA_ALL_OUTER_KEYS ) {
464 //         if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() overriding start T = " << tx <<
      "; end T = " << rx << ::std::endl;
465 //
466 //         curr_b = mid_funct(tx,rx);
467 //         curr_r = in_funct(tx,rx);
468 //       }
469 //       else {
470 //         curr_b = mid_funct(it->first,rx);
471 //         curr_r = in_funct(it->first,rx);
472 //       }
473 //
474 //       if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() curr bearing = " << curr_b << ";
      curr range = " << curr_r << ::std::endl;
475 //
476 //       CDCMediumCIt itb = it->second.begin();
477 //       if ( itb->first == DB_CDATA_ALL_MEDIUM_KEYS ) delta_b = 0;
478 //       else {
479 //         itb = it->second.lower_bound( curr_b );
480 //         if ( itb == it->second.end() ) itb = (++(it->second.rbegin())).base();
481 //
482 //         delta_b = curr_b - itb->first;
483 //         if (delta_b < 0.0) delta_b = -delta_b;
484 //         if (delta_b > M_PI) delta_b = 2.0*M_PI - delta_b ;
485 //       }
486 //
487 //       double ort_dist = pow( curr_r * sin(delta_b) , 2.0 );
488 //       double ort_projection = ::std::sqrt( curr_r*curr_r - ort_dist );
489 //
490 //       if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() nearest bearing = " <<
      itb->first << "; diff bearing = " << delta_b
491 //                                << "; orthog distance = " << ort_dist << "; orthog range projection = " <<
      ort_projection << ::std::endl;
492 //
493 //       CDCInnerCIt itr = itb->second.begin();
494 //       if ( itr->first == DB_CDATA_ALL_INNER_KEYS ) curr_dist = ort_dist;
495 //       else {
496 //         itr = itb->second.lower_bound( ort_projection );
497 //         if ( itr == itb->second.end() ) itr = (++(itb->second.rbegin())).base();
498 //         double adj_distance = ::std::abs( ort_projection - itr->first );
499 //         curr_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
500 //       }
501 //
502 //       if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() nearest range = " << itr->first
      << "; distance = " << curr_dist
503 //                                << "; min distance = " << min_dist << ::std::endl;
504 //
505 //       if ( curr_dist < min_dist ) {
506 //         min_dist = curr_dist;
507 //         ret_val = &(itr->second);
508 //         if ( curr_dist == 0 ) break;
509 //       }
510 //
511 //     }
512 //
513 //    if ( debug ) ::std::cout << "CustomDataContainer::normalizeMap() ret value " << *ret_val <<
      ::std::endl;
514 //
515 //   }
516
517
```

```
518   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
519   void CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::replace( const
      Data& d, const T& t, double b, double r ) {
520     data_map[t][b][r] = d;
521   }
522
523
524   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
525   void CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::erase( const T&
      t, double b, double r ) {
526     data_map[t][b].erase(r);
527     if ( data_map[t][b].empty() ) data_map[t].erase(b);
528     if ( data_map[t].empty() ) data_map.erase(t);
529   }
530
531
532   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
533   void CustomDataContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::clear() {
534     data_map.clear();
535   }
536
537
543   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
544   class CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp > {
545
546
547     protected:
548
549
550     typedef typename ::std::map< double, Data*, InComp > InnerData;
551     typedef typename InnerData::iterator CDCInnerIt;
552     typedef typename InnerData::reverse_iterator CDCInnerRIt;
553     typedef typename InnerData::const_iterator CDCInnerCIt;
554     typedef typename InnerData::const_reverse_iterator CDCInnerCRIt;
555
556     typedef typename ::std::map< double, InnerData, MidComp > MediumData;
557     typedef typename MediumData::iterator CDCMediumIt;
558     typedef typename MediumData::const_iterator CDCMediumCIt;
559     typedef typename MediumData::reverse_iterator CDCMediumRIt;
560     typedef typename MediumData::const_reverse_iterator CDCMediumCRIt;
561
562     typedef ::std::map< T, MediumData, OutComp > CustomContainer;
563     typedef typename CustomContainer::iterator CDCIt;
564     typedef typename CustomContainer::reverse_iterator CDCRIt;
565     typedef typename CustomContainer::const_iterator CDCCIt;
566     typedef typename CustomContainer::const_reverse_iterator CDCCRIt;
567
568
569     public:
570
571 #if __cplusplus >= 201103L // C++11 or later
572     static constexpr double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
573
574     static constexpr double DB_CDATA_ALL_INNER_KEYS = -10.0;
575 #else
576     static const double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
577
578     static const double DB_CDATA_ALL_INNER_KEYS = -10.0;
579 #endif // __cplusplus >= 201103L
580     static const T DB_CDATA_ALL_OUTER_KEYS;
581
582
583     CustomDataContainer() :  debug(false), data_map() { }
584
585
586     ~CustomDataContainer() { clear(); }
587
588
589     bool empty()const { return data_map.empty(); }
590
591
592     int size()const { return data_map.size(); }
593
594
595     Data* get( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
      DB_CDATA_ALL_INNER_KEYS ) const;
596
597     Data* get( const T& tx, const T& rx ) const;
598
599
611     bool insert( Data* data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS,
      double r = DB_CDATA_ALL_INNER_KEYS );
612
```

```
613     void replace( Data* data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS,
    double r = DB_CDATA_ALL_INNER_KEYS );
614
615
626     void erase( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
    DB_CDATA_ALL_INNER_KEYS );
627
628
629 //      void normalizeMap( const T& t );
630
631
635     void clear();
636
637
638     void setDebug( bool flag ) { debug = flag; }
639
640     bool usingDebug() { return debug; }
641
642
643     protected:
644
645
646     bool debug;
647
648
649     CustomContainer data_map;
650
651
662     Data*& find( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
    DB_CDATA_ALL_INNER_KEYS ) const;
663
664
665   };
666
667
668   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
669   const T CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
    >::DB_CDATA_ALL_OUTER_KEYS = T();
670
671
672   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
673   Data*& CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::find( const
    T& t, double b, double r )const {
674
675     static Data* not_found = NULL;
676
677     if (debug) ::std::cout « "CustomDataContainer*::find() « t = " « t « "; b = " « b « "; r = " « r «
    ::std::endl;
678
679     CDCCIt it = data_map.find( t );
680     if ( it == data_map.end() ) {
681
682       if ( debug ) ::std::cout « "CustomDataContainer*::find() t not found" « ::std::endl;
683
684       return not_found;
685     }
686
687     if ( debug ) ::std::cout « "CustomDataContainer*::find() t found" « ::std::endl;
688
689     CDCMediumCIt it2 = it->second.find( b );
690     if ( it2 == it->second.end() ) {
691
692       if ( debug ) ::std::cout « "CustomDataContainer*::find() b not found" « ::std::endl;
693
694       return not_found;
695     }
696
697     if ( debug ) ::std::cout « "CustomDataContainer*::find() b found" « ::std::endl;
698
699     CDCInnerCIt it3 = it2->second.find( r );
700     if ( it3 == it2->second.end() ) {
701
702       if ( debug ) ::std::cout « "CustomDataContainer*::find() r not found" « ::std::endl;
703
704       return not_found;
705     }
706     if ( debug ) ::std::cout « "CustomDataContainer*::find() r found, data = " « *it3->second «
    ::std::endl;
707     return const_cast<Data*&>(it3->second);
708   }
709
710
711   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
712   Data* CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::get( const T&
```

```
    t, double b, double r )const {
713
714     if ( debug ) ::std::cout « "CustomDataContainer*::get() t = " « t « "; b = " « b « "; r = " « r «
    ::std::endl;
715
716     if ( data_map.empty() ) return NULL;
717
718     Data* ptr = find( DB_CDATA_ALL_OUTER_KEYS, DB_CDATA_ALL_MEDIUM_KEYS, DB_CDATA_ALL_INNER_KEYS );
719
720     if ( ptr != NULL ) {
721       if ( debug ) ::std::cout « "CustomDataContainer*::get() found ptr = " « ptr « "; object = " « *ptr
    « ::std::endl;
722
723       return new Data( *ptr );
724     }
725
726     CDCCIt it = data_map.find( t );
727     if ( it == data_map.end() ) return NULL;
728
729     CDCMediumCIt it2 = it->second.lower_bound( b );
730     CDCMediumCRIt rit2 = it->second.rbegin();
731
732     CDCInnerCIt it3;
733     CDCInnerCRIt rit3;
734
735     if ( it2 == it->second.end() ) {
736       if( rit2 == it->second.rend() ) return NULL;
737       it3 = rit2->second.lower_bound( r );
738       rit3 = rit2->second.rbegin();
739       if ( it3 != rit2->second.end() ) return new Data( *(it3->second) );
740       if ( rit3 != rit2->second.rend() ) return new Data ( *(rit3->second) );
741       return NULL;
742     }
743     else {
744       it3 = it2->second.lower_bound( r );
745       rit3 = it2->second.rbegin();
746       if ( it3 != it2->second.end() ) return new Data( *(it3->second) );
747       if ( rit3 != it2->second.rend() ) return new Data( *(rit3->second) );
748       return NULL;
749     }
750     if ( it3 == it2->second.end() ) return NULL;
751     return new Data( *(it3->second) );
752   }
753
754
755   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
    InComp >
756   Data* CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::get( const T&
    tx, const T& rx )const {
757     MidFunctor mid_funct;
758     InFunctor in_funct;
759
760     double curr_b;
761     double curr_r;
762     double delta_b;
763 //    double delta_r;
764     const Data* ret_val = NULL;
765
766     double curr_dist;
767     double min_dist = INFINITY;
768
769     if ( data_map.empty() == true ) {
770       if ( debug ) ::std::cout « "CustomDataContainer*::get() data_map is empty " « ::std::endl;
771
772       return new Data();
773     }
774
775     for ( CDCCIt it = data_map.begin(); it != data_map.end(); it++ ) {
776
777       if ( debug ) ::std::cout « "CustomDataContainer*::get() start T = " « it->first « "; end T = " «
    rx « ::std::endl;
778
779       if ( it->first == DB_CDATA_ALL_OUTER_KEYS ) {
780         if ( debug ) ::std::cout « "CustomDataContainer*::get() overriding start T = " « tx « "; end T =
    " « rx « ::std::endl;
781
782         curr_b = mid_funct(tx,rx);
783         curr_r = in_funct(tx,rx);
784       }
785       else {
786         curr_b = mid_funct(it->first,rx);
787         curr_r = in_funct(it->first,rx);
788       }
789
790       if ( debug ) ::std::cout « "CustomDataContainer*::get() curr bearing = " « curr_b * 180.0 / M_PI
791                                « "; curr range = " « curr_r « ::std::endl;
792
```

```
793       CDCMediumCIt itb = it->second.begin();
794       if ( itb->first == DB_CDATA_ALL_MEDIUM_KEYS ) delta_b = 0;
795       else {
796         itb = it->second.lower_bound( curr_b );
797         if ( itb == it->second.end() ) itb = (++(it->second.rbegin())).base();
798 
799         delta_b = curr_b - itb->first;
800         if (delta_b < 0.0) delta_b = -delta_b;
801         if (delta_b > M_PI) delta_b = 2.0*M_PI - delta_b ;
802       }
803 
804       double ort_dist = curr_r * sin(delta_b);
805       double ort_projection = ::std::sqrt( curr_r*curr_r - ort_dist*ort_dist );
806 
807       if ( debug ) ::std::cout << "CustomDataContainer*::get() nearest bearing = " << itb->first * 180.0 /
      M_PI
808                                << "; diff bearing = " << delta_b * 180.0 / M_PI << "; orthog distance = " <<
      ort_dist
809                                << "; orthog range projection = " << ort_projection << ::std::endl;
810 
811       CDCInnerCIt itr = itb->second.begin();
812       if ( itr->first == DB_CDATA_ALL_INNER_KEYS ) curr_dist = ort_dist;
813       else {
814         itr = itb->second.lower_bound( ort_projection );
815         if ( itr == itb->second.begin() || itr == itb->second.end() || itr->first == ort_projection ) {
816           if ( itr == itb->second.end() ) itr = (++(itb->second.rbegin())).base();
817 //           ::std::cout << " range = " << itr->first << ::std::endl;
818           double adj_distance = ::std::abs( ort_projection - itr->first );
819           curr_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
820         }
821         else {
822           double adj_distance = ::std::abs( ort_projection - itr->first );
823           double first_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
824 
825           if (debug) ::std::cout << "CustomDataContainer*::get() first try, range = " << itr->first
826                                  << "; dist = " << first_dist << ::std::endl;
827 
828           itr--;
829           adj_distance = ::std::abs( ort_projection - itr->first );
830           double before_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
831 
832           if (debug) ::std::cout << "CustomDataContainer*::get() second try, range = " << itr->first
833                                  << "; dist = " << before_dist << ::std::endl;
834 
835           curr_dist = ::std::min( first_dist, before_dist );
836           if ( curr_dist == first_dist ) itr++;
837         }
838       }
839 
840       if ( debug ) ::std::cout << "CustomDataContainer*::get() nearest range = " << itr->first << ";
      distance = " << curr_dist
841                                << "; min distance = " << min_dist << ::std::endl;
842 
843       if ( curr_dist < min_dist ) {
844         min_dist = curr_dist;
845         ret_val = (itr->second);
846         if ( curr_dist == 0 ) break;
847       }
848 
849     }
850 
851     if ( debug && ( ret_val != NULL ) ) ::std::cout << "CustomDataContainer*::get() ret value " <<
      *ret_val << ::std::endl;
852 
853     if ( ret_val != NULL ) return ret_val->clone();
854     return new Data();
855   }
856 
857 
858   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
859   bool CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::insert( Data*
      d, const T& t, double b, double r ) {
860 
861     if (debug && d != NULL) ::std::cout << "CustomDataContainer*::insert() &d = " << d << "; d = " << *d <<
      "; t = " << t << "; b = " << b << "; r = " << r << ::std::endl;
862 
863     Data* ptr = find( t, b, r );
864 
865     if ( ptr != NULL ) {
866       delete d;
867       return false;
868     }
869 
870     data_map[t][b][r] = d;
871     return true;
872   }
```

```
873
874
875   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
876   void CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::replace( Data*
      d, const T& t, double b, double r ) {
877
878     if (debug) ::std::cout « "CustomDataContainer*::replace() d = " « *d « "; t = " « t « "; b = " « b «
      "; r = " « r « ::std::endl;
879
880     Data* ptr = find( t, b, r );
881
882     if ( ptr != NULL ) {
883       delete ptr;
884       ptr = d;
885     }
886     data_map[t][b][r] = d;
887   }
888
889
890   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
891   void CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::erase( const T&
      t, double b, double r ) {
892     Data* ptr = find( t, b, r );
893     if ( ptr == NULL ) return;
894     if ( ptr != NULL ) delete ptr;
895     data_map[t][b].erase(r);
896     if ( data_map[t][b].empty() ) data_map[t].erase(b);
897     if ( data_map[t].empty() ) data_map.erase(t);
898   }
899
900
901   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class
      InComp >
902   void CustomDataContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::clear() {
903     if ( data_map.empty() ) return;
904     for ( CDCIt it = data_map.begin(); it != data_map.end(); it++ ) {
905       for ( CDCMediumIt it2 = it->second.begin(); it2 != it->second.end(); it2++ ) {
906         for ( CDCInnerIt it3 = it2->second.begin(); it3 != it2->second.end(); it3++ ) {
907           if ( it3->second != NULL ) delete it3->second;
908         }
909       }
910     }
911     data_map.clear();
912   }
913
921   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp = ::std::less<T>,
      class MidComp = ::std::less<double>, class InComp = ::std::less<double> >
922   class CustomDataTimeContainer {
923
924
925     protected:
926
930     typedef typename ::std::map< time_t, Data > TimeData;
931     typedef typename TimeData::iterator CDTCTimeIt;
932     typedef typename TimeData::reverse_iterator CDTCTimeRIt;
933     typedef typename TimeData::const_iterator CDTCTimeCIt;
934     typedef typename TimeData::const_reverse_iterator CDTCTimeCRIt;
935
939     typedef typename ::std::map< double, TimeData, InComp > InnerData;
940     typedef typename InnerData::iterator CDCInnerIt;
941     typedef typename InnerData::reverse_iterator CDCInnerRIt;
942     typedef typename InnerData::const_iterator CDCInnerCIt;
943     typedef typename InnerData::const_reverse_iterator CDCInnerCRIt;
944
948     typedef typename ::std::map< double, InnerData, MidComp > MediumData;
949     typedef typename MediumData::iterator CDCMediumIt;
950     typedef typename MediumData::const_iterator CDCMediumCIt;
951     typedef typename MediumData::reverse_iterator CDCMediumRIt;
952     typedef typename MediumData::const_reverse_iterator CDCMediumCRIt;
953
957     typedef ::std::map< T, MediumData, OutComp > CustomContainer;
958     typedef typename CustomContainer::iterator CDCIt;
959     typedef typename CustomContainer::reverse_iterator CDCRIt;
960     typedef typename CustomContainer::const_iterator CDCCIt;
961     typedef typename CustomContainer::const_reverse_iterator CDCCRIt;
962
963
964     public:
965
966
967 #if __cplusplus >= 201103L // C++11 or later
968     static constexpr double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
969
970     static constexpr double DB_CDATA_ALL_INNER_KEYS = -10.0;
971 #else
```

```
972     static const double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
973
974     static const double DB_CDATA_ALL_INNER_KEYS = -10.0;
975 #endif // __cplusplus >= 201103L
976     static const T DB_CDATA_ALL_OUTER_KEYS;
977
978     static const Time DB_CDATA_ALL_TIME_KEYS;
979
980
984     CustomDataTimeContainer() :  debug(false), data_map() { }
985
989     ~CustomDataTimeContainer() { clear(); }
990
991
997     MediumData& operator[] ( const T& key ) { return data_map[key]; }
998
999
1004     bool empty()const { return data_map.empty(); }
1005
1006
1011     int size()const { return data_map.size(); }
1012
1013
1026     Data get( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
      DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1027
1036     Data get( const T& tx, const T& rx, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1037
1038
1052     bool insert( const Data& data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b =
      DB_CDATA_ALL_MEDIUM_KEYS, double r = DB_CDATA_ALL_INNER_KEYS, const Time& time_key =
      DB_CDATA_ALL_TIME_KEYS );
1053
1067     void replace( const Data& data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b =
      DB_CDATA_ALL_MEDIUM_KEYS, double r = DB_CDATA_ALL_INNER_KEYS, const Time& time_key =
      DB_CDATA_ALL_TIME_KEYS );
1068
1081     void erase( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
      DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS);
1082
1083
1087     void clear();
1088
1089
1094     void setDebug( bool flag ) { debug = flag; }
1095
1096
1101     bool usingDebug() { return debug; }
1102
1103
1104     protected:
1105
1106
1110     bool debug;
1111
1112
1116     CustomContainer data_map;
1117
1118
1119     typedef typename ::std::pair< Data, bool > DataFind;
1120
1121
1134     DataFind find( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r
      = DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1135
1148     Data calculateData( const TimeData& time_data , const Time& time_key = DB_CDATA_ALL_TIME_KEYS )
      const;
1149
1150
1151   };
1152
1153
1154   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1155   const T CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
      >::DB_CDATA_ALL_OUTER_KEYS = T();
1156
1157
1158   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1159   const Time CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
      >::DB_CDATA_ALL_TIME_KEYS = Time(1, 1, 1901, 0, 0, 0);
1160
1161
1162   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1163   Data CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get( const
```

```
    T& t, double b, double r, const Time& time_key )const {
1164      if ( data_map.empty() ) return Data();
1165
1166      if ( debug ) ::std::cout « "CustomDataTimeContainer::get() t = " « t « "; b = " « b « "; r = " « r
1167                                « "; time_key = " « time_key « ::std::endl;
1168
1169      DataFind ret_val = find();
1170      if ( ret_val.second == true ) return ret_val.first;
1171
1172      CDCCIt it = data_map.find( t );
1173      if ( it == data_map.end() ) return Data();
1174
1175      CDCMediumCIt it2 = it->second.lower_bound( b );
1176      CDCMediumCRIt rit2 = it->second.rbegin();
1177
1178      CDCInnerCIt it3;
1179      CDCInnerCRIt rit3;
1180
1181      if ( it2 == it->second.end() ) {
1182          if ( rit2 == it->second.rend() ) return Data();
1183          it3 = rit2->second.lower_bound( r );
1184          rit3 = rit2->second.rbegin();
1185          if ( it3 != rit2->second.end() ) return calculateData( it3->second, time_key);
1186          if ( rit3 != rit2->second.rend() ) return calculateData( rit3->second, time_key);
1187          return Data();
1188      }
1189      else {
1190          it3 = it2->second.lower_bound( r );
1191          rit3 = it2->second.rbegin();
1192          if ( it3 != it2->second.end() ) return calculateData(it3->second, time_key);
1193          if ( rit3 != it2->second.rend() ) return calculateData(rit3->second, time_key);
1194          return Data();
1195      }
1196      if ( it3 == it2->second.end() ) return Data();
1197      return calculateData(it3->second, time_key);
1198  }
1199
1200
1201  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp, class InComp >
1202  Data CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::get( const T& tx, const T& rx, const Time& time_key )const {
1203      MidFunctor mid_funct;
1204      InFunctor in_funct;
1205
1206      double curr_b;
1207      double curr_r;
1208      double delta_b;
1209      double delta_r;
1210
1211      double curr_dist;
1212      double min_dist = INFINITY;
1213
1214      if ( data_map.empty() == true ) {
1215          if ( debug ) ::std::cout « "CustomDataTimeContainer::get() data_map is empty " « ::std::endl;
1216
1217          return Data();
1218      }
1219      const TimeData* time_data_ptr = NULL;
1220
1221      for ( CDCCIt it = data_map.begin(); it != data_map.end(); it++ ) {
1222
1223          if ( debug ) ::std::cout « "CustomDataTimeContainer::get() start T = " « it->first « "; end T = " « rx « ::std::endl;
1224
1225          if ( it->first == DB_CDATA_ALL_OUTER_KEYS ) {
1226              if ( debug ) ::std::cout « "CustomDataTimeContainer::get() overriding start T = " « tx « "; end T = " « rx « ::std::endl;
1227
1228              curr_b = mid_funct(tx,rx);
1229              curr_r = in_funct(tx,rx);
1230          }
1231          else {
1232              curr_b = mid_funct(it->first,rx);
1233              curr_r = in_funct(it->first,rx);
1234          }
1235
1236          if ( debug ) ::std::cout « "CustomDataTimeContainer::get() curr bearing = " « curr_b * 180.0 / M_PI
1237                                   « "; curr range = " « curr_r « ::std::endl;
1238
1239          CDCMediumCIt itb = it->second.begin();
1240          if ( itb->first == DB_CDATA_ALL_MEDIUM_KEYS ) delta_b = 0;
1241          else {
1242              itb = it->second.lower_bound( curr_b );
1243              if ( itb == it->second.end() ) itb = (++(it->second.rbegin())).base();
```

```
1244
1245             delta_b = curr_b - itb->first;
1246             if (delta_b < 0.0) delta_b = -delta_b;
1247             if (delta_b > M_PI) delta_b = 2.0*M_PI - delta_b ;
1248         }
1249
1250         double ort_dist = curr_r * sin(delta_b);
1251         double ort_projection = ::std::sqrt( curr_r*curr_r - ort_dist*ort_dist );
1252
1253         if ( debug ) ::std::cout << "CustomDataTimeContainer::get() nearest bearing = " << itb->first *
     180.0 / M_PI
1254                                  << "; diff bearing = " << delta_b * 180.0 / M_PI << "; orthog
     distance = " << ort_dist
1255                                  << "; orthog range projection = " << ort_projection << ::std::endl;
1256
1257         CDCInnerCIt itr = itb->second.begin();
1258         if ( itr->first == DB_CDATA_ALL_INNER_KEYS ) curr_dist = ort_dist;
1259         else {
1260             itr = itb->second.lower_bound( ort_projection );
1261             if ( itr == itb->second.begin() || itr == itb->second.end() || itr->first ==
     ort_projection ) {
1262                 if ( itr == itb->second.end() ) itr = (++(itb->second.rbegin())).base();
1263                 double adj_distance = ::std::abs( ort_projection - itr->first );
1264                 curr_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
1265             }
1266             else {
1267                 double adj_distance = ::std::abs( ort_projection - itr->first );
1268                 double first_dist = ::std::sqrt( ort_projection*ort_projection +
     adj_distance*adj_distance );
1269
1270                 if (debug) ::std::cout << "CustomDataTimeContainer::get() first try, range = " <<
     itr->first
1271                                        << "; dist = " << first_dist << ::std::endl;
1272
1273                 itr--;
1274                 adj_distance = ::std::abs( ort_projection - itr->first );
1275                 double before_dist = ::std::sqrt( ort_projection*ort_projection +
     adj_distance*adj_distance );
1276
1277                 if (debug) ::std::cout << "CustomDataTimeContainer::get() second try, range = " <<
     itr->first
1278                                        << "; dist = " << before_dist << ::std::endl;
1279
1280                 curr_dist = ::std::min( first_dist, before_dist );
1281                 if ( curr_dist == first_dist ) itr++;
1282             }
1283         }
1284         if ( debug ) ::std::cout << "CustomDataTimeContainer::get() nearest range = " << itr->first <<
     "; distance = " << curr_dist
1285                                  << "; min distance = " << min_dist << ::std::endl;
1286
1287         if ( curr_dist < min_dist ) {
1288             min_dist = curr_dist;
1289             time_data_ptr = &(itr->second);
1290             if ( curr_dist == 0 ) break;
1291         }
1292
1293     }
1294
1295     if ( time_data_ptr != NULL ) return calculateData( *time_data_ptr, time_key);
1296     return Data();
1297  }
1298
1299
1300  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1301  typename CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
     >::DataFind CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::find(
     const T& t, double b, double r, const Time& time_key )const {
1302
1303     if (debug) ::std::cout << "CustomDataTimeContainer::find() << t = " << t << "; b = " << b << "; r = "
1304                            << r << "; time_key = " << time_key << ::std::endl;
1305
1306     CDCCIt it = data_map.find( t );
1307     if ( it == data_map.end() ) {
1308
1309         if ( debug ) ::std::cout << "CustomDataTimeContainer::find() t not found" << ::std::endl;
1310
1311         return DataFind( Data(), false );
1312     }
1313
1314     if ( debug ) ::std::cout << "CustomDataTimeContainer::find() t found" << ::std::endl;
1315
1316     CDCMediumCIt it2 = it->second.find( b );
1317     if ( it2 == it->second.end() ) {
1318
1319         if ( debug ) ::std::cout << "CustomDataTimeContainer::find() b not found" << ::std::endl;
```

```
1320
1321          return DataFind( Data(), false );
1322      }
1323
1324      if ( debug ) ::std::cout « "CustomDataTimeContainer::find() b found" « ::std::endl;
1325
1326      CDCInnerCIt it3 = it2->second.find( r );
1327      if ( it3 == it2->second.end() ) {
1328
1329          if ( debug ) ::std::cout « "CustomDataTimeContainer::find() r not found" « ::std::endl;
1330
1331          return DataFind( Data(), false );
1332      }
1333      if ( debug ) ::std::cout « "CustomDataTimeContainer::find() r found" « ::std::endl;
1334
1335      CDTCTimeCIt it4 = it3->second.find( time_key );
1336      if ( it4 == it3->second.end() ) {
1337
1338          if ( debug ) ::std::cout « "CustomDataTimeContainer::find() time_key not found" «
     ::std::endl;
1339
1340          return DataFind( Data(), false );
1341      }
1342
1343      if ( debug ) ::std::cout « "CustomDataTimeContainer::find() time_key found, Data = " «
     it4->second « ::std::endl;
1344
1345      return DataFind( it4->second, true );
1346  }
1347
1348
1349  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1350  bool CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::insert(
     const Data& d, const T& t, double b, double r, const Time& time_key ) {
1351      DataFind data = find( t, b, r, time_key );
1352      if ( data.second == true ) return false;
1353      data_map[t][b][r][time_key] = d;
1354      return true;
1355  }
1356
1357
1358  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1359  void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::replace(
     const Data& d, const T& t, double b, double r, const Time& time_key ) {
1360      data_map[t][b][r][time_key] = d;
1361  }
1362
1363
1364  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1365  void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::erase(
     const T& t, double b, double r, const Time& time_key ) {
1366      data_map[t][b][r].erase(time_key);
1367      if ( data_map[t][b][r].empty() ) data_map[t][b].erase(r);
1368      if ( data_map[t][b].empty() ) data_map[t].erase(b);
1369      if ( data_map[t].empty() ) data_map.erase(t);
1370  }
1371
1372
1373  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1374  void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp >::clear() {
1375      data_map.clear();
1376  }
1377
1378  template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1379  Data CustomDataTimeContainer< T, MidFunctor, InFunctor, Data, OutComp, MidComp, InComp
     >::calculateData( const TimeData& time_data, const Time& time_key )const {
1380     if (debug) ::std::cout « "CustomDataTimeContainer::calculateData() time_key = " « time_key «
     ::std::endl;
1381
1382     if ( time_data.empty() ) {
1383       if (debug) ::std::cout « "CustomDataTimeContainer::calculateData() time_data is empty." «
     ::std::endl;
1384
1385       return Data();
1386     }
1387
1388     if ( time_data.size() == 1 ) {
1389       if (debug) ::std::cout « "CustomDataTimeContainer::calculateData() time_data has size 1.  Data
     created "
1390                              « time_data.begin()->second « ::std::endl;
1391
1392       return time_data.begin()->second;
```

```
1393     }
1394
1395     time_t normalized_time = time_key;
1396
1397     if ( normalized_time < time_data.begin()->first ) {
1398       if (debug) ::std::cout << "CustomDataTimeContainer::calculateData() time_key has time < first key.
     Data created "
1399                              << time_data.begin()->second << ::std::endl;
1400
1401       return time_data.begin()->second;
1402     }
1403
1404     normalized_time %= ( time_data.rbegin()->first - time_data.begin()->first );
1405     if ( normalized_time == 0 )
1406       return time_data.begin()->second;
1407
1408     normalized_time += time_data.begin();
1409     CDTCTimeIt upper_it = time_data.upper_bound(normalized_time);
1410     CDTCTimeIt lower_it = upper_it;
1411     --lower_it;
1412
1413
1414     return ( lower_it->second * ( (normalized_time - lower_it->first) / ( upper_it->first -
     lower_it->first ) )
1415              + upper_it->second * ( (upper_it->first - normalized_time) / ( upper_it->first -
     lower_it->first ) ) );
1416   }
1417
1418
1426   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1427   class CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp > {
1428
1429
1430     protected:
1431
1435     typedef typename ::std::map< time_t, Data* > TimeData;
1436     typedef typename TimeData::iterator CDTCTimeIt;
1437     typedef typename TimeData::reverse_iterator CDTCTimeRIt;
1438     typedef typename TimeData::const_iterator CDTCTimeCIt;
1439     typedef typename TimeData::const_reverse_iterator CDTCTimeCRIt;
1440
1444     typedef typename ::std::map< double, TimeData, InComp > InnerData;
1445     typedef typename InnerData::iterator CDCInnerIt;
1446     typedef typename InnerData::reverse_iterator CDCInnerRIt;
1447     typedef typename InnerData::const_iterator CDCInnerCIt;
1448     typedef typename InnerData::const_reverse_iterator CDCInnerCRIt;
1449
1453     typedef typename ::std::map< double, InnerData, MidComp > MediumData;
1454     typedef typename MediumData::iterator CDCMediumIt;
1455     typedef typename MediumData::const_iterator CDCMediumCIt;
1456     typedef typename MediumData::reverse_iterator CDCMediumRIt;
1457     typedef typename MediumData::const_reverse_iterator CDCMediumCRIt;
1458
1462     typedef ::std::map< T, MediumData, OutComp > CustomContainer;
1463     typedef typename CustomContainer::iterator CDCIt;
1464     typedef typename CustomContainer::reverse_iterator CDCRIt;
1465     typedef typename CustomContainer::const_iterator CDCCIt;
1466     typedef typename CustomContainer::const_reverse_iterator CDCCRIt;
1467
1468
1469     public:
1470
1471
1472 #if __cplusplus >= 201103L // C++11 or later
1473     static constexpr double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
1474
1475     static constexpr double DB_CDATA_ALL_INNER_KEYS = -10.0;
1476 #else
1477     static const double DB_CDATA_ALL_MEDIUM_KEYS = -190.0;
1478
1479     static const double DB_CDATA_ALL_INNER_KEYS = -10.0;
1480 #endif // __cplusplus >= 201103L
1481     static const T DB_CDATA_ALL_OUTER_KEYS;
1482
1483     static const Time DB_CDATA_ALL_TIME_KEYS;
1484
1485
1489     CustomDataTimeContainer() :  debug(false), data_map() { }
1490
1494     ~CustomDataTimeContainer() { clear(); }
1495
1496
1502     MediumData& operator[] ( const T& key ) { return data_map[key]; }
1503
1504
1509     bool empty()const { return data_map.empty(); }
```

```
1510
1511
1516     int size()const { return data_map.size(); }
1517
1518
1531     Data* get( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
     DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1532
1541     Data* get( const T& tx, const T& rx, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1542
1543
1557     bool insert( Data* data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS,
     double r = DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS );
1558
1572     void replace( Data* data, const T& t = DB_CDATA_ALL_OUTER_KEYS, double b =
     DB_CDATA_ALL_MEDIUM_KEYS, double r = DB_CDATA_ALL_INNER_KEYS, const Time& time_key =
     DB_CDATA_ALL_TIME_KEYS );
1573
1586     void erase( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r =
     DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS);
1587
1588
1592     void clear();
1593
1594
1599     void setDebug( bool flag ) { debug = flag; }
1600
1601
1606     bool usingDebug() { return debug; }
1607
1608
1609     protected:
1610
1611
1615     bool debug;
1616
1617
1621     CustomContainer data_map;
1622
1623
1624     typedef typename ::std::pair< Data*, bool > DataFind;
1625
1626
1639     DataFind find( const T& t = DB_CDATA_ALL_OUTER_KEYS, double b = DB_CDATA_ALL_MEDIUM_KEYS, double r
     = DB_CDATA_ALL_INNER_KEYS, const Time& time_key = DB_CDATA_ALL_TIME_KEYS ) const;
1640
1653     Data* calculateData( const TimeData& time_data , const Time& time_key = DB_CDATA_ALL_TIME_KEYS )
     const;
1654
1655
1656   };
1657
1658
1659   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1660   const T CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
     >::DB_CDATA_ALL_OUTER_KEYS = T();
1661
1662
1663   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1664   const Time CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
     >::DB_CDATA_ALL_TIME_KEYS = Time(1, 1, 1901, 0, 0, 0);
1665
1666
1667   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1668   Data* CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::get(
     const T& t, double b, double r, const Time& time_key )const {
1669     if ( data_map.empty() ) return new Data();
1670
1671     if ( debug ) ::std::cout « "CustomDataTimeContainer*::get() t = " « t « "; b = " « b « "; r = " « r
1672       « "; time_key = " « time_key « ::std::endl;
1673
1674     DataFind ret_val = find();
1675     if ( ret_val.second == true ) return ret_val.first->clone();
1676
1677     CDCCIt it = data_map.find( t );
1678     if ( it == data_map.end() ) return new Data();
1679
1680     CDCMediumCIt it2 = it->second.lower_bound( b );
1681     CDCMediumCRIt rit2 = it->second.rbegin();
1682
1683     CDCInnerCIt it3;
1684     CDCInnerCRIt rit3;
1685
1686     if ( it2 == it->second.end() ) {
```

```
1687       if ( rit2 == it->second.rend() ) return new Data();
1688       it3 = rit2->second.lower_bound( r );
1689       rit3 = rit2->second.rbegin();
1690       if ( it3 != rit2->second.end() ) return calculateData(it3->second, time_key);
1691       if ( rit3 != rit2->second.rend() ) return calculateData(rit3->second, time_key);
1692       return new Data();
1693     }
1694     else {
1695       it3 = it2->second.lower_bound( r );
1696       rit3 = it2->second.rbegin();
1697       if ( it3 != it2->second.end() ) return calculateData(it3->second, time_key);
1698       if ( rit3 != it2->second.rend() ) return calculateData(rit3->second, time_key);
1699       return new Data();
1700     }
1701     if ( it3 == it2->second.end() ) return new Data();
1702     return calculateData(it3->second, time_key);
1703   }
1704
1705
1706   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1707   Data* CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::get(
      const T& tx, const T& rx, const Time& time_key )const {
1708     MidFunctor mid_funct;
1709     InFunctor in_funct;
1710
1711     double curr_b;
1712     double curr_r;
1713     double delta_b;
1714 //    double delta_r;
1715
1716     double curr_dist;
1717     double min_dist = INFINITY;
1718
1719     if ( data_map.empty() == true ) {
1720       if ( debug ) ::std::cout << "CustomDataTimeContainer::get() data_map is empty " << ::std::endl;
1721
1722       return new Data();
1723     }
1724     const TimeData* time_data_ptr = NULL;
1725
1726     for ( CDCCIt it = data_map.begin(); it != data_map.end(); it++ ) {
1727
1728       if ( debug ) ::std::cout << "CustomDataTimeContainer::get() start T = " << it->first << "; end T = "
      << rx << ::std::endl;
1729
1730       if ( it->first == DB_CDATA_ALL_OUTER_KEYS ) {
1731         if ( debug ) ::std::cout << "CustomDataTimeContainer::get() overriding start T = " << tx << "; end
      T = " << rx << ::std::endl;
1732
1733         curr_b = mid_funct(tx,rx);
1734         curr_r = in_funct(tx,rx);
1735       }
1736       else {
1737         curr_b = mid_funct(it->first,rx);
1738         curr_r = in_funct(it->first,rx);
1739       }
1740
1741       if ( debug ) ::std::cout << "CustomDataTimeContainer::get() curr bearing = " << curr_b * 180.0 /
      M_PI
1742         << "; curr range = " << curr_r << ::std::endl;
1743
1744       CDCMediumCIt itb = it->second.begin();
1745       if ( itb->first == DB_CDATA_ALL_MEDIUM_KEYS ) delta_b = 0;
1746       else {
1747         itb = it->second.lower_bound( curr_b );
1748         if ( itb == it->second.end() ) itb = (++(it->second.rbegin())).base();
1749
1750         delta_b = curr_b - itb->first;
1751         if (delta_b < 0.0) delta_b = -delta_b;
1752         if (delta_b > M_PI) delta_b = 2.0*M_PI - delta_b ;
1753       }
1754
1755       double ort_dist = curr_r * sin(delta_b);
1756       double ort_projection = ::std::sqrt( curr_r*curr_r - ort_dist*ort_dist );
1757
1758       if ( debug ) ::std::cout << "CustomDataTimeContainer::get() nearest bearing = " << itb->first *
      180.0 / M_PI
1759         << "; diff bearing = " << delta_b * 180.0 / M_PI << "; orthog distance = " << ort_dist
1760         << "; orthog range projection = " << ort_projection << ::std::endl;
1761
1762       CDCInnerCIt itr = itb->second.begin();
1763       if ( itr->first == DB_CDATA_ALL_INNER_KEYS ) curr_dist = ort_dist;
1764       else {
1765         itr = itb->second.lower_bound( ort_projection );
1766         if ( itr == itb->second.begin() || itr == itb->second.end() || itr->first == ort_projection ) {
1767           if ( itr == itb->second.end() ) itr = (++(itb->second.rbegin())).base();
```

```
1768          double adj_distance = ::std::abs( ort_projection - itr->first );
1769          curr_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
1770        }
1771        else {
1772          double adj_distance = ::std::abs( ort_projection - itr->first );
1773          double first_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance );
1774
1775          if (debug) ::std::cout << "CustomDataTimeContainer::get() first try, range = " << itr->first
1776            << "; dist = " << first_dist << ::std::endl;
1777
1778          itr--;
1779          adj_distance = ::std::abs( ort_projection - itr->first );
1780          double before_dist = ::std::sqrt( ort_projection*ort_projection + adj_distance*adj_distance
     );
1781
1782          if (debug) ::std::cout << "CustomDataTimeContainer::get() second try, range = " << itr->first
1783            << "; dist = " << before_dist << ::std::endl;
1784
1785          curr_dist = ::std::min( first_dist, before_dist );
1786          if ( curr_dist == first_dist ) itr++;
1787        }
1788      }
1789      if ( debug ) ::std::cout << "CustomDataTimeContainer::get() nearest range = " << itr->first << ";
     distance = " << curr_dist
1790        << "; min distance = " << min_dist << ::std::endl;
1791
1792      if ( curr_dist < min_dist ) {
1793        min_dist = curr_dist;
1794        time_data_ptr = &(itr->second);
1795        if ( curr_dist == 0 ) break;
1796      }
1797
1798    }
1799
1800    if ( time_data_ptr != NULL ) return calculateData( *time_data_ptr, time_key);
1801    return new Data();
1802  }
1803
1804
1805   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
     class InComp >
1806   typename CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
     >::DataFind CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
     >::find( const T& t, double b, double r, const Time& time_key )const {
1807
1808    if (debug) ::std::cout << "CustomDataTimeContainer*::find() << t = " << t << "; b = " << b << "; r = "
1809      << r << "; time_key = " << time_key << ::std::endl;
1810
1811    CDCCIt it = data_map.find( t );
1812    if ( it == data_map.end() ) {
1813
1814      if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() t not found" << ::std::endl;
1815
1816      return DataFind( NULL, false );
1817    }
1818
1819    if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() t found" << ::std::endl;
1820
1821    CDCMediumCIt it2 = it->second.find( b );
1822    if ( it2 == it->second.end() ) {
1823
1824      if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() b not found" << ::std::endl;
1825
1826      return DataFind( NULL, false );
1827    }
1828
1829    if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() b found" << ::std::endl;
1830
1831    CDCInnerCIt it3 = it2->second.find( r );
1832    if ( it3 == it2->second.end() ) {
1833
1834      if ( debug ) ::std::cout << "CustomDataTimeContainer::find() r not found" << ::std::endl;
1835
1836      return DataFind( NULL, false );
1837    }
1838    if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() r found" << ::std::endl;
1839
1840    CDTCTimeCIt it4 = it3->second.find( time_key );
1841    if ( it4 == it3->second.end() ) {
1842
1843      if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() time_key not found" << ::std::endl;
1844
1845      return DataFind( NULL, false );
1846    }
1847
1848    if ( debug ) ::std::cout << "CustomDataTimeContainer*::find() time_key found, Data = " << it4->second
     << ::std::endl;
```

```
1849
1850     return DataFind( it4->second, true );
1851   }
1852
1853
1854   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1855   bool CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::insert(
      Data* d, const T& t, double b, double r, const Time& time_key ) {
1856     DataFind data = find( t, b, r, time_key );
1857     if ( data.second == true ) return false;
1858     data_map[t][b][r][time_key] = d;
1859     return true;
1860   }
1861
1862
1863   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1864   void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::replace(
      Data* d, const T& t, double b, double r, const Time& time_key ) {
1865
1866     if (debug) ::std::cout << "CustomDataTimeContainer*::replace() d = " << *d << "; t = " << t << "; b = "
      << b << "; r = " << r << "; time_key " << time_key << ::std::endl;
1867
1868     DataFind ptr = find( t, b, r, time_key );
1869
1870     if ( ptr.first != NULL ) {
1871       delete ptr;
1872       ptr = d;
1873     }
1874
1875     data_map[t][b][r][time_key] = d;
1876   }
1877
1878
1879   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1880   void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::erase(
      const T& t, double b, double r, const Time& time_key ) {
1881     DataFind ptr = find( t, b, r, time_key );
1882     if ( ptr.first == NULL ) return;
1883     if ( ptr.first != NULL ) delete ptr.first;
1884
1885     data_map[t][b][r].erase(time_key);
1886     if ( data_map[t][b][r].empty() ) data_map[t][b].erase(r);
1887     if ( data_map[t][b].empty() ) data_map[t].erase(b);
1888     if ( data_map[t].empty() ) data_map.erase(t);
1889   }
1890
1891
1892   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1893   void CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp >::clear() {
1894     if ( data_map.empty() ) return;
1895     for ( CDCIt it = data_map.begin(); it != data_map.end(); it++ ) {
1896       for ( CDCMediumIt it2 = it->second.begin(); it2 != it->second.end(); it2++ ) {
1897         for ( CDCInnerIt it3 = it2->second.begin(); it3 != it2->second.end(); it3++ ) {
1898           for ( CDTCTimeCIt it4 = it3->second.begin(); it4 != it3->second.end(); it4++) {
1899             if ( it4->second != NULL ) delete it4->second;
1900           }
1901         }
1902       }
1903     }
1904     data_map.clear();
1905   }
1906
1907
1908   template < class T, class MidFunctor, class InFunctor, class Data, class OutComp, class MidComp,
      class InComp >
1909   Data* CustomDataTimeContainer< T, MidFunctor, InFunctor, Data*, OutComp, MidComp, InComp
      >::calculateData( const TimeData& time_data, const Time& time_key )const {
1910     if (debug) ::std::cout << "CustomDataTimeContainer*::calculateData() time_key = "
1911                            << time_key << "; time_key in time_t = " << (time_t)time_key
1912                            << "; time_data size = " << time_data.size()
1913                            << "; time data min value = " << time_data.begin()->first
1914                            << "; time_data max value = " << time_data.rbegin()->first << ::std::endl;
1915
1916     if ( time_data.empty() ) {
1917       if (debug) ::std::cout << "CustomDataTimeContainer*::calculateData() time_data is empty." <<
      ::std::endl;
1918
1919       return new Data();
1920     }
1921
1922     if ( time_data.size() == 1 ) {
1923       if (debug) ::std::cout << "CustomDataTimeContainer*::calculateData() time_data has size 1.  Data
      cloned from "
```

```
1924                            « time_data.begin()->second « "; ret value " «
      *(time_data.begin()->second) « ::std::endl;
1925
1926        return (time_data.begin()->second)->clone();
1927      }
1928
1929      CDTCTimeCIt fit = time_data.find(time_key);
1930      if ( fit != time_data.end() ) {
1931        if (debug) ::std::cout « "CustomDataTimeContainer*::calculateData() time key found, ret value "
1932                            « *(fit->second) « ::std::endl;
1933
1934        return fit->second->clone();
1935      }
1936
1937      time_t normalized_time = time_key;
1938
1939      CDTCTimeCIt upper_it;
1940      CDTCTimeCIt lower_it;
1941      double alpha, beta;
1942
1943      if ( (normalized_time > time_data.rbegin()->first) || (normalized_time < time_data.begin()->first)
      ) {
1944        normalized_time %= time_data.rbegin()->first - time_data.begin()->first;
1945
1946        if (debug) ::std::cout « "CustomDataTimeContainer*::calculateData() time out of bound, normalized
      on time = "
1947                            « (time_data.rbegin()->first - time_data.begin()->first) « ";
      normalized_time = "
1948                            « normalized_time « "; final time = " « (time_data.begin()->first +
      normalized_time )
1949                            « ::std::endl;
1950
1951        normalized_time += time_data.begin()->first;
1952      }
1953
1954      upper_it = time_data.upper_bound(normalized_time);
1955
1956      if ( upper_it == time_data.begin() ) {
1957        if (debug) ::std::cout « "CustomDataTimeContainer*::calculateData() normalized_time = " «
      normalized_time
1958                            « "; upper_it time is first in time data = " « upper_it->first
1959                            « "; ret value = " « *(upper_it->second) « ::std::endl;
1960
1961        return (upper_it->second)->clone();
1962      }
1963      lower_it = upper_it;
1964      --lower_it;
1965
1966      alpha = ( ::std::abs((double)(upper_it->first - normalized_time)) / ::std::abs((double)(
      upper_it->first - lower_it->first )) );
1967      beta = ( ::std::abs((double)(normalized_time - lower_it->first)) / ::std::abs((double)(
      upper_it->first - lower_it->first )) );
1968
1969      if (debug) ::std::cout « "CustomDataTimeContainer*::calculateData() normalized_time = " «
      normalized_time
1970                          « "; lower_it time = " « lower_it->first
1971                          « "; upper_it time = " « upper_it->first
1972                          « "; alpha = " « alpha « "; beta " « beta « ::std::endl;
1973
1974      Data* ret_val = (lower_it->second)->clone();
1975
1976      *ret_val = (*ret_val) * alpha + (*(upper_it->second)) * beta;
1977
1978      if (debug) ::std::cout « "CustomDataTimeContainer*::calculateData() return value = "
1979                          « *ret_val « ::std::endl;
1980
1981      return ret_val;
1982    }
1983
1984
1985 }
1986
1987
1988 #endif /* WOSS_DB_CUSTOM_DATA_CONTAINER_H */
1989
```

14.100 woss/woss_db/woss-db-manager.cpp File Reference

Provides the implementation of woss::WossDbManager class.

14.100.1 Detailed Description

Provides the implementation of woss::WossDbManager class.

Author

Federico Guerra

Provides the implementation of the woss::WossDbManager class

14.101 woss/woss_db/woss-db-manager.h File Reference

Provides the interface for woss::WossDbManager class.

Classes

- class woss::WossDbManager
  Abstraction layer for database and data manipulation.
- class woss::WossDbManager::BearingOperator
  Bearing operator function object.
- class woss::WossDbManager::RangeOperator
  Range operator function object.

14.101.1 Detailed Description

Provides the interface for woss::WossDbManager class.

Author

Federico Guerra

Provides the interface for the woss::WossDbManager class

14.102 woss-db-manager.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

```
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DB_MANAGER_IMPLEMENT_H
41 #define WOSS_DB_MANAGER_IMPLEMENT_H
42
43
44 #include "woss-db-custom-data-container.h"
45 #include <coordinates-definitions.h>
46 #include <sediment-definitions.h>
47 #include <altimetry-definitions.h>
48 #include <ssp-definitions.h>
49 #include <definitions.h>
50
51
52 namespace woss {
53
54
55   class WossBathymetryDb;
56
57   class WossSedimentDb;
58
59   class WossSSPDb;
60
61   class WossResTimeArrDb;
62
63   class WossResPressDb;
64
65   class Time;
66
67   class Pressure;
68
69   class TimeArr;
70
71
84   class WossDbManager {
85
86
87     protected:
88
89
95     class BearingOperator {
96
97
98       public:
99
100
110       double operator()( const Coord& x, const Coord& y )const {
111         return( x.getInitialBearing(y) );
112       }
113
114     };
115
116
122     class RangeOperator {
123
124
125       public:
126
127
137       double operator()( const Coord& x, const Coord& y )const {
138         return( x.getGreatCircleDistance(y) );
139       }
140
141     };
142
143
144     public:
145
146
147     typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Bathymetry > CCBathymetry;
148
149     typedef CustomDataTimeContainer< Coord, BearingOperator, RangeOperator, SSP* > CCSSP;
150
151     typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Sediment* > CCSediment;
152
153     typedef CustomDataContainer< Coord, BearingOperator, RangeOperator, Altimetry* > CCAltimetry;
154
155
159     WossDbManager();
```

```
160
164     WossDbManager( WossDbManager& instance );
165
169     WossDbManager& operator=( WossDbManager& instance );
170
175     virtual ~WossDbManager();
176
177
185     virtual Altimetry* getAltimetry( const CoordZ& tx, const CoordZ& rx ) const;
186
187
195     virtual Sediment* getSediment( const CoordZ& tx, const CoordZ& rx ) const;
196
204     virtual Sediment* getSediment( const CoordZ& tx, const CoordZVector& rx_coordz_vector ) const;
205
206
212     virtual Bathymetry getBathymetry( const Coord& tx, const Coord& rx ) const;
213
218     virtual void getBathymetry( const Coord& tx, CoordZVector& rx_coordz_vector ) const;
219
220
229     virtual SSP* getSSP( const Coord& tx, const Coord& rx, const Time& time, long double
    ssp_depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const;
230
242     virtual SSP* getAverageSSP( const Coord& tx, const Coord& rx, const Time& time_start, const Time&
    time_end, int max_time_values, long double ssp_depth_precision = SSP_CUSTOM_DEPTH_PRECISION ) const;
243
244
255     virtual TimeArr* getTimeArr( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
    const Time& time_value ) const;
256
265     virtual void insertTimeArr( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
    const Time& time_value, const TimeArr& channel ) const;
266
267
278     virtual Pressure* getPressure( const CoordZ& coord_tx, const CoordZ& coord_rx, const double
    frequency, const Time& time_value ) const;
279
288     virtual void insertPressure( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
    const Time& time_value, const Pressure& pressure ) const;
289
290
296     WossDbManager& setBathymetryDb( WossBathymetryDb* ptr ) { bathymetry_db = ptr; return *this; }
297
303     WossDbManager& setSedimentDb( WossSedimentDb* ptr ) { sediment_db = ptr; return *this; }
304
310     WossDbManager& setSSPDb( WossSSPDb* ptr ) { ssp_db = ptr; return *this; }
311
317     WossDbManager& setResTimeArrDb( WossResTimeArrDb* ptr ) { results_arrivals_db = ptr; return *this; }
318
324     WossDbManager& setResPressureDb( WossResPressDb* ptr ) { results_pressure_db = ptr; return *this; }
325
326
334     bool setCustomAltimetry( Altimetry* const altimetry, const Coord& tx_coord =
    CCAltimetry::DB_CDATA_ALL_OUTER_KEYS,
335                              double bearing = CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS,
336                              double range = CCAltimetry::DB_CDATA_ALL_INNER_KEYS );
337
345     Altimetry* getCustomAltimetry( const Coord& tx_coord = CCAltimetry::DB_CDATA_ALL_OUTER_KEYS,
346                                    double bearing = CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS,
347                                    double range = CCAltimetry::DB_CDATA_ALL_INNER_KEYS );
348
356     WossDbManager& eraseCustomAltimetry( const Coord& tx_coord = CCAltimetry::DB_CDATA_ALL_OUTER_KEYS,
357                                          double bearing = CCAltimetry::DB_CDATA_ALL_MEDIUM_KEYS,
358                                          double range = CCAltimetry::DB_CDATA_ALL_INNER_KEYS );
359
360
369     bool setCustomSediment( Sediment* const sediment, const Coord& tx_coord =
    CCSediment::DB_CDATA_ALL_OUTER_KEYS,
370                             double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
371                             double range = CCSediment::DB_CDATA_ALL_INNER_KEYS );
372
380     Sediment* getCustomSediment( const Coord& tx_coord = CCSediment::DB_CDATA_ALL_OUTER_KEYS,
381                                  double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
382                                  double range = CCSediment::DB_CDATA_ALL_INNER_KEYS );
383
391     WossDbManager& eraseCustomSediment( const Coord& tx_coord = CCSediment::DB_CDATA_ALL_OUTER_KEYS,
392                                         double bearing = CCSediment::DB_CDATA_ALL_MEDIUM_KEYS,
393                                         double range = CCSediment::DB_CDATA_ALL_INNER_KEYS );
394
395
405     bool setCustomSSP( SSP* const ssp, const Coord& tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
406                                        double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
407                                        double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
408                                        const Time& time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS );
409
431     virtual bool importCustomSSP( const ::std::string& filename, const Time& =
```

```
    CCSSP::DB_CDATA_ALL_TIME_KEYS,
432                                     const Coord& tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS, double
    bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS );
433
442     SSP* getCustomSSP( const Coord& tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
443                        double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
444                        double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
445                        const Time& time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS );
446
455     WossDbManager& eraseCustomSSP( const Coord& tx_coord = CCSSP::DB_CDATA_ALL_OUTER_KEYS,
456                                    double bearing = CCSSP::DB_CDATA_ALL_MEDIUM_KEYS,
457                                    double range = CCSSP::DB_CDATA_ALL_INNER_KEYS,
458                                    const Time& time_value = CCSSP::DB_CDATA_ALL_TIME_KEYS );
459
468     bool setCustomBathymetry( Bathymetry* const bathymetry, const Coord& tx_coord =
    CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
469                               double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
470                               double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS );
471
479     virtual bool importCustomBathymetry( const ::std::string& filename, const Coord& tx_coord =
    CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
480                                          double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS );
481
489     Bathymetry* getCustomBathymetry( const Coord& tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
490                                      double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
491                                      double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS );
492
500     WossDbManager& eraseCustomBathymetry( const Coord& tx_coord = CCBathymetry::DB_CDATA_ALL_OUTER_KEYS,
501                                           double bearing = CCBathymetry::DB_CDATA_ALL_MEDIUM_KEYS,
502                                           double range = CCBathymetry::DB_CDATA_ALL_INNER_KEYS );
503
504
505     WossDbManager& setDebug( bool flag ) { debug = flag; updateDebugFlag(); return *this; }
506
507     bool getDebug() { return debug; }
508
509
510     protected:
511
512
516     WossBathymetryDb* bathymetry_db;
517
521     WossSedimentDb* sediment_db;
522
526     WossSSPDb* ssp_db;
527
531     WossResTimeArrDb* results_arrivals_db;
532
536     WossResPressDb* results_pressure_db;
537
538
542     bool debug;
543
544
548     CCBathymetry ccbathy_map;
549
553     CCSediment ccsediment_map;
554
558     CCSSP ccssp_map;
559
563     CCAltimetry ccaltimetry_map;
564
568     virtual bool closeAllConnections();
569
570     virtual void updateDebugFlag();
571
572   };
573
574
575   //inline functions
577   inline bool WossDbManager::setCustomAltimetry( Altimetry* const altimetry, const Coord& tx_coord,
    double bearing, double range ) {
578     return ccaltimetry_map.insert( altimetry, tx_coord, bearing, range );
579   }
580
581
582   inline Altimetry* WossDbManager::getCustomAltimetry( const Coord& tx_coord, double bearing, double
    range ) {
583     return ccaltimetry_map.get( tx_coord, bearing, range );
584   }
585
586
587   inline WossDbManager& WossDbManager::eraseCustomAltimetry( const Coord& tx_coord, double bearing,
    double range ) {
588     ccaltimetry_map.erase( tx_coord, bearing, range );
589     return *this;
590   }
```

```
591
592
593   inline bool WossDbManager::setCustomSediment( Sediment* const sediment, const Coord& tx_coord, double
      bearing, double range ) {
594     return ccsediment_map.insert( sediment, tx_coord, bearing, range );
595   }
596
597
598   inline Sediment* WossDbManager::getCustomSediment( const Coord& tx_coord, double bearing, double range
      ) {
599     return ccsediment_map.get( tx_coord, bearing, range );
600   }
601
602
603   inline WossDbManager& WossDbManager::eraseCustomSediment( const Coord& tx_coord, double bearing,
      double range ) {
604     ccsediment_map.erase( tx_coord, bearing, range );
605     return *this;
606   }
607
608
609   inline bool WossDbManager::setCustomSSP( SSP* const ssp, const Coord& tx_coord, double bearing, double
      range, const Time& time_key ) {
610     return ccssp_map.insert( ssp, tx_coord, bearing, range, time_key );
611   }
612
613
614   inline SSP* WossDbManager::getCustomSSP( const Coord& tx_coord, double bearing, double range, const
      Time& time_key ) {
615     return ccssp_map.get( tx_coord, bearing, range, time_key );
616   }
617
618
619 inline WossDbManager& WossDbManager::eraseCustomSSP( const Coord& tx_coord, double bearing, double
      range, const Time& time_key ) {
620     ccssp_map.erase( tx_coord, bearing, range, time_key );
621     return *this;
622   }
623
624
625   inline bool WossDbManager::setCustomBathymetry( Bathymetry* const bathy, const Coord& tx_coord, double
      bearing, double range ) {
626     return ccbathy_map.insert(*bathy, tx_coord, bearing, range );
627   }
628
629
630   inline Bathymetry* WossDbManager::getCustomBathymetry( const Coord& tx_coord, double bearing, double
      range ) {
631     return const_cast< Bathymetry* >( ccbathy_map.get( tx_coord, bearing, range ) );
632   }
633
634
635   inline WossDbManager& WossDbManager::eraseCustomBathymetry( const Coord& tx_coord, double bearing,
      double range ) {
636     ccbathy_map.erase( tx_coord, bearing, range );
637     return *this;
638   }
639
640
641 }
642
643
644 #endif /* WOSS_DB_MANAGER_IMPLEMENT_H */
645
646
647
```

14.103 woss/woss_db/woss-db.cpp File Reference

Provides the implementation of woss::WossDb class.

14.103.1 Detailed Description

Provides the implementation of woss::WossDb class.

Author

Federico Guerra

Provides the implementation of woss::WossDb class

14.104 woss/woss_db/woss-db.h File Reference

Provides the interface for woss::WossDb class.

Classes

- class woss::WossDb

  Abstract class that provides the interface of databases.
- class woss::WossNetcdfDb

  NetCDF implementation of WossDb.
- class woss::WossTextualDb

  Textual implementation of WossDb.
- class woss::WossBathymetryDb

  Data behaviour class for bathymetry database.
- class woss::WossSedimentDb

  Data behaviour class for Sediment database.
- class woss::WossSSPDb

  Data behaviour class for SSP database.
- class woss::WossResTimeArrDb

  Data behaviour class for storing calculated TimeArr.
- class woss::WossResPressDb

  Data behaviour class for storing calculated Pressure.

Typedefs

- typedef ::std::pair< ::std::string, ::std::string > woss::PathName

14.104.1 Detailed Description

Provides the interface for woss::WossDb class.

Author

Federico Guerra

Provides the interface for the woss::WossDb class

14.104.2 Typedef Documentation

14.104.2.1 PathName `typedef ::std::pair< ::std::string, ::std::string > woss::PathName`

A pair of strings containing:

Parameters

first	pathname
second	filename

14.105 woss-db.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DB_H
41 #define WOSS_DB_H
42
43 #include <fstream>
44
45 #if defined(WOSS_NETCDF4_SUPPORT)
46 #include <ncFile.h>
47 #elif defined(WOSS_NETCDF_SUPPORT)
48 #include <netcdfcpp.h>
49 #endif
50
51 #include <definitions.h>
52 #include <coordinates-definitions.h>
53
54
55 namespace woss {
56
57
58   class Sediment;
59   class Pressure;
60   class SSP;
61   class TimeArr;
62   class Time;
63
64
68 #define DB_NAME_NOT_SET "DB_NAME_NOT_SET"
69
70
76   typedef ::std::pair< ::std::string, ::std::string > PathName; // path , filename
77
78
89   class WossDb {
90
91
92     public:
93
94
99     WossDb( const ::std::string& name );
100
101     virtual ~WossDb() { }
102
103
```

```
108     void setDbName( const ::std::string& pathname ) { db_name = pathname; }
109
114     ::std::string getDbName()const { return db_name; }
115
116
121     void setDebug( double flag = true ) { debug = flag; }
122
127     bool isUsingDebug()const { return debug; }
128
129
134     virtual bool isValid() { return( db_name.length() > 0 ); }
135
136
141     virtual bool openConnection() = 0;
142
143
148     virtual bool finalizeConnection() = 0;
149
150
155     virtual bool closeConnection() = 0;
156
157
158     protected:
159
160
164     ::std::string db_name;
165
166
170     bool debug;
171
172
178     PathName getPathName( const ::std::string& complete_path );
179
180
181   };
182
184 #ifdef WOSS_NETCDF_SUPPORT
193   class WossNetcdfDb :  public WossDb {
194
195
196     public:
197
198
203     WossNetcdfDb( const ::std::string& name );
204
205     virtual ~WossNetcdfDb();
206
207
212     virtual bool openConnection();
213
214
219     virtual bool closeConnection();
220
221
222     protected:
223
224
228 #if defined(WOSS_NETCDF4_SUPPORT)
229     netCDF::NcFile* netcdf_db;
230 #else
231     NcFile* netcdf_db;
232 #endif // defined(WOSS_NETCDF4_SUPPORT)
233
234   };
235 #endif // WOSS_NETCDF_SUPPORT
236
237
245   class WossTextualDb :  public WossDb {
246
247
248     public:
249
254     WossTextualDb( const ::std::string& name );
255
256     virtual ~WossTextualDb();
257
258
263     virtual bool openConnection();
264
265
270     virtual bool closeConnection();
271
272
273     protected:
274
275
279     ::std::fstream textual_db;
```

```
280
281
282   };
283
285
292   class WossBathymetryDb {
293
294
295     public:
296
297
298     WossBathymetryDb() { }
299
300     virtual ~WossBathymetryDb() { }
301
302
309     virtual bool insertValue( const Coord& coordinates, const Bathymetry& bathymetry_value ) = 0;
310
316     virtual Bathymetry getValue( const Coord& coords ) const = 0;
317
318
319   };
320
321
328   class WossSedimentDb {
329
330
331     public:
332
333
334     WossSedimentDb() { }
335
336     virtual ~WossSedimentDb() { }
337
338
345     virtual bool insertValue( const Coord& coordinates, const Sediment& sediment_value ) = 0;
346
347
354     virtual Sediment* getValue( const CoordZ& coords ) const = 0;
355
363     virtual Sediment* getValue( const CoordZVector& coordz_vector ) const = 0;
364
365
366   };
367
368
375   class WossSSPDb {
376
377
378     public:
379
380
381     WossSSPDb() { }
382
383     virtual ~WossSSPDb() { }
384
385
393     virtual bool insertValue( const Coord& coordinates, const Time& time_value, const SSP& ssp_value ) =
      0;
394
395
404     virtual SSP* getValue( const Coord& coords, const Time& time, long double ssp_depth_precision )
      const = 0;
405
406
407   };
408
409
416   class WossResTimeArrDb {
417
418
419     public:
420
421
422     WossResTimeArrDb() { }
423
424     virtual ~WossResTimeArrDb() { }
425
435     virtual TimeArr* getValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value ) const = 0;
436
437
446     virtual bool insertValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value, const TimeArr& channel ) = 0;
447
448
449   };
```

```
450
451
458   class WossResPressDb {
459
460
461     public:
462
463
464     WossResPressDb() {  }
465
466     virtual ~WossResPressDb() { }
467
468
477     virtual Pressure* getValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value ) const = 0;
478
479
488     virtual bool insertValue( const CoordZ& coord_tx, const CoordZ& coord_rx, const double frequency,
      const Time& time_value,  const Pressure& pressure ) = 0;
489
490
491   };
492
493 }
494
495
496 #endif /* WOSS_DB_H */
497
498
```

14.106 woss/woss_def/altimetry-definitions.cpp File Reference

Implementations and library for woss::Altimetry class.

14.106.1 Detailed Description

Implementations and library for woss::Altimetry class.

Author

Federico Guerra

Implementations and library for woss::Altimetry class

14.107 woss/woss_def/altimetry-definitions.h File Reference

Definitions and library for woss::Altimetry class.

Classes

- class woss::Altimetry

  Altimetry profile class.
- class woss::AltimBretschneider

  AltimBretschneider supports Bretschneider wave model.

Typedefs

- typedef std::map< PDouble, double > woss::AltimetryMap
- typedef AltimetryMap::iterator **woss::AltIt**
- typedef AltimetryMap::const_iterator **woss::AltCIt**
- typedef AltimetryMap::reverse_iterator **woss::AltRIt**
- typedef AltimetryMap::const_reverse_iterator **woss::AltCRIt**

Functions

- const Altimetry woss::operator+ (const Altimetry &left, const Altimetry &right)
- const Altimetry woss::operator- (const Altimetry &left, const Altimetry &right)
- const Altimetry woss::operator+ (const Altimetry &left, const double right)
- const Altimetry woss::operator- (const Altimetry &left, const double right)
- const Altimetry woss::operator/ (const Altimetry &left, const double right)
- const Altimetry woss::operator* (const Altimetry &left, const double right)
- const Altimetry woss::operator+ (const double left, const Altimetry &right)
- const Altimetry woss::operator- (const double left, const Altimetry &right)
- const Altimetry woss::operator/ (const double left, const Altimetry &right)
- const Altimetry woss::operator* (const double left, const Altimetry &right)
- Altimetry & woss::operator+= (Altimetry &left, const Altimetry &right)
- Altimetry & woss::operator-= (Altimetry &left, const Altimetry &right)
- Altimetry & woss::operator+= (Altimetry &left, double right)
- Altimetry & woss::operator-= (Altimetry &left, double right)
- Altimetry & woss::operator/= (Altimetry &left, double right)
- Altimetry & woss::operator*= (Altimetry &left, double right)
- std::ostream & woss::operator<< (std::ostream &os, const Altimetry &instance)
- bool woss::operator== (const Altimetry &left, const Altimetry &right)
- bool woss::operator!= (const Altimetry &left, const Altimetry &right)

14.107.1 Detailed Description

Definitions and library for woss::Altimetry class.

Author

Federico Guerra

Definitions and library for woss::Altimetry class

14.107.2 Typedef Documentation

14.107.2.1 AltimetryMap `typedef std::map< PDouble , double > woss::AltimetryMap`

Map that links a PDouble range [m] to a double altimetry value. Negative values ==> under water. Positive values ==> above water.

14.107.3 Function Documentation

14.107.3.1 operator"!=() `bool woss::operator!= ( const Altimetry & left, const Altimetry & right ) [inline]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.107.3.2 operator∗() [1/2]

```
const Altimetry woss::operator* (
    const Altimetry & left,
    const double right )
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

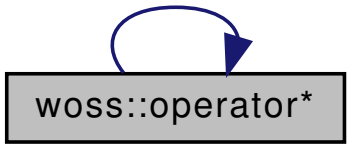
Returns

a new const instance holding the operation result

References woss::operator∗().

Referenced by woss::operator∗().

Here is the call graph for this function:



14.107.3.3 operator∗() [2/2]

```
const Altimetry woss::operator* (
    const double left,
    const Altimetry & right )
```

Multiplication operator

Parameters

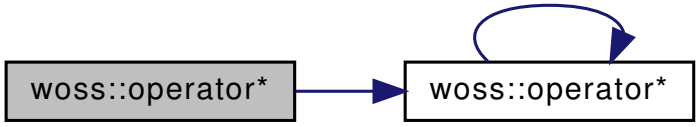
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator∗().

Here is the call graph for this function:



14.107.3.4 operator∗=() `Altimetry & woss::operator*= ( Altimetry & left, double right )`

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

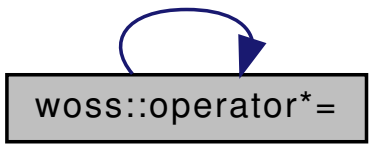
Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator∗=().

Referenced by woss::operator∗=().

Here is the call graph for this function:



14.107.3.5 operator+() [1/3]

```
const Altimetry woss::operator+ (
    const Altimetry & left,
    const Altimetry & right )
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

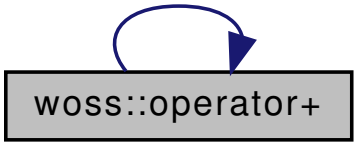
Returns

a new const instance holding the operation result

References woss::operator+().

Referenced by woss::operator+().

Here is the call graph for this function:



14.107.3.6 operator+() [2/3]

```
const Altimetry woss::operator+ (
    const Altimetry & left,
    const double right )
```

Sum operator

Parameters

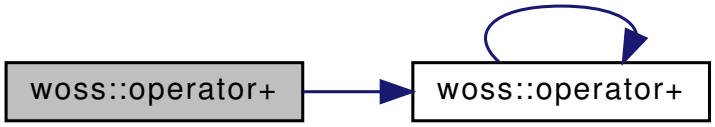
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator+().

Here is the call graph for this function:



14.107.3.7 operator+() [3/3] const Altimetry woss::operator+ (const double *left*, const Altimetry & *right*)

Sum operator

Parameters

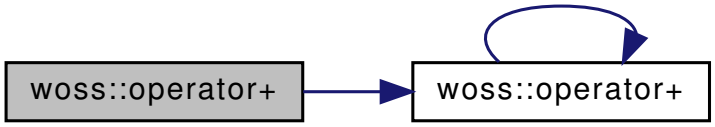
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator+().

Here is the call graph for this function:



14.107.3.8 operator+=() [1/2] Altimetry & woss::operator+= (Altimetry & *left*, const Altimetry & *right*)

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

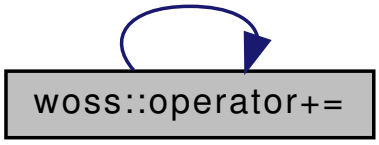
Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator+=().

Referenced by woss::operator+=().

Here is the call graph for this function:



14.107.3.9 operator+=() [2/2] Altimetry & woss::operator+= (Altimetry & *left*, double *right*)

Compound assignment sum operator

Parameters

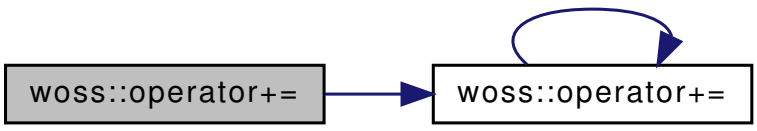
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator+=().

Here is the call graph for this function:



14.107.3.10 operator-() [1/3] const Altimetry woss::operator- (const Altimetry & *left*, const Altimetry & *right*)

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

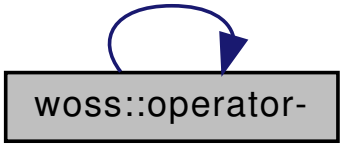
Returns

a new const instance holding the operation result

References woss::operator-().

Referenced by woss::operator-().

Here is the call graph for this function:



14.107.3.11 operator-() [2/3] const Altimetry woss::operator- (const Altimetry & *left*, const double *right*)

Subtraction operator

Parameters

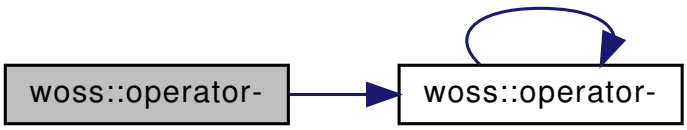
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator-().

Here is the call graph for this function:



14.107.3.12 operator-() [3/3]

```
const Altimetry woss::operator- (
    const double left,
    const Altimetry & right )
```

Subtraction operator

Parameters

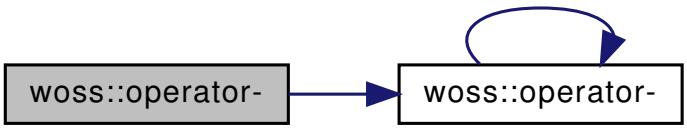
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator-().

Here is the call graph for this function:



14.107.3.13 operator-=() [1/2]

```
Altimetry & woss::operator-= (
    Altimetry & left,
    const Altimetry & right )
```

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

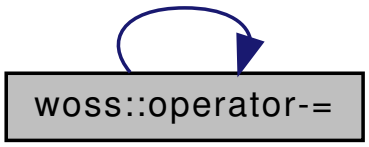
Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator-=().

Referenced by woss::operator-=().

Here is the call graph for this function:



14.107.3.14 operator-=() [2/2] Altimetry & woss::operator-= (Altimetry & *left*, double *right*)

Compound assignment subtraction operator

Parameters

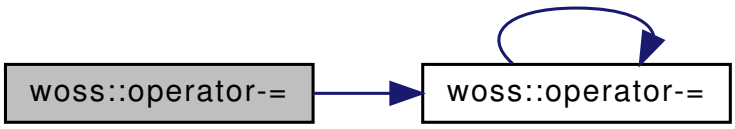
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator-=().

Here is the call graph for this function:



14.107.3.15 operator/() [1/2] const Altimetry woss::operator/ (const Altimetry & *left*, const double *right*)

Division operator

Parameters

left	left operand const reference
right	right operand const reference

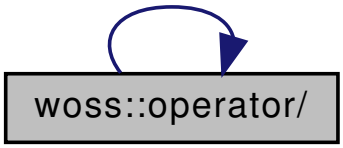
Returns

a new const instance holding the operation result

References woss::operator/().

Referenced by woss::operator/().

Here is the call graph for this function:



14.107.3.16 operator/() [2/2]

```
const Altimetry woss::operator/ (
    const double left,
    const Altimetry & right )
```

Division operator

Parameters

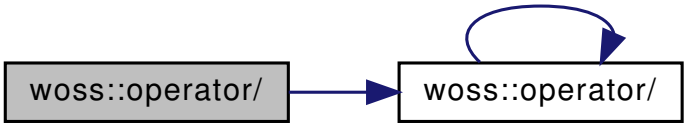
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator/().

Here is the call graph for this function:



14.107.3.17 operator/=()

```
Altimetry & woss::operator/= (
    Altimetry & left,
    double right )
```

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

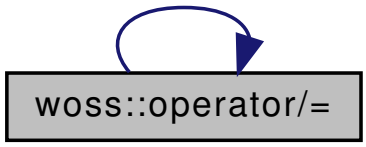
Returns

left reference after the operation

References woss::Altimetry::altimetry_map, and woss::operator/=().

Referenced by woss::operator/=().

Here is the call graph for this function:



14.107.3.18 operator<<() std::ostream & woss::operator<< (std::ostream & *os*, const Altimetry & *instance*) [inline]

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Pressure reference

Returns

os reference after the operation

14.107.3.19 operator==() bool woss::operator== (const Altimetry & *left*, const Altimetry & *right*) [inline]

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.108 altimetry-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef ALTIMETRY_DEFINITIONS_H
41 #define ALTIMETRY_DEFINITIONS_H
42
43
44 #include <cassert>
45 #include <climits>
46 #include <map>
47 #include "custom-precision-double.h"
48 #include "time-definitions.h"
49
50
51 namespace woss {
52
53
59   typedef std::map < PDouble , double > AltimetryMap;
60   typedef AltimetryMap::iterator AltIt;
61   typedef AltimetryMap::const_iterator AltCIt;
62   typedef AltimetryMap::reverse_iterator AltRIt;
63   typedef AltimetryMap::const_reverse_iterator AltCRIt;
64
65
69 #define ALTIMETRY_RANGE_STEPS (1)
70
71
78   class Altimetry {
79
80
81     public:
82
83
87     Altimetry();
88
93     Altimetry( AltimetryMap& map );
94
100     Altimetry( double range, double altimetry );
101
106     Altimetry( const Altimetry& copy );
107
108     virtual ~Altimetry() { }
109
110
115     virtual Altimetry* create()const {
116       return &((new Altimetry() )->updateMinMaxAltimetryValues()); ; }
117
123     virtual Altimetry* create( AltimetryMap& map )const {
124       return &((new Altimetry( map ) )->updateMinMaxAltimetryValues()); }
125
```

```
131     virtual Altimetry* create( const Altimetry& copy )const { return new Altimetry( copy ); }
132
137     virtual Altimetry* clone()const { return new Altimetry(*this); }
138
143     static AltimetryMap& createNotValid();
144
149     static AltimetryMap& createFlat(double altimetry = 0 );
150
151
158     Altimetry& insertValue( double range, double altimetry );
159
165     Altimetry& sumValue( double range, double altimetry );
166
167
173     AltCIt findValue( double range )const { return( altimetry_map.find(range) ); }
174
175
181     Altimetry& eraseValue( double range ) { altimetry_map.erase(range); return *this; }
182
183
191     virtual Altimetry* crop( double range_start, double range_end );
192
198     virtual Altimetry* randomize( double ratio_incr_value ) const;
199
204     AltCIt begin()const { return altimetry_map.begin(); }
205
210     AltCIt end()const { return altimetry_map.end(); }
211
217     AltCIt at( const int i ) const ;
218
223     int size()const { return altimetry_map.size(); }
224
229     bool empty()const { return altimetry_map.empty(); }
230
231
235     void clear() { altimetry_map.clear(); }
236
237
242     static void setDebug( bool flag ) { debug = flag; }
243
248     static bool getDebug( bool flag ) { return debug; }
249
254     Altimetry& setEvolutionTimeQuantum( double quantum ) { evolution_time_quantum = quantum;  return
    *this;}
255
260     Altimetry& setTotalRangeSteps( int r_s ) { total_range_steps = r_s;  return *this;}
261
266     Altimetry& setRange( double r ) { range = r;  return *this; }
267
272     Altimetry& setDepth( double d) { depth = d; return *this; }
273
274
279     double getMaxRangeValue()const { return( altimetry_map.rbegin()->first ); }
280
285     double getMinRangeValue()const { return( altimetry_map.begin()->first ); }
286
291     double getMaxAltimetryValue()const { return( max_altimetry_value ); }
292
297     double getMinAltimetryValue()const { return( min_altimetry_value ); }
298
303     long double getRangePrecision()const { return range_precision; }
304
309     double getEvolutionTimeQuantum()const { return evolution_time_quantum; }
310
315     double getRange()const { return range; }
316
321     int getTotalRangeSteps()const { return total_range_steps; }
322
327     double getDepth()const { return depth; }
328
329
334     virtual bool isValid() const;
335
341     virtual bool initialize();
342
348     virtual Altimetry* timeEvolve( const Time& time_value );
349
350
356     Altimetry& operator=( const Altimetry& copy ) ;
357
358
365     friend bool operator==( const Altimetry& left, const Altimetry& right ) ;
366
373     friend bool operator!=( const Altimetry& left, const Altimetry& right );
374
375
382     friend const Altimetry operator+( const Altimetry& left, const Altimetry& right );
```

```
383
390     friend const Altimetry operator-( const Altimetry& left, const Altimetry& right );
391
392
399     friend const Altimetry operator+( const Altimetry& left, const double right );
400
407     friend const Altimetry operator-( const Altimetry& left, const double right );
408
415     friend const Altimetry operator/( const Altimetry& left, const double right );
416
423     friend const Altimetry operator*( const Altimetry& left, const double right );
424
425
432     friend const Altimetry operator+( const double left, const Altimetry& right );
433
440     friend const Altimetry operator-( const double left, const Altimetry& right );
441
448     friend const Altimetry operator/( const double left, const Altimetry& right );
449
456     friend const Altimetry operator*( const double left, const Altimetry& right );
457
458
465     friend Altimetry& operator+=( Altimetry& left, const Altimetry& right );
466
473     friend Altimetry& operator-=( Altimetry& left, const Altimetry& right );
474
475
482     friend Altimetry& operator+=( Altimetry& left, double right );
483
490     friend Altimetry& operator-=( Altimetry& left, double right );
491
498     friend Altimetry& operator/=( Altimetry& left, double right );
499
506     friend Altimetry& operator*=( Altimetry& left, double right );
507
508
515     friend std::ostream& operator«( std::ostream& os, const Altimetry& instance );
516
517
518     protected:
519
520
521     virtual Altimetry& updateMinMaxAltimetryValues();
522
526     static bool debug;
527
531     double range;
532
536     int total_range_steps;
537
541     double min_altimetry_value;
542
546     double max_altimetry_value;
547
551     long double range_precision;
552
556     Time last_evolution_time;
557
561     double evolution_time_quantum;
562
566     double depth;
567
571     AltimetryMap altimetry_map;
572
573
574   };
575
576
582   class AltimBretschneider :  public Altimetry {
583
584     public:
585
589     AltimBretschneider();
590
595     AltimBretschneider( AltimetryMap& map );
596
604     AltimBretschneider( double ch_height, double avg_per, int total_range_steps, double depth );
605
610     AltimBretschneider( const AltimBretschneider& copy );
611
612     virtual ~AltimBretschneider() { }
613
614
619     virtual AltimBretschneider* create()const {
620       AltimBretschneider* ret_val = new AltimBretschneider();
621       ret_val->updateMinMaxAltimetryValues();
622       return ret_val; }
```

```
623
629     virtual AltimBretschneider* create( AltimetryMap& map )const {
630       AltimBretschneider* ret_val = new AltimBretschneider( map );
631       ret_val->updateMinMaxAltimetryValues();
632       return ret_val;  }
633
639     virtual AltimBretschneider* create( const AltimBretschneider& copy )const {
640       return new AltimBretschneider( copy ); }
641
647     virtual AltimBretschneider* create( const Altimetry& copy )const {
648       AltimBretschneider const* dyn_ptr = dynamic_cast<AltimBretschneider const*>(&copy);
649       if (dyn_ptr) {
650         return new AltimBretschneider( *dyn_ptr );
651       }
652       return create();
653     }
654
663     virtual AltimBretschneider* create( double ch_height, double avg_per, int total_range_steps, double
    depth )const {
664       return new AltimBretschneider( ch_height, avg_per, total_range_steps, depth);
665     }
666
671     virtual AltimBretschneider* clone()const { return new AltimBretschneider(*this); }
672
673
679     AltimBretschneider& operator=( const AltimBretschneider& copy ) ;
680
681
687     virtual bool initialize();
688
689
694     virtual bool isValid() const;
695
696
702     virtual AltimBretschneider* timeEvolve( const Time& time_value );
703
704
710     virtual AltimBretschneider* randomize( double ratio_incr_value ) const;
711
717     AltimBretschneider& setCharacteristicHeight( double h ) { char_height = h; return *this; }
718
724     AltimBretschneider& setAveragePeriod( double p ) { average_period = p; return *this; }
725
730     double getCharacteristicHeight()const { return char_height; }
731
736     double getAveragePeriod()const { return average_period; }
737
738
739     protected:
740
741
742     virtual AltimBretschneider& createWaveSpectrum();
743
749     double char_height;
750
756     double average_period;
757
758   };
759
760   //non-inline operator declarations
762   const Altimetry operator+( const Altimetry& left, const Altimetry& right );
763
764   const Altimetry operator-( const Altimetry& left, const Altimetry& right );
765
766
767   const Altimetry operator+( const Altimetry& left, const double right );
768
769   const Altimetry operator-( const Altimetry& left, const double right );
770
771   const Altimetry operator/( const Altimetry& left, const double right );
772
773   const Altimetry operator*( const Altimetry& left, const double right );
774
775
776   const Altimetry operator+( const double left, const Altimetry& right );
777
778   const Altimetry operator-( const double left, const Altimetry& right );
779
780   const Altimetry operator/( const double left, const Altimetry& right );
781
782   const Altimetry operator*( const double left, const Altimetry& right );
783
784
785   Altimetry& operator+=( Altimetry& left, const Altimetry& right );
786
787   Altimetry& operator-=( Altimetry& left, const Altimetry& right );
788
```

```
789   Altimetry& operator+=( Altimetry& left, double right );
790
791
792   Altimetry& operator-=( Altimetry& left, double right );
793
794   Altimetry& operator/=( Altimetry& left, double right );
795
796   Altimetry& operator*=( Altimetry& left, double right );
797
799   //inline functions
800   inline AltimetryMap& Altimetry::createNotValid() {
801     static AltimetryMap altimetry_map;
802     altimetry_map.clear();
803     return altimetry_map;
804   }
805
806
807   inline AltimetryMap& Altimetry::createFlat( double altimetry ) {
808     static AltimetryMap altimetry_map;
809     altimetry_map.clear();
810     altimetry_map[0.0] = altimetry;
811     return altimetry_map;
812   }
813
814
815   inline Altimetry& Altimetry::sumValue( double range, double altimetry ) {
816     if ( altimetry < min_altimetry_value ) min_altimetry_value = altimetry;
817     if ( altimetry > max_altimetry_value ) max_altimetry_value = altimetry;
818
819     AltIt it = altimetry_map.find( range );
820     if ( it == altimetry_map.end() ) altimetry_map.insert( std::make_pair( range, altimetry ) );
821     else it->second += altimetry;
822     return *this;
823   }
824
825
826   inline std::ostream& operator«( std::ostream& os, const Altimetry& instance ) {
827     if ( !instance.altimetry_map.empty() ) {
828       os « "size = " « instance.altimetry_map.size() « "; min range = " «
      instance.altimetry_map.begin()->first
829                       « "; altimetry = " « instance.begin()->second
830                       « "; max range = " « instance.altimetry_map.rbegin()->first
831                       « "; altimetry = " « instance.altimetry_map.rbegin()->second;
832     }
833     else {
834       os « "is empty.";
835     }
836     return os;
837   }
838
839
840   inline bool operator==( const Altimetry& left, const Altimetry& right ) {
841     if ( &left == &right ) return true;
842     return( left.altimetry_map == right.altimetry_map );
843   }
844
845
846   inline bool operator!=( const Altimetry& left, const Altimetry& right ) {
847     if ( &left == &right ) return false;
848     return( left.altimetry_map != right.altimetry_map );
849   }
850
851 }
852
853
854 #endif /* ALTIMETRY_DEFINITIONS_H */
855
856
857
```

14.109 woss/woss_def/coordinates-definitions.cpp File Reference

Provides the implementation of the woss::Coord and woss::CoordZ classes.

14.109.1 Detailed Description

Provides the implementation of the woss::Coord and woss::CoordZ classes.

Author

Federico Guerra

Provides the implementation of the woss::Coord and woss::CoordZ classes

14.110 woss/woss_def/coordinates-definitions.h File Reference

Provides the interface for the woss::Coord and woss::CoordZ classes.

Classes

- class woss::Coord

  Coordinates (lat, long) class definitions and functions library.
- class woss::CoordZ

  3D-Coordinates (lat, long, depth) class definitions and functions library
- class woss::CoordZ::CartCoords

  Class that represents cartesian coordinates.
- class woss::UtmWgs84
- class woss::CoordComparator< CompUser, T >

  Function object for partial ordering of coordinates.
- class woss::CoordComparator< CompUser, CoordZ >

  Partial specialization for partial ordering of CoordZ coordinates.

Typedefs

- typedef char woss::UtmZoneChar
- typedef ::std::vector< CoordZ > woss::CoordZVector
- typedef int woss::Marsden
- typedef ::std::pair< int, int > woss::MarsdenCoord
- typedef ::std::vector< Coord > woss::CoordVector
- typedef ::std::vector< Marsden > woss::MarsdenVector
- typedef ::std::vector< MarsdenCoord > woss::MarsdenCoordVector

Functions

- Coord & woss::operator+= (Coord &left, const Coord &right)
- Coord & woss::operator-= (Coord &left, const Coord &right)
- CoordZ & woss::operator+= (CoordZ &left, const CoordZ &right)
- CoordZ & woss::operator-= (CoordZ &left, const CoordZ &right)
- const Coord woss::operator+ (const Coord &left, const Coord &right)
- const Coord woss::operator- (const Coord &left, const Coord &right)
- bool woss::operator== (const Coord &left, const Coord &right)
- bool woss::operator!= (const Coord &left, const Coord &right)
- bool woss::operator> (const Coord &left, const Coord &right)
- bool woss::operator< (const Coord &left, const Coord &right)
- bool woss::operator>= (const Coord &left, const Coord &right)
- bool woss::operator<= (const Coord &left, const Coord &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const Coord &x)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const CoordZ::CartCoords &instance)

- bool woss::operator== (const CoordZ &left, const CoordZ &right)
- bool woss::operator!= (const CoordZ &left, const CoordZ &right)
- bool woss::operator> (const CoordZ &left, const CoordZ &right)
- bool woss::operator< (const CoordZ &left, const CoordZ &right)
- bool woss::operator>= (const CoordZ &left, const CoordZ &right)
- bool woss::operator<= (const CoordZ &left, const CoordZ &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const CoordZ &instance)
- const CoordZ woss::operator+ (const CoordZ &left, const CoordZ &right)
- const CoordZ woss::operator- (const CoordZ &left, const CoordZ &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const UtmWgs84 &instance)

14.110.1 Detailed Description

Provides the interface for the woss::Coord and woss::CoordZ classes.

Author

Federico Guerra

Provides the interface for the woss::Coord and woss::CoordZ classes

14.110.2 Typedef Documentation

14.110.2.1 CoordVector `typedef ::std::vector< Coord > woss::CoordVector`

A vector of Coord

14.110.2.2 CoordZVector `typedef ::std::vector< CoordZ > woss::CoordZVector`

A vector of CoordZ. Beware of object slicing if using this container for objects *derived* from CoordZ

14.110.2.3 Marsden `typedef int woss::Marsden`

The marsden square

14.110.2.4 MarsdenCoord `typedef ::std::pair< int, int > woss::MarsdenCoord`

The Marsden coordinates

Parameters

int	marsden square
int	marsden one degree square

14.110.2.5 MarsdenCoordVector `typedef ::std::vector< MarsdenCoord > woss::MarsdenCoordVector`

A vector of marsden coordinates

14.110.2.6 MarsdenVector `typedef ::std::vector< Marsden > woss::MarsdenVector`

A vector of marsden squares

14.110.2.7 UtmZoneChar `typedef char woss::UtmZoneChar`

The letter that identify a UTM zone

14.110.3 Function Documentation

14.110.3.1 operator"!=() [1/2]

```
bool woss::operator!= (
    const Coord & left,
    const Coord & right )  [inline]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.110.3.2 operator"!=() [2/2]

```
bool woss::operator!= (
    const CoordZ & left,
    const CoordZ & right )  [inline]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.110.3.3 operator+() [1/2]

```
const Coord woss::operator+ (
    const Coord & left,
    const Coord & right )  [inline]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.110.3.4 operator+() [2/2]

```
const CoordZ woss::operator+ (
    const CoordZ & left,
    const CoordZ & right )  [inline]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.110.3.5 operator+=() [1/2]

```
Coord & woss::operator+= (
    Coord & left,
    const Coord & right )
```

Compound assignment sum operator

Parameters

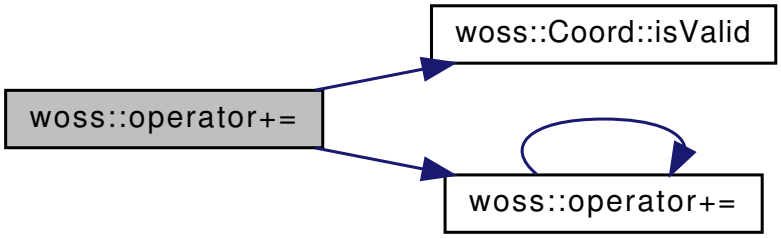
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Coord::isValid(), woss::Coord::latitude, woss::Coord::longitude, and woss::operator+=().

Here is the call graph for this function:



14.110.3.6 operator+=() [2/2] CoordZ & woss::operator+= (CoordZ & *left*, const CoordZ & *right*)

Compound assignment sum operator

Parameters

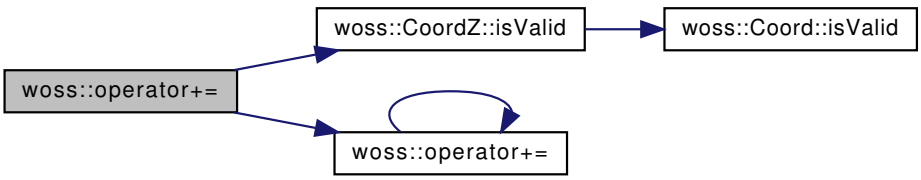
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::CoordZ::depth, woss::CoordZ::isValid(), woss::Coord::latitude, woss::Coord::longitude, and woss::operator+=().

Here is the call graph for this function:



14.110.3.7 operator-() [1/2]

```
const Coord woss::operator- (
    const Coord & left,
    const Coord & right )  [inline]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.110.3.8 operator-() [2/2]

```
const CoordZ woss::operator- (
    const CoordZ & left,
    const CoordZ & right )  [inline]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.110.3.9 operator-=() [1/2]

```
Coord & woss::operator-= (
    Coord & left,
    const Coord & right )
```

Compound assignment subtraction operator

Parameters

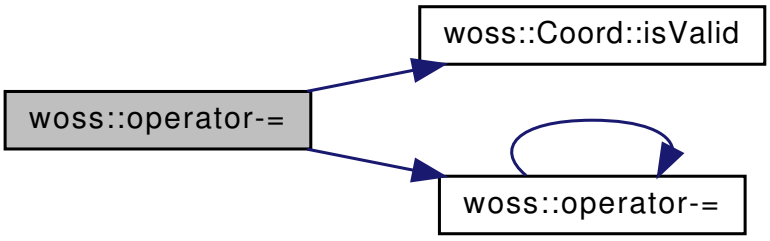
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Coord::isValid(), woss::Coord::latitude, woss::Coord::longitude, and woss::operator-=().

Here is the call graph for this function:



14.110.3.10 operator-=() [2/2]

```
CoordZ & woss::operator-= (
    CoordZ & left,
    const CoordZ & right )
```

Compound assignment subtraction operator

Parameters

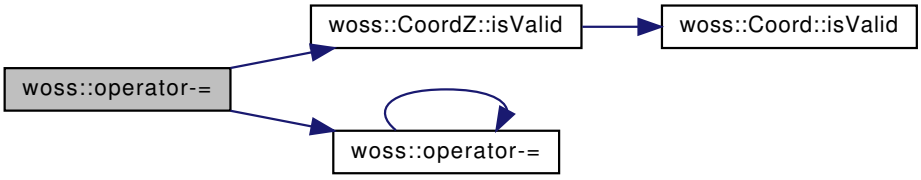
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::CoordZ::depth, woss::CoordZ::isValid(), woss::Coord::latitude, woss::Coord::longitude, and woss::operator-=().

Here is the call graph for this function:



14.110.3.11 operator<() [1/2]

```
bool woss::operator< (
    const Coord & left,
    const Coord & right )  [inline]
```

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

Referenced by woss::CoordComparator< CompUser, CoordZ >::operator()(), and woss::CoordComparator< CompUser, T >::opera

14.110.3.12 operator<() [2/2] bool woss::operator< (

```
const CoordZ & left,
const CoordZ & right )  [inline]
```

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

14.110.3.13 operator<=() [1/2] bool woss::operator<= (

```
const Coord & left,
const Coord & right )  [inline]
```

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

14.110.3.14 operator<=() [2/2] bool woss::operator<= (const CoordZ & *left*, const CoordZ & *right*) [inline]

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

14.110.3.15 operator==() [1/2] bool woss::operator== (const Coord & *left*, const Coord & *right*) [inline]

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

14.110.3.16 operator==() [2/2] bool woss::operator== (const CoordZ & *left*, const CoordZ & *right*) [inline]

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

14.110.3.17 operator>() [1/2] bool woss::operator> (const Coord & *left*, const Coord & *right*) [inline]

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

14.110.3.18 operator>() [2/2] bool woss::operator> (const CoordZ & *left*, const CoordZ & *right*) [inline]

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

14.110.3.19 operator>=() [1/2] bool woss::operator>= (const Coord & *left*, const Coord & *right*) [inline]

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

14.110.3.20 operator>=() [2/2] bool woss::operator>= (const CoordZ & *left*, const CoordZ & *right*) [inline]

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

14.111 coordinates-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_COORDINATES_DEFINITIONS_H
41 #define WOSS_COORDINATES_DEFINITIONS_H
42
43
44 #include <vector>
45 #include <utility>
46 #include <iostream>
47
48
49 namespace woss {
50
51
52   class Coord;
53   class CoordZ;
54
58   typedef char UtmZoneChar;
59
64   typedef ::std::vector< CoordZ > CoordZVector;
65
69   typedef int Marsden;
70
76   typedef ::std::pair< int, int > MarsdenCoord;
77
81   typedef ::std::vector< Coord > CoordVector;
82
86   typedef ::std::vector< Marsden > MarsdenVector;
87
```

```
91   typedef ::std::vector< MarsdenCoord > MarsdenCoordVector;
92
93
97 #define COORD_NOT_SET_VALUE (-2000)
98
99
107   class Coord {
108
109     public:
110
111     static const double COORD_MIN_LATITUDE;
112
113     static const double COORD_MAX_LATITUDE;
114
115     static const double COORD_MIN_LONGITUDE;
116
117     static const double COORD_MAX_LONGITUDE;
118
119     static const double EARTH_RADIUS;
120
122     static const double EARTH_SEMIMAJOR_AXIS;
123
125     static const double EARTH_GRS80_POLAR_RADIUS;
126
128     static const double EARTH_WGS84_POLAR_RADIUS;
129
131     static const double EARTH_GRS80_ECCENTRICITY;
132
134     static const double EARTH_WGS84_ECCENTRICITY;
135
141     Coord( double lat = COORD_NOT_SET_VALUE, double lon = COORD_NOT_SET_VALUE );
142
147     Coord( const Coord& copy );
148
149     virtual ~Coord() { }
150
155     void setLatitude( double lat ) { latitude = lat; updateMarsdenCoord(); }
156
161     void setLongitude( double lon ) { longitude = lon; updateMarsdenCoord(); }
162
163
168     virtual bool isValid()const { return( latitude >= COORD_MIN_LATITUDE && latitude <=
    COORD_MAX_LATITUDE
169                                           && longitude >= COORD_MIN_LONGITUDE && longitude <=
    COORD_MAX_LONGITUDE ); }
170
175     double getLatitude()const { return latitude; }
176
181     double getLongitude()const { return longitude; }
182
187     int getMarsdenSquare()const { return marsden_square; }
188
193     int getMarsdenOneDegreeSquare()const { return marsden_one_degree; }
194
199     MarsdenCoord getMarsdenCoord()const { return( ::std::make_pair( marsden_square, marsden_one_degree)
    ); }
200
201
207     double getInitialBearing( const Coord& destination ) const ;
208
214     double getFinalBearing( const Coord& destination ) const ;
215
222     double getGreatCircleDistance( const Coord& destination, double depth = 0 ) const ;
223
224
234     static const Coord getCoordFromBearing( const Coord& start_coord, double bearing, double distance,
    double depth = 0.0 );
235
245     static const Coord getCoordAlongGreatCircle( const Coord& start_coord, const Coord& end_coord,
    double distance, double depth = 0.0 );
246
255     static const Coord getCoordFromUtmWgs84( double easting, double northing, double utm_zone_number,
    UtmZoneChar utm_zone_char );
256
257
263     Coord& operator=( const Coord& copy );
264
265
272     friend const Coord operator+( const Coord& left , const Coord& right ) ;
273
280     friend const Coord operator-( const Coord& left , const Coord& right ) ;
281
282
289     friend Coord& operator+=( Coord& left, const Coord& right );
290
297     friend Coord& operator-=( Coord& left, const Coord& right );
298
```

```
299
306     friend bool operator==( const Coord& left , const Coord& right ) ;
307
314     friend bool operator!=( const Coord& left , const Coord& right ) ;
315
316
323     friend bool operator>( const Coord& left , const Coord& right ) ;
324
331     friend bool operator<( const Coord& left , const Coord& right ) ;
332
339     friend bool operator>=( const Coord& left , const Coord& right ) ;
340
347     friend bool operator<=( const Coord& left , const Coord& right ) ;
348
349
356     friend ::std::ostream& operator«( ::std::ostream& os, const Coord& instance );
357
358
359     protected:
360
364     double latitude;
365
369     double longitude;
370
371
375     int marsden_square;
376
380     int marsden_one_degree;
381
382
386     void updateMarsdenCoord();
387
393     static bool isValidUtmZoneChar( UtmZoneChar utm_zone_char );
394
395   };
396
397
403   class CoordZ : public Coord {
404
405
406     public:
407
408     static const double COORDZ_MIN_DEPTH;
409
417     enum CoordZSpheroidType
418     {
419       COORDZ_SPHERE = 0,
420       COORDZ_GRS80,
421       COORDZ_WGS84
422     };
423
429     class CartCoords {
430       public:
431
435       CartCoords();
436
444       CartCoords(double in_x, double in_y, double in_z, CoordZSpheroidType in_type);
445
450       double getX()const { return x; }
451
456       double getY()const { return y; }
457
462       double getZ()const { return z; }
463
468       CoordZSpheroidType getType()const { return type; }
469
476       friend ::std::ostream& operator«( ::std::ostream& os, const CartCoords& instance );
477
478       protected:
479
480       double x;
481
482       double y;
483
484       double z;
485
486       CoordZSpheroidType type;
487
488     };
489
496     CoordZ( double lat = COORD_NOT_SET_VALUE, double lon = COORD_NOT_SET_VALUE, double z =
    COORD_NOT_SET_VALUE );
497
503     explicit CoordZ( const Coord& coords, double depth = COORD_NOT_SET_VALUE );
504
509     CoordZ( const CoordZ& copy );
510
```

```
511     virtual ~CoordZ() { }
512
513
518     void setDepth( double d ) { depth = d; }
519
520
525     double getDepth()const { return depth; }
526
532     CartCoords getCartCoords(CoordZSpheroidType type = COORDZ_SPHERE) const;
533
539     double getCartX(CoordZSpheroidType type = COORDZ_SPHERE) const;
540
546     double getCartY(CoordZSpheroidType type = COORDZ_SPHERE) const;
547
553     double getCartZ(CoordZSpheroidType type = COORDZ_SPHERE) const;
554
559     double getSphericalRho() const;
560
565     double getSphericalTheta() const;
566
571     double getSphericalPhi() const;
572
573
579     double getCartDistance( const CoordZ& coords, CoordZSpheroidType type = COORDZ_SPHERE ) const;
580
585     double getCartRelZenith( const CoordZ& coords ) const;
586
591     double getCartRelAzimuth( const CoordZ& coords ) const;
592
593
602     static const CoordZ getCoordZAlongCartLine( const CoordZ& start, const CoordZ& end, double distance
    );
603
613     static const CoordZ getCoordZAlongGreatCircle( const CoordZ& start, const CoordZ& end, double
    distance );
614
623     static const CoordZ getCoordZFromCartesianCoords( double x, double y, double z, CoordZSpheroidType
    type = COORDZ_SPHERE);
624
630     static const CoordZ getCoordZFromCartesianCoords( const CartCoords& cart_coords );
631
639     static const CoordZ getCoordZFromSphericalCoords( double rho, double theta, double phi);
640
645     virtual bool isValid()const { return( Coord::isValid() && depth >= COORDZ_MIN_DEPTH ); }
646
647
653     CoordZ& operator=( const CoordZ& coordz );
654
655
662     friend const CoordZ operator+( const CoordZ& left , const CoordZ& right );
663
670     friend const CoordZ operator-( const CoordZ& left , const CoordZ& right );
671
672
679     friend CoordZ& operator+=( CoordZ& left, const CoordZ& right );
680
687     friend CoordZ& operator-=( CoordZ& left, const CoordZ& right );
688
689
696     friend bool operator==( const CoordZ& left , const CoordZ& right ) ;
697
704     friend bool operator!=( const CoordZ& left , const CoordZ& right ) ;
705
706
713     friend bool operator>( const CoordZ& left , const CoordZ& right ) ;
714
721     friend bool operator<( const CoordZ& left , const CoordZ& right ) ;
722
729     friend bool operator>=( const CoordZ& left , const CoordZ& right ) ;
730
737     friend bool operator<=( const CoordZ& left , const CoordZ& right ) ;
738
739
746     friend ::std::ostream& operator«( ::std::ostream& os, const CoordZ& instance );
747
748
749     protected:
750
751
755     double depth;
756   };
757
758    class UtmWgs84 {
759
760     public:
761
768     UtmWgs84(int utmZone = COORD_NOT_SET_VALUE, double east = COORD_NOT_SET_VALUE, double north =
```

```
    COORD_NOT_SET_VALUE);
769
770     ~UtmWgs84() { }
771
772
773     int getZone() const;
774
775     double getEasting() const;
776
777     double getNorthing() const;
778
779     bool isValid() const;
780
788     static const UtmWgs84 getUtmWgs84FromCoord(const Coord& coords);
789
796     friend ::std::ostream& operator<( ::std::ostream& os, const UtmWgs84& instance );
797
798     protected:
799
800     int zone;
801     double easting;
802     double northing;
803   };
804
805
806
814   template < class CompUser, class T = Coord >
815   class CoordComparator {
816
817
818     public:
819
820
829     bool operator()( const T& x, const T& y )const {
830 //       ::std::cout << "CoordComparator::operator() x " <<  x << "; y " << y << "; dist " <<
    x.getCartDistance(y)
831 //                   << "; space samp " << CompUser::getSpaceSampling() << ::std::endl;
832
833       if ( CompUser::getSpaceSampling() <= 0.0 ) return operator<(x,y);
834       if ( x.getGreatCircleDistance(y) <= CompUser::getSpaceSampling() ) return false;
835       return operator<(x,y);
836     }
837
838   };
839
840
848   template < class CompUser >
849   class CoordComparator< CompUser, CoordZ > {
850
851     public:
852
853
862     bool operator()( const CoordZ& x, const CoordZ& y )const {
863 //       ::std::cout << "CoordComparator::operator() x " <<  x << "; y " << y << "; dist " <<
    x.getCartDistance(y)
864 //                   << "; space samp " << CompUser::getSpaceSampling() << ::std::endl;
865
866       if ( CompUser::getSpaceSampling() <= 0.0 ) return operator<(x,y);
867       if ( x.getCartDistance(y) <= CompUser::getSpaceSampling() ) return false;
868       return operator<(x,y);
869     }
870
871   };
872
873   //non-inline operator declarations
875   Coord& operator+=( Coord& left, const Coord& right );
876
877   Coord& operator-=( Coord& left, const Coord& right );
878
879   CoordZ& operator+=( CoordZ& left, const CoordZ& right );
880
881   CoordZ& operator-=( CoordZ& left, const CoordZ& right );
882
883
885   // inline functions
886   inline const Coord operator+( const Coord& left , const Coord& right )  {
887     if( !( left.isValid() && right.isValid() ) ) return Coord();
888     return( Coord( (left.latitude + right.latitude), (left.longitude + right.longitude) ) );
889   }
890
891
892   inline const Coord operator-( const Coord& left , const Coord& right )  {
893     if( !( left.isValid() && right.isValid() ) ) return Coord();
894     return( Coord( (left.latitude - right.latitude), (left.longitude - right.longitude) ) );
895   }
896
897
```

```
898   inline bool operator==( const Coord& left , const Coord& right )  {
899     if ( &left == &right ) return true;
900     return( left.latitude == right.latitude && left.longitude == right.longitude );
901   }
902
903
904   inline bool operator!=( const Coord& left , const Coord& right )  {
905     if ( &left == &right ) return false;
906     return( left.latitude != right.latitude || left.longitude != right.longitude );
907   }
908
909
910   inline bool operator>( const Coord& left , const Coord& right )  {
911     if ( &left == &right ) return false;
912     return( ( left.latitude > right.latitude ) || ( ( left.latitude == right.latitude ) && (
      left.longitude > right.longitude ) ) );
913   }
914
915
916   inline bool operator<( const Coord& left , const Coord& right )  {
917     if ( &left == &right ) return false;
918     return( ( left.latitude < right.latitude ) || ( ( left.latitude == right.latitude ) && (
      left.longitude < right.longitude ) ) );
919   }
920
921
922   inline bool operator>=( const Coord& left , const Coord& right )  {
923     if ( left == right ) return true;
924     return( left > right );
925   }
926
927
928   inline bool operator<=( const Coord& left , const Coord& right )  {
929     if ( left == right ) return true;
930     return( left < right );
931   }
932
933
934   inline ::std::ostream& operator«(::std::ostream& os, const Coord& x) {
935     os « "Latitude = " « x.latitude « "; Longitude = " « x.longitude
936        « "; Marsden square = " « x.marsden_square
937        « "; Marsden One Degree square = " « x.marsden_one_degree;
938     return os;
939   }
940
941
942   inline ::std::ostream& operator«( ::std::ostream& os, const CoordZ::CartCoords& instance ) {
943     os « "X = " « instance.x « "; Y = " « instance.y « "; Z = " « instance.z « "; type = " «
      instance.type;
944     return os;
945   }
946
947
948   inline bool operator==( const CoordZ& left, const CoordZ& right ) {
949     if ( &left == &right ) return true;
950     return( left.latitude == right.latitude && left.longitude == right.longitude && left.depth ==
      right.depth );
951   }
952
953
954   inline bool operator!=( const CoordZ& left, const CoordZ& right ) {
955     if ( &left == &right ) return false;
956     return( left.latitude != right.latitude || left.longitude != right.longitude || left.depth !=
      right.depth );
957   }
958
959
960   inline bool operator>( const CoordZ& left, const CoordZ& right ) {
961     if ( &left == &right ) return false;
962
963     return( ( left.latitude > right.latitude ) || ( ( left.latitude == right.latitude ) && (
      left.longitude > right.longitude ) )
964         || ( ( left.latitude == right.latitude ) && ( left.longitude == right.longitude ) && ( left.depth
      > right.depth ) ) );
965   }
966
967
968   inline bool operator<( const CoordZ& left, const CoordZ& right ) {
969     if ( &left == &right ) return false;
970     return ( ( left.latitude < right.latitude ) || ( ( left.latitude == right.latitude ) && (
      left.longitude < right.longitude ) )
971         || ( ( left.latitude == right.latitude ) && ( left.longitude == right.longitude ) && ( left.depth
      < right.depth ) ) );
972   }
973
974
975   inline bool operator>=( const CoordZ& left, const CoordZ& right ) {
```

```
976     if ( left == right ) return true;
977     return( left > right );
978   }
979
980
981   inline bool operator<=( const CoordZ& left, const CoordZ& right ) {
982     if ( left == right ) return true;
983     return( left < right );
984   }
985
986
987   inline ::std::ostream& operator«( ::std::ostream& os, const CoordZ& instance ) {
988     os « "Latitude = " « instance.latitude « "; Longitude = " « instance.longitude « "; Depth = " «
      instance.depth
989        « "; Marsden square = " « instance.marsden_square
990        « "; Marsden One Degree square = " « instance.marsden_one_degree;
991     return os;
992   }
993
994
995   inline const CoordZ operator+( const CoordZ& left , const CoordZ& right ) {
996     if( !( left.isValid() && right.isValid() ) ) return CoordZ();
997     return( CoordZ( left.latitude + right.latitude, left.longitude + right.longitude, left.depth +
      right.depth ) );
998   }
999
1000
1001    inline const CoordZ operator-( const CoordZ& left , const CoordZ& right ) {
1002      if( !( left.isValid() && right.isValid() ) ) return CoordZ();
1003      return( CoordZ( left.latitude - right.latitude, left.longitude - right.longitude, left.depth -
      right.depth ) );
1004    }
1005
1006    inline int UtmWgs84::getZone()const {
1007      return zone;
1008    }
1009
1010    inline double UtmWgs84::getEasting()const {
1011      return easting;
1012    }
1013
1014    inline double UtmWgs84::getNorthing()const {
1015      return northing;
1016    }
1017
1018    inline bool UtmWgs84::isValid()const {
1019      return ((zone != COORD_NOT_SET_VALUE) && (easting != COORD_NOT_SET_VALUE) && (northing !=
      COORD_NOT_SET_VALUE));
1020    }
1021
1022    inline ::std::ostream& operator«( ::std::ostream& os, const UtmWgs84& instance ) {
1023      os « "Zone = " « instance.zone « "; Easting = " « instance.easting « "; Northing = " «
      instance.northing;
1024      return os;
1025    }
1026 }
1027
1028 #endif /* WOSS_COORDINATES_DEFINITIONS_H */
1029
```

14.112 woss/woss_def/custom-precision-double.cpp File Reference

Provides the implementation for the woss::PDouble class and friend functions.

14.112.1 Detailed Description

Provides the implementation for the woss::PDouble class and friend functions.

Author

Federico Guerra

Provides the implementation for the PDouble class and friend functions

14.113 woss/woss_def/custom-precision-double.h File Reference

Provides the interface for the woss::PDouble class.

Classes

- class woss::PDouble

  Custom precision long double class.

Functions

- const PDouble woss::operator+ (const PDouble &left, const PDouble &right)
- const PDouble woss::operator- (const PDouble &left, const PDouble &right)
- const PDouble woss::operator/ (const PDouble &left, const PDouble &right)
- const PDouble woss::operator∗ (const PDouble &left, const PDouble &right)
- const PDouble woss::operator% (const PDouble &left, const PDouble &right)
- PDouble & woss::operator+= (PDouble &left, const PDouble &right)
- PDouble & woss::operator-= (PDouble &left, const PDouble &right)
- PDouble & woss::operator/= (PDouble &left, const PDouble &right)
- PDouble & woss::operator∗= (PDouble &left, const PDouble &right)
- PDouble & woss::operator%= (PDouble &left, const PDouble &right)
- bool woss::operator== (const PDouble &left, const PDouble &right)
- bool woss::operator!= (const PDouble &left, const PDouble &right)
- bool woss::operator> (const PDouble &left, const PDouble &right)
- bool woss::operator< (const PDouble &left, const PDouble &right)
- bool woss::operator>= (const PDouble &left, const PDouble &right)
- bool woss::operator<= (const PDouble &left, const PDouble &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const PDouble &instance)
- inline ::std::istream & **woss::operator**>> (::std::istream &is, PDouble &instance)

14.113.1 Detailed Description

Provides the interface for the woss::PDouble class.

Author

Federico Guerra

Provides the interface for the PDouble class. PDouble stands for PrecisionDouble: the value and the precision provided are used for arithmetic computations and comparisons. This class is not intented for inheritance: all containers used in WOSS are based on PDouble *objects*, not pointers to objects. Therefore, inheritance requires modification in other classes, to prevent object slicing when inserting in containers PDouble-derived objects instead of PDouble objects. Moreover, due to intensive use of PDouble objects, using heap-based pointers adds another layer of indirection, therefore adding an efficiency penalty.

14.113.2 Function Documentation

14.113.2.1 operator"!=() `bool woss::operator!= ( const PDouble & left, const PDouble & right ) [inline]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.113.2.2 operator%() `const PDouble woss::operator% ( const PDouble & left, const PDouble & right ) [inline]`

Modulo operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.113.2.3 operator%=() `PDouble & woss::operator%= ( PDouble & left, const PDouble & right ) [inline]`

Compound assignment modulo operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.113.2.4 operator*() const PDouble woss::operator* (const PDouble & *left*, const PDouble & *right*) [inline]

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.113.2.5 operator*=() PDouble & woss::operator*= (PDouble & *left*, const PDouble & *right*) [inline]

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.113.2.6 operator+() const PDouble woss::operator+ (const PDouble & *left*, const PDouble & *right*) [inline]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.113.2.7 operator+=() PDouble & woss::operator+= (PDouble & *left*, const PDouble & *right*) [inline]

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.113.2.8 operator-() const PDouble woss::operator- (const PDouble & *left*, const PDouble & *right*) [inline]

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.113.2.9 operator-=() PDouble & woss::operator-= (PDouble & *left*, const PDouble & *right*) [inline]

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.113.2.10 operator/() `const PDouble woss::operator/ (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.113.2.11 operator/=() `PDouble & woss::operator/= (`

```
PDouble & left,
const PDouble & right )  [inline]
```

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.113.2.12 operator<() `bool woss::operator< (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

14.113.2.13 operator<=() `bool woss::operator<= (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

14.113.2.14 operator==() `bool woss::operator== (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.113.2.15 operator>() `bool woss::operator> (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

14.113.2.16 operator>=() `bool woss::operator>= (`

```
const PDouble & left,
const PDouble & right )  [inline]
```

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

14.114 custom-precision-double.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
45 #ifndef CUSTOM_PRECISION_DOUBLE_H
46 #define CUSTOM_PRECISION_DOUBLE_H
47
48
49 #include <iostream>
50 #include <cmath>
51
52
53
54 namespace woss {
55
56
60 #define PDOUBLE_DEFAULT_PRECISION (1.0e-17)
61
67   class PDouble {
```

```
68
69
70      public:
71
77      PDouble( const long double input = 0.0, const long double precision = PDOUBLE_DEFAULT_PRECISION );
78
83      PDouble( const PDouble& copy );
84
89      ~PDouble();
90
91
96      static void setDebug( bool flag ) { debug = flag; }
97
102      void setPrecision( double value ) { precision = value; }
103
104
109      long double getPrecision()const { return precision; }
110
115      long double getValue()const { return value; }
116
117
123      PDouble& operator=( const PDouble& copy );
124
125
130      operator int() const;
131
136      operator float() const;
137
142      operator double() const;
143
148      operator long double() const;
149
150
157      friend const PDouble operator+( const PDouble& left, const PDouble& right );
158
165      friend const PDouble operator-( const PDouble& left, const PDouble& right );
166
173      friend const PDouble operator/( const PDouble& left, const PDouble& right );
174
181      friend const PDouble operator*( const PDouble& left, const PDouble& right );
182
189      friend const PDouble operator%( const PDouble& left, const PDouble& right );
190
191
198      friend PDouble& operator+=( PDouble& left, const PDouble& right );
199
206      friend PDouble& operator-=( PDouble& left, const PDouble& right );
207
214      friend PDouble& operator/=( PDouble& left, const PDouble& right );
215
222      friend PDouble& operator*=( PDouble& left, const PDouble& right );
223
230      friend PDouble& operator%=( PDouble& left, const PDouble& right );
231
232
239      friend bool operator==( const PDouble& left, const PDouble& right );
240
247      friend bool operator!=( const PDouble& left, const PDouble& right );
248
249
256      friend bool operator>( const PDouble& left, const PDouble& right );
257
264      friend bool operator<( const PDouble& left, const PDouble& right );
265
272      friend bool operator>=( const PDouble& left, const PDouble& right );
273
280      friend bool operator<=( const PDouble& left, const PDouble& right );
281
282
289      friend ::std::ostream& operator«( ::std::ostream& os, const PDouble& instance );
290
297      friend ::std::istream& operator»( ::std::istream& is, PDouble& instance );
298
299
300
301      protected:
302
306      long double value;
307
311      long double precision;
312
313
317      static bool debug;
318
319    };
320
321
```

```
323   // inline functions
324
325   inline PDouble::operator double()const {
326     return( (double) value) ;
327   }
328
329
330   inline PDouble::operator int()const {
331     return( (int) value );
332   }
333
334
335   inline PDouble::operator float()const {
336     return( (float) value );
337   }
338
339
340   inline PDouble::operator long double()const {
341     return( (long double) value );
342   }
343
344
345   inline const PDouble operator+( const PDouble& left, const PDouble& right ) {
346     return( PDouble(left.value + right.value, ::std::max(left.precision, right.precision)) );
347   }
348
349
350   inline const PDouble operator-( const PDouble& left, const PDouble& right ) {
351     return( PDouble(left.value - right.value, ::std::max(left.precision, right.precision)) );
352   }
353
354
355   inline const PDouble operator/( const PDouble& left, const PDouble& right ) {
356     return( PDouble(left.value / right.value, ::std::max(left.precision, right.precision)) );
357   }
358
359
360   inline const PDouble operator*( const PDouble& left, const PDouble& right ) {
361     return( PDouble(left.value * right.value, ::std::max(left.precision, right.precision)) );
362   }
363
364
365   inline const PDouble operator%( const PDouble& left, const PDouble& right ) {
366     return( PDouble( ::std::fmod(left.value, right.value), ::std::max(left.precision, right.precision) )
      );
367   }
368
369
370   inline PDouble& operator+=( PDouble& left, const PDouble& right ) {
371   //   if ( &left == &right ) { /* self assignment code here */ }
372     left.value += right.value;
373     return left;
374   }
375
376
377   inline PDouble& operator-=( PDouble& left, const PDouble& right ) {
378   //   if ( &left == &right ) { /* self assignment code here */ }
379     left.value -= right.value;
380     return left;
381   }
382
383
384   inline PDouble& operator/=( PDouble& left, const PDouble& right ) {
385   //   if ( &left == &right ) { /* self assignment code here */ }
386     left.value /= right.value;
387     return left;
388   }
389
390
391   inline PDouble& operator*=( PDouble& left, const PDouble& right ) {
392   //   if ( &left == &right ) { /* self assignment code here */ }
393     left.value *= right.value;
394     return left;
395   }
396
397
398   inline PDouble& operator%=( PDouble& left, const PDouble& right ) {
399   //   if ( &left == &right ) { /* self assignment code here */ }
400     left.value = ::std::fmod(left.value, right.value);
401     return left;
402   }
403
404
405   inline bool operator==( const PDouble& left, const PDouble& right ) {
406     if ( &left == &right ) { return true; }
407     return( ::std::abs(left.value - right.value) <= left.precision );
408   }
```

```
409
410
411   inline bool operator!=( const PDouble& left, const PDouble& right ) {
412     if ( &left == &right ) { return false; }
413     return( ::std::abs(left.value - right.value) > left.precision );
414   }
415
416
417   inline bool operator>( const PDouble& left, const PDouble& right ) {
418     if ( left == right ) return false;
419     return( left.value > right.value );
420   }
421
422
423   inline bool operator<( const PDouble& left, const PDouble& right ) {
424     if ( left == right ) return false;
425     return( left.value < right.value );
426   }
427
428
429   inline bool operator>=( const PDouble& left, const PDouble& right ) {
430     if ( left == right ) return true;
431     return( left.value > right.value );
432   }
433
434
435   inline bool operator<=( const PDouble& left, const PDouble& right ) {
436     if ( left == right ) return true;
437     return( left.value < right.value );
438   }
439
440
441   inline ::std::ostream& operator«( ::std::ostream& os, const PDouble& instance ) {
442     os « instance.value;
443     return os;
444   }
445
446
447   inline ::std::istream& operator»( ::std::istream& is, PDouble& instance ) {
448     is » instance.value;
449     return is;
450   }
451
452 }
453
454 #endif /* CUSTOM_PRECISION_DOUBLE_H */
455
```

14.115 woss/woss_def/definitions-handler.cpp File Reference

Provides the implementation of woss::DefHandler class.

14.115.1 Detailed Description

Provides the implementation of woss::DefHandler class.

Author

Federico Guerra

Provides the implementation of the woss::DefHandler class

14.116 woss/woss_def/definitions-handler.h File Reference

Provides the interface for woss::DefHandler class.

Classes

- class woss::DefHandler

  Class for managing dynamic instantiation of foundation classes.

Typedefs

- typedef Singleton< DefHandler > woss::SDefHandler

  Singleton implementation of DefHandler class.

14.116.1 Detailed Description

Provides the interface for woss::DefHandler class.

Author

Federico Guerra

Provides the interface for the woss::DefHandler class

14.116.2 Typedef Documentation

14.116.2.1 SDefHandler `typedef Singleton< DefHandler > woss::SDefHandler`

Singleton implementation of DefHandler class.

Singleton implementation of DefHandler class

14.117 definitions-handler.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
```

```
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DEF_HANDLER_H
41 #define WOSS_DEF_HANDLER_H
42
43
44 #include <cassert>
45 #include <singleton-definitions.h>
46 #include <time-definitions.h>
47 #include <random-generator-definitions.h>
48
49
50 namespace woss {
51
52
53   class WossDbCreator;
54   class WossCreator;
55   class WossDbManager;
56   class WossManager;
57   class SSP;
58   class Sediment;
59   class Pressure;
60   class TimeArr;
61   class Transducer;
62   class Altimetry;
63
64
75   class DefHandler {
76
77
78     public:
79
80
84     DefHandler();
85
86
90     DefHandler( const DefHandler& copy );
91
95     DefHandler& operator=( const DefHandler& copy );
96
97
101     virtual ~DefHandler();
102
103
108     double getRand() const;
109
114     int getRandInt() const;
115
116
121     double getTimeReference() const;
122
123
124     void setPressure( Pressure* const ptr ) { pressure_creator = ptr; }
125
126     void setTimeArr( TimeArr* const ptr ) { time_arr_creator = ptr; }
127
128     void setSSP( SSP* const ptr ) { ssp_creator = ptr; }
129
130     void setSediment( Sediment* const ptr ) { sediment_creator = ptr; }
131
132     void setTimeReference( TimeReference* const ptr ) { time_reference = ptr; }
133
134     void setRandGenerator( RandomGenerator* const ptr ) { rand_generator = ptr; }
135
136     void setTransducer( Transducer* const ptr ) { transducer = ptr; }
137
138     void setAltimetry( Altimetry* const ptr ) { altimetry_creator = ptr; }
139
140
141     const Pressure* const getPressure()const { assert(pressure_creator); return pressure_creator; }
142
143     const TimeArr* const getTimeArr()const { assert(time_arr_creator); return time_arr_creator; }
144
145     const SSP* const getSSP()const { assert(ssp_creator); return ssp_creator; }
146
147     const Sediment* const getSediment()const { assert(sediment_creator); return sediment_creator; }
148
149     RandomGenerator* const getRandomGenerator()const { assert(rand_generator); return rand_generator; }
150
151     const Transducer* const getTransducer()const { assert(transducer); return transducer; }
152
153     const Altimetry* const getAltimetry()const { assert(altimetry_creator); return altimetry_creator; }
154
155
156     void setDebug( bool flag ) { debug = flag; }
```

```
157
158     bool getDebug() { return debug; }
159
160
161     protected:
162
163
167     bool debug;
168
169
170     SSP* ssp_creator;
171
172     Sediment* sediment_creator;
173
174     Pressure* pressure_creator;
175
176     TimeArr* time_arr_creator;
177
178     TimeReference* time_reference;
179
180     RandomGenerator* rand_generator;
181
182     Transducer* transducer;
183
184     Altimetry* altimetry_creator;
185
186   };
187
188
194   typedef Singleton< DefHandler > SDefHandler;
195
196
197   //inline functions
198
199   inline double DefHandler::getRand()const {
200     if (rand_generator) return rand_generator->getRand();
201     else {
202       ::std::cerr << "DefHandler::getRand() ERROR, random generator wasn't set" << ::std::endl;
203       exit(1);
204     }
205   }
206
207
208   inline int DefHandler::getRandInt()const {
209     if (rand_generator) return rand_generator->getRandInt();
210     else {
211       ::std::cerr << "DefHandler::getRandInt() ERROR, random generator wasn't set" << ::std::endl;
212       exit(1);
213     }
214   }
215
216
217   inline double DefHandler::getTimeReference()const {
218     if (time_reference) return time_reference->getTimeReference();
219     else {
220       ::std::cerr << "DefHandler::getTimeReference() ERROR, time reference wasn't set" << ::std::endl;
221       exit(1);
222     }
223   }
224
225
226 }
227
228
229 #endif /* WOSS_DEF_HANDLER_H */
```

14.118 woss/woss_def/definitions.cpp File Reference

Implementations for the definitions interface.

14.118.1 Detailed Description

Implementations for the definitions interface.

Author

Federico Guerra

Implementations for the definitions interface

14.119 woss/woss_def/definitions.h File Reference

Generic functions and variables

Typedefs

- typedef double **woss::Bathymetry**

Functions

- void woss::debugWaitForUser ()

14.119.1 Detailed Description

Generic functions and variables

Author

Federico Guerra

Collects all generic functions and variables

14.119.2 Function Documentation

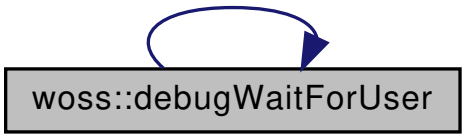
14.119.2.1 debugWaitForUser() `void woss::debugWaitForUser ( ) [inline]`

Pause the execution of the program. Restart it by pressing any key.

References woss::debugWaitForUser().

Referenced by woss::debugWaitForUser().

Here is the call graph for this function:



14.120 definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_DEFINITIONS_H
41 #define WOSS_DEFINITIONS_H
42
43
44 #include <iostream>
45
46
47 namespace woss {
48
49   typedef double Bathymetry;
50
54   static const int WOSS_DECIMAL_PRECISION = 17;
55
56   static const int WOSS_STREAM_TAB_SPACE = 25;
57
58
59
63   void debugWaitForUser();
64
65
67   inline void debugWaitForUser()
68   {
69     ::std::string response;
70     ::std::cout « "Press Enter to continue";
71     ::std::getline( ::std::cin, response);
72   }
73
74 }
75
76 #endif /* WOSS_DEFINITIONS_H */
77
```

14.121 woss/woss_def/location-definitions.h File Reference

Implementation of woss::Location class.

Classes

- class woss::Location

  Class that stores the coordinates of moving entity.

14.121.1 Detailed Description

Implementation of woss::Location class.

Definitions and library for woss::Location class.

Author

Federico Guerra

Implementation of woss::Location class

Author

Federico Guerra

Definitions and library for woss::Location class

14.122 location-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef LOCATION_DEFINITIONS_H
41 #define LOCATION_DEFINITIONS_H
42
43
44 #include <cmath>
45 #include "coordinates-definitions.h"
46 #include <iostream>
47
48
49 namespace woss {
50
51
52 #define LOCATION_COMPARISON_DISTANCE (0.0)
53
54
60   class Location {
61
62
63     public:
64
65
```

```
71     Location( const CoordZ& coordz = CoordZ(), double dist = LOCATION_COMPARISON_DISTANCE );
72
80     Location( double latitude, double longitude, double depth = 0, double dist =
       LOCATION_COMPARISON_DISTANCE );
81
85     virtual ~Location() { }
86
95     virtual Location* create( double latitude, double longitude, double depth = 0, double dist =
       LOCATION_COMPARISON_DISTANCE )const {
96        return new Location(latitude, longitude, depth, dist); }
97
104    virtual Location* create( const CoordZ& coordz = CoordZ(), double dist =
       LOCATION_COMPARISON_DISTANCE )const {
105       return new Location(coordz,dist); }
106
111    virtual Location* clone()const { return new Location(*this); }
112
113
118    virtual bool isValid()const { return curr_coordz.isValid(); }
119
120
126    virtual bool isEquivalentTo( const woss::CoordZ& coordz );
127
128
133    virtual void setLocation( const CoordZ& coordz );
134
139    virtual void setLatitude( double lat );
140
145    virtual void setLongitude( double lon );
146
151    virtual void setDepth( double depth );
152
153
159    virtual void setVerticalOrientation( double angle );
160
166    virtual void setHorizontalOrientation( double angle );
167
168
173    virtual CoordZ getLocation();
174
179    virtual double getLatitude();
180
185    virtual double getLongitude();
186
191    virtual double getDepth();
192
193
198    virtual double getX();
199
204    virtual double getY();
205
210    virtual double getZ();
211
217    virtual double getVerticalOrientation();
218
224    virtual double getHorizontalOrientation();
225
230    virtual double getBearing();
231
232
239    friend ::std::ostream& operator«( ::std::ostream& os, const Location& instance ) {
240      os « "Location, latitude = " « instance.curr_coordz.getLatitude() « "; longitude = " «
       instance.curr_coordz.getLongitude()
241         « "; depth = " « instance.curr_coordz.getDepth();
242      return os;
243    }
244
245
246    protected:
247
248
252    double comparison_distance;
253
254
258    CoordZ curr_coordz;
259
260
265    double vertical_orientation;
266
271    double horizontal_orientation;
272
276    double bearing;
277
278
279  };
280
281
```

```
282 }
283
284
285 #endif /* LOCATION_DEFINITIONS_H */
286
```

14.123 woss/woss_def/pressure-definitions.cpp File Reference

Implementation of woss::Pressure class.

14.123.1 Detailed Description

Implementation of woss::Pressure class.

Author

Federico Guerra

Implementation of woss::Pressure class

14.124 woss/woss_def/pressure-definitions.h File Reference

Definitions and library for woss::Pressure class.

Classes

- class woss::Pressure

  Complex attenuated pressure class.

Functions

- bool woss::operator== (const Pressure &left, const Pressure &right)
- bool woss::operator!= (const Pressure &left, const Pressure &right)
- const Pressure woss::operator+ (const Pressure &left, const Pressure &right)
- const Pressure woss::operator- (const Pressure &left, const Pressure &right)
- const Pressure woss::operator/ (const Pressure &left, const Pressure &right)
- const Pressure woss::operator∗ (const Pressure &left, const Pressure &right)
- Pressure & woss::operator+= (Pressure &left, const Pressure &right)
- Pressure & woss::operator-= (Pressure &left, const Pressure &right)
- Pressure & woss::operator/= (Pressure &left, const Pressure &right)
- Pressure & woss::operator∗= (Pressure &left, const Pressure &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const Pressure &instance)

14.124.1 Detailed Description

Definitions and library for woss::Pressure class.

Author

Federico Guerra

Definitions and library for complex pressure

14.124.2 Function Documentation

14.124.2.1 operator"!=() `bool woss::operator!= (`

```
const Pressure & left,
const Pressure & right )  [inline]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.124.2.2 operator∗() `const Pressure woss::operator* (`

```
const Pressure & left,
const Pressure & right )  [inline]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.124.2.3 operator∗=() `Pressure & woss::operator*= (`

```
Pressure & left,
const Pressure & right )  [inline]
```

Compound assignment multiplication operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.124.2.4 operator+() `const Pressure woss::operator+ ( const Pressure & left, const Pressure & right ) [inline]`

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.124.2.5 operator+=() `Pressure & woss::operator+= ( Pressure & left, const Pressure & right ) [inline]`

Compound assignment sum operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.124.2.6 operator-() `const Pressure woss::operator- ( const Pressure & left, const Pressure & right ) [inline]`

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.124.2.7 operator-=() `Pressure & woss::operator-= ( Pressure & left, const Pressure & right ) [inline]`

Compound assignment subtraction operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.124.2.8 operator/() `const Pressure woss::operator/ ( const Pressure & left, const Pressure & right ) [inline]`

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.124.2.9 operator/=() `Pressure & woss::operator/= ( Pressure & left, const Pressure & right ) [inline]`

Compound assignment division operator

Parameters

left	left operand reference
right	right operand const reference

Returns

left reference after the operation

14.124.2.10 operator==()

```
bool woss::operator== (
    const Pressure & left,
    const Pressure & right )  [inline]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.125 pressure-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef PRESSURE_DEFINITIONS_H
41 #define PRESSURE_DEFINITIONS_H
42
43
```

```
44 #include <complex>
45
46
47 namespace woss {
48
49
50   class TimeArr;
51
59   class Pressure {
60
61
62     public:
63
64
70     Pressure( double real = 0.0, double imag = 0.0 );
71
76     Pressure( const ::std::complex<double>& complex_press );
77
82     Pressure( const TimeArr& time_arr );
83
88     Pressure( const Pressure& copy );
89
90
97     virtual Pressure* create( double real = 0.0, double imag = 0.0 )const { return new Pressure( real,
      imag ); }
98
104     virtual Pressure* create( const ::std::complex<double>& complex_press )const { return new Pressure(
      complex_press ); }
105
111     virtual Pressure* create( const TimeArr& time_arr )const { return new Pressure( time_arr ); }
112
118     virtual Pressure* create( const Pressure& copy )const { return new Pressure( copy ); }
119
124     virtual Pressure* clone()const { return new Pressure(*this); }
125
131     virtual Pressure* createArray( unsigned int array_size )const { return new Pressure[array_size]; }
132
133
134     virtual ~Pressure() { }
135
136
141     operator ::std::complex<double>()const { return complex_pressure; }
142
143
148     static const ::std::complex<double> createNotValid() { return( ::std::complex<double>( HUGE_VAL,
      HUGE_VAL ) ); }
149
150
155     double real()const { return complex_pressure.real(); }
156
161     double imag()const { return complex_pressure.imag(); }
162
163
168     double abs()const { return ::std::abs( complex_pressure ); }
169
174     double phase()const { return ::std::arg( complex_pressure ); }
175
180     Pressure sqrt()const { return ::std::sqrt( complex_pressure ); }
181
182
187     virtual bool isValid()const { return( complex_pressure != ::std::complex<double>( HUGE_VAL, HUGE_VAL
      ) ); }
188
189
194     static double getTxLossDb( const ::std::complex<double>& val ) { if ( val == ::std::complex<double>(
      HUGE_VAL, HUGE_VAL ) ) return -HUGE_VAL;
195                                 else if( val == ::std::complex<double>( 0.0, 0.0 ) ) return HUGE_VAL;
196                                 else return( -20.0 * log10( ::std::abs(val) ) ); }
197
198
207     virtual bool checkAttenuation( double distance, double frequency );
208
209
214     static void setDebug( bool flag ) { debug = flag; }
215
216
222     Pressure& operator=( const Pressure& x );
223
224
231     friend const Pressure operator+( const Pressure& left, const Pressure& right );
232
239     friend const Pressure operator-( const Pressure& left, const Pressure& right );
240
247     friend const Pressure operator/( const Pressure& left, const Pressure& right );
248
255     friend const Pressure operator*( const Pressure& left, const Pressure& right );
256
```

```
257
264     friend Pressure& operator+=( Pressure& left, const Pressure& right );
265
272     friend Pressure& operator-=( Pressure& left, const Pressure& right );
273
280     friend Pressure& operator/=( Pressure& left, const Pressure& right );
281
288     friend Pressure& operator*=( Pressure& left, const Pressure& right );
289
290
297     friend bool operator==( const Pressure& left, const Pressure& right );
298
305     friend bool operator!=( const Pressure& left, const Pressure& right );
306
307
314     friend ::std::ostream& operator«( ::std::ostream& os, const Pressure& instance );
315
316
317     protected:
318
319
323     static bool debug;
324
325
329     ::std::complex< double > complex_pressure;
330
337     virtual double getAttenuation( double dist, double freq );
338
344     double getThorpAtt( double frequency );
345
346   };
347
348
349   //inline functions
351   inline bool operator==( const Pressure& left, const Pressure& right ) {
352     if ( &left == &right ) return true;
353     return( left.complex_pressure == right.complex_pressure );
354   }
355
356
357   inline bool operator!=( const Pressure& left, const Pressure& right ) {
358     if ( &left == &right ) return false;
359     return( left.complex_pressure != right.complex_pressure );
360   }
361
362
363   inline const Pressure operator+( const Pressure& left, const Pressure& right ) {
364     return( Pressure( left.complex_pressure + right.complex_pressure ) );
365   }
366
367
368   inline const Pressure operator-( const Pressure& left, const Pressure& right ) {
369     return( Pressure( left.complex_pressure - right.complex_pressure ) );
370   }
371
372
373   inline const Pressure operator/( const Pressure& left, const Pressure& right ) {
374     return( Pressure( left.complex_pressure / right.complex_pressure ) );
375   }
376
377
378   inline const Pressure operator*( const Pressure& left, const Pressure& right ) {
379     return( Pressure( left.complex_pressure * right.complex_pressure ) );
380   }
381
382
383   inline Pressure& operator+=( Pressure& left, const Pressure& right ) {
384     left.complex_pressure += right.complex_pressure;
385     return left;
386   }
387
388
389   inline Pressure& operator-=( Pressure& left, const Pressure& right ) {
390     left.complex_pressure -= right.complex_pressure;
391     return left;
392   }
393
394
395   inline Pressure& operator/=( Pressure& left, const Pressure& right ) {
396     left.complex_pressure /= right.complex_pressure;
397     return left;
398   }
399
400
401   inline Pressure& operator*=( Pressure& left, const Pressure& right ) {
402     left.complex_pressure *= right.complex_pressure;
403     return left;
```

```
404   }
405
406
407   inline ::std::ostream& operator«( ::std::ostream& os, const Pressure& instance ) {
408     os « instance.complex_pressure;
409     return os;
410   }
411
412
413 }
414
415
416 #endif /* PRESSURE_DEFINITIONS_H */
417
```

14.126 woss/woss_def/random-generator-definitions.cpp File Reference

Implementation of woss::RandomGenerator class.

14.126.1 Detailed Description

Implementation of woss::RandomGenerator class.

Author

Federico Guerra

Implementation of woss::RandomGenerator class

14.127 woss/woss_def/random-generator-definitions.h File Reference

Definitions and library for woss::RandomGenerator class.

Classes

- class woss::RandomGenerator
  woss::RandomGenerator class

14.127.1 Detailed Description

Definitions and library for woss::RandomGenerator class.

Author

Federico Guerra

Definitions and library for woss::RandomGenerator class

14.128 random-generator-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef RANDOM_GENERATOR_DEFINITIONS_H
41 #define RANDOM_GENERATOR_DEFINITIONS_H
42
43
44 #include <cstdlib>
45 #include <cassert>
46
47
48 namespace woss {
49
50
56   class RandomGenerator {
57
58
59     public:
60
61
66     RandomGenerator( int s = 0 ) :  seed(s), initialized(false) { }
67
71     virtual ~RandomGenerator() { }
72
73
79     virtual RandomGenerator* create( double seed ) { return new RandomGenerator(seed); }
80
85     virtual RandomGenerator* clone()const { return new RandomGenerator(*this); }
86
87
92     virtual bool isValid()const { return initialized; }
93
94
99     virtual void setSeed( int s ) { seed = s; }
100
105     virtual int getSeed()const { return seed; }
106
107
111     virtual void initialize();
112
113
118     virtual double getRand() const;
119
124     virtual int getRandInt() const;
125
126     protected:
127
128
132     int seed;
133
134
138     bool initialized;
139
140
141   };
142
143
```

```
144 }
145
146
147 #endif /* RANDOM_GENERATOR_DEFINITIONS_H */
148
```

14.129 woss/woss_def/sediment-definitions.cpp File Reference

Implementation of woss::Sediment class.

14.129.1 Detailed Description

Implementation of woss::Sediment class.

Author

Federico Guerra

Implementation of Sediment class

14.130 woss/woss_def/sediment-definitions.h File Reference

Definitions and library for woss::Sediment class.

Classes

- class woss::Sediment
  Surficial sediment geoacoustic parameters definitions.
- class woss::SedimentGravel
  Gravel type implementation.
- class woss::SedimentSand
  Sand type implementation.
- class woss::SedimentSilt
  Silt type implementation.
- class woss::SedimentClay
  Clay type implementation.
- class woss::SedimentOoze
  Ooze type implementation.
- class woss::SedimentMud
  Mud type implementation.
- class woss::SedimentRocks
  Rocks type implementation.
- class woss::SedimentOrganic
  Organic type implementation.
- class woss::SedimentNodules
  Deck41 nodules type implementation.
- class woss::SedimentHardBottom
  Hard bottom type implementation.

Typedefs

- typedef ::std::pair< int, int > **woss::Deck41Types**

Functions

- const Sediment woss::operator/ (const double left, const Sediment &right)
- const Sediment woss::operator* (const double left, const Sediment &right)
- const Sediment woss::operator/ (const Sediment &left, const double right)
- const Sediment woss::operator* (const Sediment &left, const double right)
- Sediment & woss::operator+= (Sediment &left, const Sediment &right)
- Sediment & woss::operator-= (Sediment &left, const Sediment &right)
- Sediment & woss::operator/= (Sediment &left, const Sediment &right)
- Sediment & woss::operator*= (Sediment &left, const Sediment &right)
- Sediment & woss::operator+= (Sediment &left, double right)
- Sediment & woss::operator-= (Sediment &left, double right)
- Sediment & woss::operator/= (Sediment &left, double right)
- Sediment & woss::operator*= (Sediment &left, double right)
- bool woss::operator== (const Sediment &left, const Sediment &right)
- bool woss::operator!= (const Sediment &left, const Sediment &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const Sediment &instance)
- const Sediment woss::operator+ (const Sediment &left, const Sediment &right)
- const Sediment woss::operator- (const Sediment &left, const Sediment &right)
- const Sediment woss::operator/ (const Sediment &left, const Sediment &right)
- const Sediment woss::operator* (const Sediment &left, const Sediment &right)

14.130.1 Detailed Description

Definitions and library for woss::Sediment class.

Author

Federico Guerra

Definitions and library for surficial sediment

14.130.2 Function Documentation

14.130.2.1 operator"!=() `bool woss::operator!= (`

```
const Sediment & left,
const Sediment & right )  [inline]
```

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.130.2.2 operator∗() [1/3]

```
const Sediment woss::operator* (
    const double left,
    const Sediment & right )
```

Scalar multiplication operator

Parameters

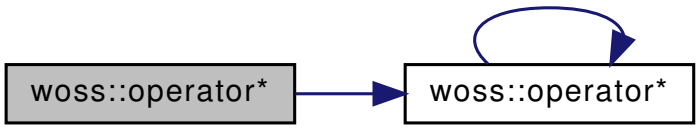
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator∗(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.3 operator∗() [2/3]

```
const Sediment woss::operator* (
    const Sediment & left,
    const double right )
```

Scalar multiplication operator

Parameters

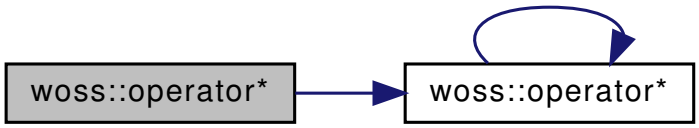
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator∗(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.4 operator∗() [3/3]

```
const Sediment woss::operator* (
    const Sediment & left,
    const Sediment & right )  [inline]
```

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.130.2.5 operator∗=() [1/2]

```
Sediment & woss::operator*= (
    Sediment & left,
    const Sediment & right )
```

Compound assignment multiplication operator

Parameters

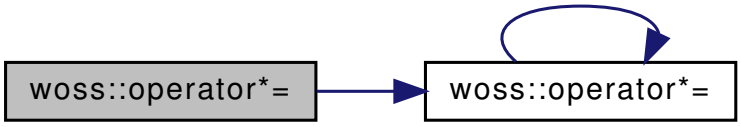
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator∗=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.6 operator*=() [2/2]

```
Sediment & woss::operator*= (
    Sediment & left,
    double right )
```

Compound assignment multiplication operator

Parameters

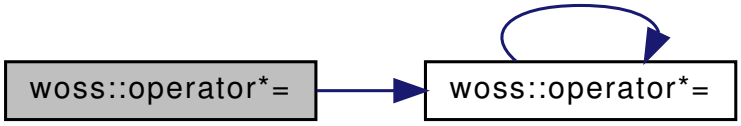
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator*=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.7 operator+()

```
const Sediment woss::operator+ (
    const Sediment & left,
    const Sediment & right )  [inline]
```

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.130.2.8 operator+=() [1/2]

```
Sediment & woss::operator+= (
    Sediment & left,
    const Sediment & right )
```

Compound assignment sum operator

Parameters

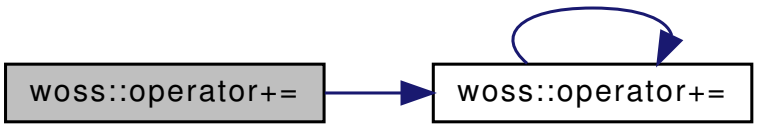
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator+=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.9 operator+=() [2/2]

```
Sediment & woss::operator+= (
    Sediment & left,
    double right )
```

Compound assignment sum operator

Parameters

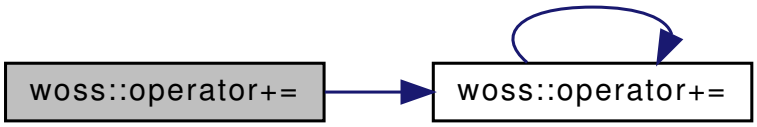
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator+=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.10 operator-() `const Sediment woss::operator- ( const Sediment & left, const Sediment & right ) [inline]`

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.130.2.11 operator-=() [1/2] `Sediment & woss::operator-= ( Sediment & left, const Sediment & right )`

Compound assignment subtraction operator

Parameters

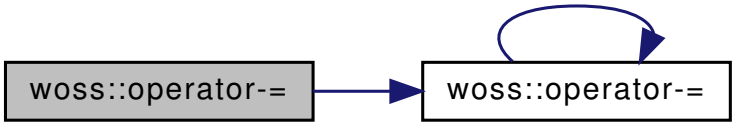
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator-=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.12 operator-=() [2/2]

```
Sediment & woss::operator-= (
    Sediment & left,
    double right )
```

Compound assignment subtraction operator

Parameters

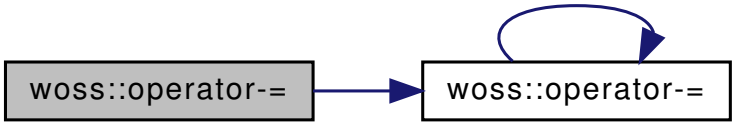
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator-=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.13 operator/() [1/3]

```
const Sediment woss::operator/ (
    const double left,
    const Sediment & right )
```

Scalar division operator

Parameters

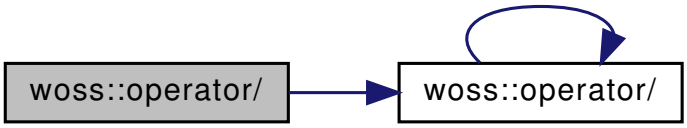
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator/(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.14 operator/() [2/3]

```
const Sediment woss::operator/ (
    const Sediment & left,
    const double right )
```

Scalar division operator

Parameters

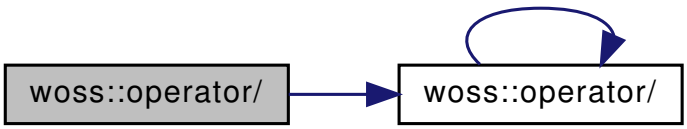
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator/(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.15 operator/() [3/3]

```
const Sediment woss::operator/ (
    const Sediment & left,
    const Sediment & right )  [inline]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.130.2.16 operator/=() [1/2]

```
Sediment & woss::operator/= (
    Sediment & left,
    const Sediment & right )
```

Compound assignment division operator

Parameters

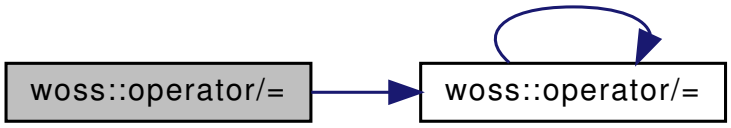
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator/=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.17 operator/=() [2/2]

```
Sediment & woss::operator/= (
    Sediment & left,
    double right )
```

Compound assignment division operator

Parameters

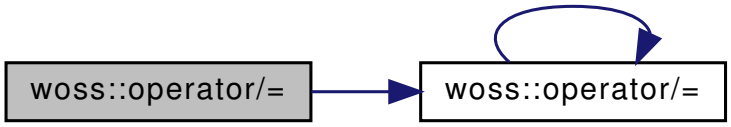
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::Sediment::att_c, woss::Sediment::att_s, woss::Sediment::density, woss::operator/=(), woss::Sediment::type, woss::Sediment::vel_c, and woss::Sediment::vel_s.

Here is the call graph for this function:



14.130.2.18 operator==()

```
bool woss::operator== (
    const Sediment & left,
    const Sediment & right )  [inline]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.131 sediment-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
```

```
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SEDIMENT_DEFINITIONS_H
41 #define WOSS_SEDIMENT_DEFINITIONS_H
42
43
44 #include <string>
45 #include "coordinates-definitions.h"
46
47
48 namespace woss {
49
50
51 #define SEDIMENT_NOT_SET_VALUE (-10000.0)
52
53
54   typedef ::std::pair < int , int > Deck41Types;
55
56
63   class Sediment {
64
65
66     public:
67
71     Sediment();
72
83     Sediment( const ::std::string& name, double velc, double vels, double dens, double attc, double atts,
     double bottom_depth = 1.0 );
84
89     Sediment( const Sediment& copy );
90
91
96     virtual Sediment* create()const { return new Sediment(); }
97
109     virtual Sediment* create( const ::std::string& name, double velc, double vels, double dens, double
     attc, double atts, double bottom_depth = 1.0 )const {
110       return new Sediment( name, velc, vels, dens, attc, atts, bottom_depth); }
111
117     virtual Sediment* create( const Sediment& copy )const { return new Sediment(copy); }
118
123     virtual Sediment* clone()const  { return new Sediment(*this); }
124
125
126     virtual ~Sediment() { };
127
128
134     Sediment& setType( const ::std::string& name ) { type = name; return *this; }
135
141     Sediment& setVelocityC( double vel  ) { vel_c = vel; return *this; }
142
148     Sediment& setVelocityS( double vel ) { vel_s = vel; return *this; }
149
155     Sediment& setDensity( double dens ) { density = dens; return *this; }
156
162     Sediment& setAttenuationC( double att ) { att_c = att; return *this; }
163
169     Sediment& setAttenuationS( double att ) { att_s = att; return *this; }
170
176     Sediment& setDepth( double bottom_depth ) { depth = bottom_depth; return *this; }
177
189     Sediment& set( const ::std::string& name, double velc, double vels, double dens, double attc ,
     double atts, double bottom_depth ) {
190                         type = name; vel_c = velc; vel_s = vels, density = dens;
191                         att_c = attc; att_s = atts; depth = bottom_depth; return *this; }
192
193
198     static void setDebug( bool flag ) { debug = flag; }
199
200
205     double getVelocityC()const { return vel_c; }
206
211     double getVelocityS()const { return vel_s; }
212
217     double getDensity()const { return density; }
218
223     double getAttenuationC()const { return att_c; }
224
229     double getAttenuationS()const { return att_s; }
230
235     double getDepth()const { return depth; }
236
241     ::std::string getType()const { return type; }
242
243
248     virtual bool isValid()const { return ( vel_c != SEDIMENT_NOT_SET_VALUE && vel_s !=
```

```
    SEDIMENT_NOT_SET_VALUE && density != SEDIMENT_NOT_SET_VALUE
249                                       && att_c != SEDIMENT_NOT_SET_VALUE && att_s !=
    SEDIMENT_NOT_SET_VALUE ); }
250
251
256     virtual const ::std::string getStringValues() const;
257
258
264     Sediment& operator=( const Sediment& time );
265
266
273     friend const Sediment operator+( const Sediment& left, const Sediment& right );
274
281     friend const Sediment operator-( const Sediment& left, const Sediment& right );
282
289     friend const Sediment operator/( const Sediment& left, const Sediment& right );
290
297     friend const Sediment operator*( const Sediment& left, const Sediment& right );
298
305     friend Sediment& operator+=( Sediment& left, const Sediment& right );
306
313     friend Sediment& operator-=( Sediment& left, const Sediment& right );
314
321     friend Sediment& operator/=( Sediment& left, const Sediment& right );
322
329     friend Sediment& operator*=( Sediment& left, const Sediment& right );
330
331
338     friend Sediment& operator+=( Sediment& left, double right );
339
346     friend Sediment& operator-=( Sediment& left, double right );
347
354     friend Sediment& operator/=( Sediment& left, double right );
355
362     friend Sediment& operator*=( Sediment& left, double right );
363
364
371     friend bool operator==( const Sediment& left, const Sediment& right );
372
379     friend bool operator!=( const Sediment& left, const Sediment& right );
380
381
388     friend const Sediment operator+( const double left, const Sediment& right );
389
396     friend const Sediment operator-( const double left, const Sediment& right );
397
404     friend const Sediment operator/( const double left, const Sediment& right );
405
412     friend const Sediment operator*( const double left, const Sediment& right );
413
414
421     friend const Sediment operator+( const Sediment& left, double right );
422
429     friend const Sediment operator-( const Sediment& left, double right );
430
437     friend const Sediment operator/( const Sediment& left, double right );
438
445     friend const Sediment operator*( const Sediment& left, double right );
446
447
454     friend ::std::ostream& operator«( ::std::ostream& os, const Sediment& instance );
455
456
457     protected:
458
462     ::std::string type;
463
464
468     double depth;
469
473     double vel_c;
474
478     double vel_s;
479
483     double density;
484
488     double att_c;
489
493     double att_s;
494
495
499     static bool debug;
500
501
502 };
503
504 // non-inline friend operator declarations
```

```
506   const Sediment operator/( const double left, const Sediment& right );
507
508   const Sediment operator*( const double left, const Sediment& right );
509
510   const Sediment operator/( const Sediment& left, const double right );
511
512   const Sediment operator*( const Sediment& left, const double right );
513
514
515   Sediment& operator+=( Sediment& left, const Sediment& right );
516
517   Sediment& operator-=( Sediment& left, const Sediment& right );
518
519   Sediment& operator/=( Sediment& left, const Sediment& right );
520
521   Sediment& operator*=( Sediment& left, const Sediment& right );
522
523
524   Sediment& operator+=( Sediment& left, double right );
525
526   Sediment& operator-=( Sediment& left, double right );
527
528   Sediment& operator/=( Sediment& left, double right );
529
530   Sediment& operator*=( Sediment& left, double right );
531
532   //inline functions
534   inline bool operator==( const Sediment& left, const Sediment& right ) {
535     if ( &left == &right ) return true;
536     return( left.vel_c == right.vel_c && left.vel_s == right.vel_s && left.att_c == right.att_c &&
      left.att_s == right.att_s
537          && left.density == right.density );
538   }
539
540
541   inline bool operator!=( const Sediment& left, const Sediment& right ) {
542     if ( &left == &right ) return false;
543     return( left.vel_c != right.vel_c || left.vel_s != right.vel_s || left.att_c != right.att_c ||
      left.att_s != right.att_s
544          || left.density != right.density );
545   }
546
547
548   inline ::std::ostream& operator«(::std::ostream& os, const Sediment& instance ) {
549     os « "Sediment type = " « instance.type « "; velocity_c = " « instance.vel_c « "; velocity_s = " «
      instance.vel_s « "; density = " « instance.density
550        « "; attenuation_c = " « instance.att_c « "; attenuation_s = " « instance.att_s;
551     return os;
552   }
553
554
555   inline const Sediment operator+( const Sediment& left, const Sediment& right ) {
556     return( Sediment( left.type + " + " + right.type, left.vel_c + right.vel_c, left.vel_s +
      right.vel_s, left.density + right.density, left.att_c + right.att_c, left.att_s + right.att_s ) );
557   }
558
559
560   inline const Sediment operator-( const Sediment& left, const Sediment& right ) {
561     return( Sediment( left.type + " - " + right.type, left.vel_c - right.vel_c, left.vel_s -
      right.vel_s, left.density - right.density, left.att_c - right.att_c, left.att_s - right.att_s ) );
562   }
563
564
565   inline const Sediment operator/( const Sediment& left, const Sediment& right ) {
566     return( Sediment( left.type + " / " + right.type, left.vel_c / right.vel_c, left.vel_s /
      right.vel_s, left.density / right.density, left.att_c / right.att_c, left.att_s / right.att_s ) );
567   }
568
569
570   inline const Sediment operator*( const Sediment& left, const Sediment& right ) {
571     return( Sediment( left.type + " * " + right.type, left.vel_c * right.vel_c, left.vel_s *
      right.vel_s, left.density * right.density, left.att_c * right.att_c, left.att_s * right.att_s ) );
572   }
574
580   class SedimentGravel :  public Sediment {
581
582     public:
583
584     SedimentGravel(double depth = 1.0);
585
586
587     protected:
588
592     double calculateVelocityS(double vels, double bottom_depth);
593
594   };
595
```

```
596
602   class SedimentSand :  public Sediment {
603
604
605     public:
606
607
608     SedimentSand();
609
610
611   };
612
613
619   class SedimentSilt :  public Sediment {
620
621
622     public:
623
624
625     SedimentSilt(double depth = 1.0);
626
627
628     protected:
629
633     double calculateVelocityS(double vels, double bottom_depth);
634
635
636   };
637
638
644   class SedimentClay :  public Sediment {
645
646
647     public:
648
649
650     SedimentClay();
651
652
653   };
654
655
661   class SedimentOoze :  public Sediment {
662
663
664     public:
665
666
667     SedimentOoze();
668
669
670   };
671
672
678   class SedimentMud :  public Sediment {
679
680
681     public:
682
683
684     SedimentMud(double depth = 1.0);
685
686
687     protected:
688
692     double calculateVelocityS(double vels, double bottom_depth);
693
694
695   };
696
697
705   class SedimentRocks :  public Sediment {
706
707
708     public:
709
710
711     SedimentRocks();
712
713
714   };
715
716
724   class SedimentOrganic :  public Sediment {
725
726
727     public:
```

```
728
729
730     SedimentOrganic();
731
732
733   };
734
735
743   class SedimentNodules :  public Sediment {
744
745
746     public:
747
748
749     SedimentNodules();
750
751
752   };
753
754
760   class SedimentHardBottom :  public Sediment {
761
762
763     public:
764
765
766     SedimentHardBottom();
767
768
769   };
770
771
772 }
773
774
775 #endif /* WOSS_SEDIMENT_DEFINITIONS_H */
776
```

14.132 woss/woss_def/singleton-definitions.h File Reference

Definitions of woss::Singleton template.

Classes

- class woss::Singleton< T >
  Singleton design pattern template.

14.132.1 Detailed Description

Definitions of woss::Singleton template.

Author

Federico Guerra

Definitions of woss::Singleton template

14.133 singleton-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef SINGLETON_DEFINITIONS_H
41 #define SINGLETON_DEFINITIONS_H
42
43
44 namespace woss {
45
46
53   template< typename T >
54   class Singleton {
55
56
57     public:
58
63     static T* instance();
64
65
66     private:
67
68
72     static T* the_instance;
73
74
78     Singleton() { }
79
83     Singleton( const Singleton& copy ) { }
84
88     Singleton& operator=( const Singleton& copy ) { return( *this ); }
89
90
91   };
92
93
94   template <typename T>
95   T* Singleton<T>::the_instance = 0;
96
97
98   template <typename T>
99   inline T* Singleton<T>::instance() {
100     if (the_instance)
101       return the_instance;
102     else {
103       the_instance = new T();
104       return the_instance;
105     }
106   }
107
108
109 }
110
111
112 #endif // SINGLETON_DEFINITIONS_H
113
```

14.134 woss/woss_def/ssp-definitions.cpp File Reference

Implementation of woss::SSP (Sound Speed Profile) class.

14.134.1 Detailed Description

Implementation of woss::SSP (Sound Speed Profile) class.

Author

Federico Guerra

Implementation of woss::SSP (Sound Speed Profile) class

14.135 woss/woss_def/ssp-definitions.h File Reference

Definitions and library for Sound Speed Profiles.

Classes

- class woss::SSP

  SSP class offers multiple creation and manipulation capabilities for sound speed profile.

Typedefs

- typedef ::std::map< PDouble, double > woss::DepthMap
- typedef DepthMap::const_iterator **woss::DConstIter**
- typedef DepthMap::iterator **woss::DIter**
- typedef DepthMap::reverse_iterator **woss::DRIter**
- typedef DepthMap::const_reverse_iterator **woss::DConstRIter**

Functions

- SSP & woss::operator+= (SSP &left, const SSP &right)
- SSP & woss::operator-= (SSP &left, const SSP &right)
- SSP & woss::operator*= (SSP &left, const SSP &right)
- SSP & woss::operator/= (SSP &left, const SSP &right)
- SSP & woss::operator+= (SSP &left, const double right)
- SSP & woss::operator-= (SSP &left, const double right)
- SSP & woss::operator/= (SSP &left, const double right)
- SSP & woss::operator*= (SSP &left, const double right)
- bool woss::operator== (const SSP &left, const SSP &right)
- bool woss::operator!= (const SSP &left, const SSP &right)
- const SSP woss::operator+ (const SSP &left, const SSP &right)
- const SSP woss::operator- (const SSP &left, const SSP &right)
- const SSP woss::operator* (const SSP &left, const SSP &right)
- const SSP woss::operator/ (const SSP &left, const SSP &right)
- const SSP woss::operator+ (const SSP &left, const double right)
- const SSP woss::operator- (const SSP &left, const double right)
- const SSP woss::operator/ (const SSP &left, const double right)
- const SSP woss::operator* (const SSP &left, const double right)
- const SSP woss::operator+ (const double left, const SSP &right)
- const SSP woss::operator- (const double left, const SSP &right)
- const SSP woss::operator/ (const double left, const SSP &right)
- const SSP woss::operator* (const double left, const SSP &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const SSP &instance)
- inline ::std::istream & **woss::operator**>> (::std::istream &is, SSP &instance)

14.135.1 Detailed Description

Definitions and library for Sound Speed Profiles.

Author

Federico Guerra

Definitions and library for woss::SSP (Sound Speed profile) class

14.135.2 Typedef Documentation

14.135.2.1 DepthMap `typedef ::std::map< PDouble, double > woss::DepthMap`

Multipurpose map that links a depth [m] to a value (temperature [C°], pressure [bar], salinity [ppu], or sound speed profile [m/s])

14.135.3 Function Documentation

14.135.3.1 operator"!=() `bool woss::operator!= ( const SSP & left, const SSP & right ) [inline]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.135.3.2 operator∗() [1/3] `const SSP woss::operator* ( const double left, const SSP & right ) [inline]`

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.3 operator∗() [2/3] const SSP woss::operator* (const SSP & left, const double right) [inline]

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.4 operator∗() [3/3] const SSP woss::operator* (const SSP & left, const SSP & right) [inline]

Multiplication operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.5 operator*=() [1/2]

```
SSP & woss::operator*= (
    SSP & left,
    const double right )
```

Compound assignment multiplication operator

Parameters

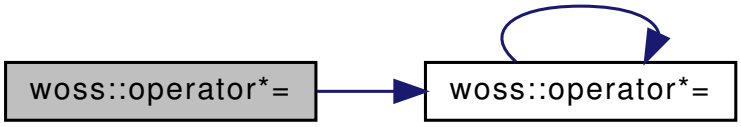
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator*=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.6 operator*=() [2/2]

```
SSP & woss::operator*= (
    SSP & left,
    const SSP & right )
```

Compound assignment multiplication operator

Parameters

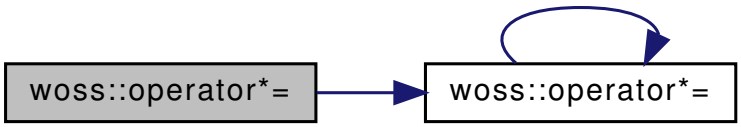
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator*=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.7 operator+() [1/3] const SSP woss::operator+ (const double *left*, const SSP & *right*) [inline]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.8 operator+() [2/3] const SSP woss::operator+ (const SSP & *left*, const double *right*) [inline]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.9 operator+() [3/3] const SSP woss::operator+ (const SSP & *left*, const SSP & *right*) [inline]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a const instance holding the operation result

14.135.3.10 operator+=() [1/2] SSP & woss::operator+= (SSP & *left*, const double *right*)

Compound assignment sum operator

Parameters

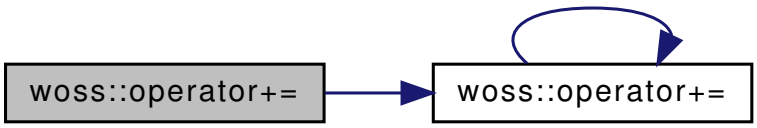
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator+=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.11 operator+=() [2/2] SSP & woss::operator+= (SSP & *left*, const SSP & *right*)

Compound assignment sum operator

Parameters

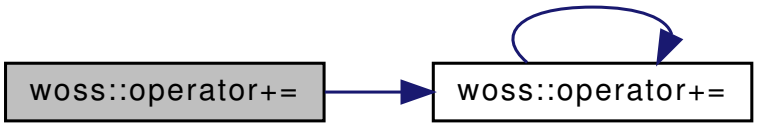
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator+=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.12 operator-() [1/3] const SSP woss::operator- (const double *left*, const SSP & *right*) [inline]

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.13 operator-() [2/3] const SSP woss::operator- (const SSP & *left*, const double *right*) [inline]

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.14 operator-() [3/3]

```
const SSP woss::operator- (
    const SSP & left,
    const SSP & right )  [inline]
```

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.15 operator-=() [1/2]

```
SSP & woss::operator-= (
    SSP & left,
    const double right )
```

Compound assignment subtraction operator

Parameters

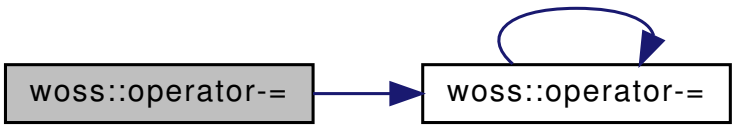
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator-=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.16 operator-=() [2/2]

```
SSP & woss::operator-= (
    SSP & left,
    const SSP & right )
```

Compound assignment subtraction operator

Parameters

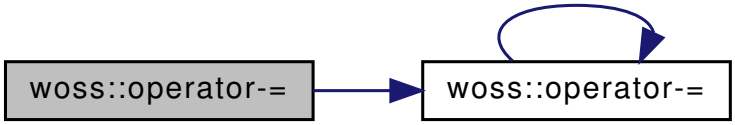
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator-=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.17 operator/() [1/3]

```
const SSP woss::operator/ (
    const double left,
    const SSP & right )  [inline]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.18 operator/() [2/3]

```
const SSP woss::operator/ (
    const SSP & left,
    const double right )  [inline]
```

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.19 operator/() [3/3] const SSP woss::operator/ (const SSP & *left*, const SSP & *right*) [inline]

Division operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.135.3.20 operator/=() [1/2] SSP & woss::operator/= (SSP & *left*, const double *right*)

Compound assignment division operator

Parameters

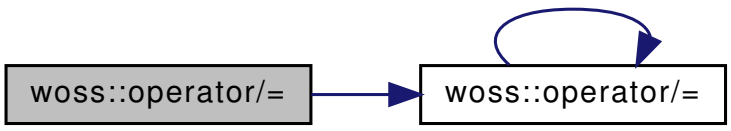
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator/=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.21 operator/=() [2/2]

```
SSP & woss::operator/= (
    SSP & left,
    const SSP & right )
```

Compound assignment division operator

Parameters

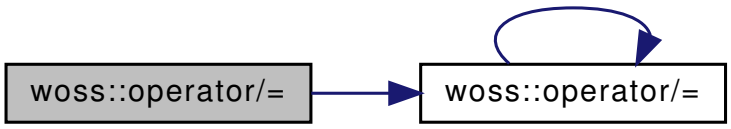
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator/=(), woss::SSP::pressure_map, woss::SSP::salinity_map, woss::SSP::ssp_map, and woss::SSP::temperature_map.

Here is the call graph for this function:



14.135.3.22 operator==()

```
bool woss::operator== (
    const SSP & left,
    const SSP & right )  [inline]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

14.136 ssp-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
```

```
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_SSP_DEFINITIONS_H
41 #define WOSS_SSP_DEFINITIONS_H
42
43
44 #include <cassert>
45 #include <cstdlib>
46 #include <map>
47 #include "coordinates-definitions.h"
48 #include "pressure-definitions.h"
49 #include "custom-precision-double.h"
50 #include "definitions-handler.h"
51
52
53 namespace woss {
54
55
56   // UNESCO CHEN AND MILLERO EQUATIONS COEFFICIENTS
57 #define cf_C00 (1402.388)
58 #define cf_A02 (7.166e-5)
59 #define cf_C01 (5.03830)
60 #define cf_A03 (2.008e-6)
61 #define cf_C02 (-5.81090e-2)
62 #define cf_A04 (-3.21e-8)
63 #define cf_C03 (3.3432e-4)
64 #define cf_A10 (9.4742e-5)
65 #define cf_C04 (-1.47797e-6)
66 #define cf_A11 (-1.2583e-5)
67 #define cf_C05 (3.1419e-9)
68 #define cf_A12 (-6.4928e-8)
69 #define cf_C10 (0.153563)
70 #define cf_A13 (1.0515e-8)
71 #define cf_C11 (6.8999e-4)
72 #define cf_A14 (-2.0142e-10)
73 #define cf_C12 (-8.1829e-6)
74 #define cf_A20 (-3.9064e-7)
75 #define cf_C13 (1.3632e-7)
76 #define cf_A21 (9.1061e-9)
77 #define cf_C14 (-6.1260e-10)
78 #define cf_A22 (-1.6009e-10)
79 #define cf_C20 (3.1260e-5)
80 #define cf_A23 (7.994e-12)
81 #define cf_C21 (-1.7111e-6)
82 #define cf_A30 (1.100e-10)
83 #define cf_C22 (2.5986e-8)
84 #define cf_A31 (6.651e-12)
85 #define cf_C23 (-2.5353e-10)
86 #define cf_A32 (-3.391e-13)
87 #define cf_C24 (1.0415e-12)
88 #define cf_B00 (-1.922e-2)
89 #define cf_C30 (-9.7729e-9)
90 #define cf_B01 (-4.42e-5)
91 #define cf_C31 (3.8513e-10)
92 #define cf_B10 (7.3637e-5)
93 #define cf_C32 (-2.3654e-12)
94 #define cf_B11 (1.7950e-7)
95 #define cf_A00 (1.389)
96 #define cf_D00 (1.727e-3)
97 #define cf_A01 (-1.262e-2)
98 #define cf_D10 (-7.9836e-6)
99
100   // TEOS-10 constants
101 #define TEO_10_c000  -6.07991438090e-5
102 #define TEO_10_c001  1.99712338438e-5
```

```
103 #define TEO_10_c002  -3.39280843110e-6
104 #define TEO_10_c003  4.21246123200e-7
105 #define TEO_10_c004  -6.32363064300e-8
106 #define TEO_10_c005  1.17681023580e-8
107 #define TEO_10_c010  1.85057654290e-5
108 #define TEO_10_c011  -2.34727734620e-6
109 #define TEO_10_c012  -1.09581019659e-6
110 #define TEO_10_c013  1.25816399608e-6
111 #define TEO_10_c020  -1.17166068530e-5
112 #define TEO_10_c021  4.26100574800e-6
113 #define TEO_10_c022  8.60877154770e-7
114 #define TEO_10_c030  7.92796561730e-6
115 #define TEO_10_c031  -9.22650800740e-7
116 #define TEO_10_c040  -3.41021874820e-6
117 #define TEO_10_c041  -1.26705833028e-7
118 #define TEO_10_c050  5.07367668140e-7
119 #define TEO_10_c100  2.42624687470e-5
120 #define TEO_10_c101  -1.16968865968e-6
121 #define TEO_10_c102  1.08930565545e-6
122 #define TEO_10_c103  -4.45885016920e-7
123 #define TEO_10_c110  -9.56770881560e-6
124 #define TEO_10_c111  -1.11398309114e-5
125 #define TEO_10_c112  -8.18870887110e-7
126 #define TEO_10_c120  -2.36783083610e-7
127 #define TEO_10_c121  7.82747741600e-7
128 #define TEO_10_c130  -3.45587736550e-6
129 #define TEO_10_c131  1.55237776184e-8
130 #define TEO_10_c140  1.29567177830e-6
131 #define TEO_10_c200  -3.47924609740e-5
132 #define TEO_10_c201  -9.62445031940e-6
133 #define TEO_10_c202  5.02389113400e-8
134 #define TEO_10_c210  1.11008347650e-5
135 #define TEO_10_c211  1.09241497668e-5
136 #define TEO_10_c220  2.92833462950e-6
137 #define TEO_10_c221  -1.31462208134e-6
138 #define TEO_10_c230  3.16553060780e-7
139 #define TEO_10_c300  3.74707773050e-5
140 #define TEO_10_c301  9.85262139960e-6
141 #define TEO_10_c310  -9.84471178440e-6
142 #define TEO_10_c311  -2.70883712540e-6
143 #define TEO_10_c320  -4.88261392000e-7
144 #define TEO_10_c400  -1.73222186120e-5
145 #define TEO_10_c401  -3.56239494540e-6
146 #define TEO_10_c410  2.59092252600e-6
147 #define TEO_10_c500  3.09274272530e-6
148 #define TEO_10_v000  1.0769995862e-3
149 #define TEO_10_v001  -6.0799143809e-5
150 #define TEO_10_v002  9.9856169219e-6
151 #define TEO_10_v003  -1.1309361437e-6
152 #define TEO_10_v004  1.0531153080e-7
153 #define TEO_10_v005  -1.2647261286e-8
154 #define TEO_10_v006  1.9613503930e-9
155 #define TEO_10_v010  -3.1038981976e-4
156 #define TEO_10_v011  2.4262468747e-5
157 #define TEO_10_v012  -5.8484432984e-7
158 #define TEO_10_v013  3.6310188515e-7
159 #define TEO_10_v014  -1.1147125423e-7
160 #define TEO_10_v020  6.6928067038e-4
161 #define TEO_10_v021  -3.4792460974e-5
162 #define TEO_10_v022  -4.8122251597e-6
163 #define TEO_10_v023  1.6746303780e-8
164 #define TEO_10_v030  -8.5047933937e-4
165 #define TEO_10_v031  3.7470777305e-5
166 #define TEO_10_v032  4.9263106998e-6
167 #define TEO_10_v040  5.8086069943e-4
168 #define TEO_10_v041  -1.7322218612e-5
169 #define TEO_10_v042  -1.7811974727e-6
170 #define TEO_10_v050  -2.1092370507e-4
171 #define TEO_10_v051  3.0927427253e-6
172 #define TEO_10_v060  3.1932457305e-5
173 #define TEO_10_v100  -1.5649734675e-5
174 #define TEO_10_v101  1.8505765429e-5
175 #define TEO_10_v102  -1.1736386731e-6
176 #define TEO_10_v103  -3.6527006553e-7
177 #define TEO_10_v104  3.1454099902e-7
178 #define TEO_10_v110  3.5009599764e-5
179 #define TEO_10_v111  -9.5677088156e-6
180 #define TEO_10_v112  -5.5699154557e-6
181 #define TEO_10_v113  -2.7295696237e-7
182 #define TEO_10_v120  -4.3592678561e-5
183 #define TEO_10_v121  1.1100834765e-5
184 #define TEO_10_v122  5.4620748834e-6
185 #define TEO_10_v130  3.4532461828e-5
186 #define TEO_10_v131  -9.8447117844e-6
187 #define TEO_10_v132  -1.3544185627e-6
188 #define TEO_10_v140  -1.1959409788e-5
189 #define TEO_10_v141  2.5909225260e-6
```

```
190 #define TEO_10_v150  1.3864594581e-6
191 #define TEO_10_v200  2.7762106484e-5
192 #define TEO_10_v201  -1.1716606853e-5
193 #define TEO_10_v202  2.1305028740e-6
194 #define TEO_10_v203  2.8695905159e-7
195 #define TEO_10_v210  -3.7435842344e-5
196 #define TEO_10_v211  -2.3678308361e-7
197 #define TEO_10_v212  3.9137387080e-7
198 #define TEO_10_v220  3.5907822760e-5
199 #define TEO_10_v221  2.9283346295e-6
200 #define TEO_10_v222  -6.5731104067e-7
201 #define TEO_10_v230  -1.8698584187e-5
202 #define TEO_10_v231  -4.8826139200e-7
203 #define TEO_10_v240  3.8595339244e-6
204 #define TEO_10_v300  -1.6521159259e-5
205 #define TEO_10_v301  7.9279656173e-6
206 #define TEO_10_v302  -4.6132540037e-7
207 #define TEO_10_v310  2.4141479483e-5
208 #define TEO_10_v311  -3.4558773655e-6
209 #define TEO_10_v312  7.7618888092e-9
210 #define TEO_10_v320  -1.4353633048e-5
211 #define TEO_10_v321  3.1655306078e-7
212 #define TEO_10_v330  2.2863324556e-6
213 #define TEO_10_v400  6.9111322702e-6
214 #define TEO_10_v401  -3.4102187482e-6
215 #define TEO_10_v402  -6.3352916514e-8
216 #define TEO_10_v410  -8.7595873154e-6
217 #define TEO_10_v411  1.2956717783e-6
218 #define TEO_10_v420  4.3703680598e-6
219 #define TEO_10_v500  -8.0539615540e-7
220 #define TEO_10_v501  5.0736766814e-7
221 #define TEO_10_v510  -3.3052758900e-7
222 #define TEO_10_v600  2.0543094268e-7
223
224 #define TEO_10_gsw_sfac (0.0248826675584615)
225 #define TEO_10_offset  (5.971840214030754e-1)
226
231   typedef ::std::map< PDouble, double > DepthMap;
232   typedef DepthMap::const_iterator DConstIter;
233   typedef DepthMap::iterator DIter;
234   typedef DepthMap::reverse_iterator DRIter;
235   typedef DepthMap::const_reverse_iterator DConstRIter;
236
237
242 #define SSP_CUSTOM_DEPTH_PRECISION (1.0e-6)
243
248 #define SSP_CUSTOM_DEPTH_STEPS (20.0)
249
250
258   class SSP {
259
260
261     public:
262
266     enum SSPEqType {
267       SSP_EQ_CHEN_MILLERO = 0,
268       SSP_EQ_TEOS_10 = 1,
269       SSP_EQ_TEOS_10_EXACT = 2,
270       SSP_EQ_INVALID
271     };
272
277     SSP( long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION );
278
287     SSP( DepthMap& ssp_map, DepthMap& temp_map, DepthMap& sal_map, DepthMap& press_map, long double
    depth_precision = SSP_CUSTOM_DEPTH_PRECISION );
288
294     SSP( DepthMap& ssp_map, long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION );
295
300     SSP( const SSP& copy );
301
302     virtual ~SSP() { }
303
304
310     virtual SSP* create( long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION )const { return new
    SSP(depth_precision); }
311
321     virtual SSP* create( DepthMap& ssp_map, DepthMap& temp_map, DepthMap& sal_map, DepthMap& press_map,
    long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION )const {
322       return new SSP( ssp_map, temp_map, sal_map, press_map, depth_precision); }
323
330     virtual SSP* create( DepthMap& ssp_map, long double depth_precision = SSP_CUSTOM_DEPTH_PRECISION
    )const {
331       return new SSP( ssp_map, depth_precision); }
332
338     virtual SSP* create( const SSP& copy )const { return new SSP( copy ); }
339
344     virtual SSP* clone()const { return new SSP(*this); }
```

```
345
346
351     int size()const { return( ssp_map.size() ); }
352
353
358     bool empty()const { return ssp_map.empty(); }
359
360
364     void clear() { ssp_map.clear(); temperature_map.clear(); pressure_map.clear(); salinity_map.clear();
    }
365
366
371     DConstIter begin()const { return( ssp_map.begin() ); }
372
377     DConstIter end()const { return( ssp_map.end() ); }
378
383     DConstRIter rbegin()const { return( ssp_map.rbegin() ); }
384
389     DConstRIter rend()const { return( ssp_map.rend() ); }
390
391
397     DConstIter lower_bound( const PDouble& depth )const { return( ssp_map.lower_bound(depth) ); }
398
404     DConstIter upper_bound( const PDouble& depth )const { return( ssp_map.upper_bound(depth) ); }
405
411     DConstIter at( const int i ) const ;
412
413
418     DConstIter pressure_begin()const { return( pressure_map.begin() ); }
419
424     DConstIter pressure_end()const { return( pressure_map.end() ); }
425
430     DConstRIter pressure_rbegin()const { return( pressure_map.rbegin() ); }
431
436     DConstRIter pressure_rend()const { return( pressure_map.rend() ); }
437
438
444     DConstIter pressure_lower_bound( const PDouble& depth )const { return(
    pressure_map.lower_bound(depth) ); }
445
451     DConstIter pressure_upper_bound( const PDouble& depth )const { return(
    pressure_map.upper_bound(depth) ); }
452
458     DConstIter pressure_find( const PDouble& depth )const { return( pressure_map.find(depth) ); }
459
460
465     DConstIter temperature_begin()const { return( temperature_map.begin() ); }
466
471     DConstIter temperature_end()const { return( temperature_map.end() ); }
472
477     DConstRIter temperature_rbegin()const { return( temperature_map.rbegin() ); }
478
483     DConstRIter temperature_rend()const { return( temperature_map.rend() ); }
484
485
491     DConstIter temperature_lower_bound( const PDouble& depth )const { return(
    temperature_map.lower_bound(depth) ); }
492
498     DConstIter temperature_upper_bound( const PDouble& depth )const { return(
    temperature_map.upper_bound(depth) ); }
499
505     DConstIter temperature_find( const PDouble& depth )const { return( temperature_map.find(depth) ); }
506
507
512     DConstIter salinity_begin()const { return( salinity_map.begin() ); }
513
518     DConstIter salinity_end()const { return( salinity_map.end() ); }
519
524     DConstRIter salinity_rbegin()const { return( salinity_map.rbegin() ); }
525
530     DConstRIter salinity_rend()const { return( salinity_map.rend() ); }
531
532
538     DConstIter salinity_lower_bound( const PDouble& depth )const { return(
    salinity_map.lower_bound(depth) ); }
539
545     DConstIter salinity_upper_bound( const PDouble& depth )const { return(
    salinity_map.upper_bound(depth) ); }
546
552     DConstIter salinity_find( const PDouble& depth )const { return( salinity_map.find(depth) ); }
553
554
559     virtual bool isValid()const { return( !ssp_map.empty() ); }
560
565     virtual bool isTransformable()const { return( isValid() /*&& !temperature_map.empty() &&
    !pressure_map.empty() && !salinity_map.empty() */ ); }
566
```

```
571     virtual bool isRandomizable()const { return( isValid() && !temperature_map.empty() &&
      !pressure_map.empty() && !salinity_map.empty() ); }
572
573
585     virtual SSP* transform( const Coord& coordinates, double new_min_depth = -HUGE_VAL, double
      new_max_depth = HUGE_VAL, int total_depth_steps = SSP_CUSTOM_DEPTH_STEPS ) const;
586
593     virtual SSP* truncate( double max_depth ) const;
594
601     virtual SSP* fullRandomize( double ratio_incr_value ) const;
602
608     virtual SSP* randomize( double ratio_incr_value ) const;
609
610
616     virtual bool import( ::std::istream& stream_in );
617
623     virtual bool write( ::std::ostream& stream_out ) const;
624
625
632     SSP& insertValue( double depth, double ssp_value );
633
643     SSP& insertValue( double depth, double temperature, double salinity, const ::std::complex<double>&
      pressure, double ssp_value );
644
653     SSP& insertValue( double temperature, double salinity, const ::std::complex<double>& pressure, const
      Coord& coordinates = Coord( 0.0, 0.0) );
654
663     SSP& insertValue( double depth, double temperature, double salinity, const Coord& coordinates =
      Coord(0.0, 0.0) );
664
665
671     DConstIter findValue( const double& depth )const { return( ssp_map.find(depth) ); }
672
673
679     SSP& eraseValue( const double& depth ) { ssp_map.erase(depth); return *this; }
680
681
686     double getMaxDepthValue()const { if ( ssp_map.rbegin() != ssp_map.rend() ) return
      ssp_map.rbegin()->first;
687                                      else return 0.0; }
688
693     double getMinDepthValue()const { if ( ssp_map.begin() != ssp_map.end() ) return
      ssp_map.begin()->first;
694                                      else return 0.0; }
695
696
701     double getMaxSSPValue()const { return( max_ssp_value ); }
702
707     double getMinSSPValue()const { return( min_ssp_value ); }
708
709
715     virtual void setDepthPrecision( long double prec );
716
721     long double getDepthPrecision()const { return depth_precision; }
722
728     SSP& setSSPEqType(SSPEqType eq_type) { ssp_eq_type = eq_type; return *this; }
729
734     SSPEqType getSSPEqType()const {return ssp_eq_type; }
735
741     SSP& operator=( const SSP& x ) ;
742
743
750     friend bool operator==( const SSP& left, const SSP& right );
751
758     friend bool operator!=( const SSP& left, const SSP& right );
759
760
767     friend const SSP operator+( const SSP& left, const SSP& right );
768
775     friend const SSP operator-( const SSP& left, const SSP& right );
776
783     friend const SSP operator*( const SSP& left, const SSP& right );
784
791     friend const SSP operator/( const SSP& left, const SSP& right );
792
793
800     friend const SSP operator+( const SSP& left, const double right );
801
808     friend const SSP operator-( const SSP& left, const double right );
809
816     friend const SSP operator/( const SSP& left, const double right );
817
824     friend const SSP operator*( const SSP& left, const double right );
825
826
833     friend const SSP operator+( const double left, const SSP& right );
834
```

```
841     friend const SSP operator-( const double left, const SSP& right );
842
849     friend const SSP operator/( const double left, const SSP& right );
850
857     friend const SSP operator*( const double left, const SSP& right );
858
859
866     friend SSP& operator+=( SSP& left, const SSP& right );
867
874     friend SSP& operator-=( SSP& left, const SSP& right );
875
882     friend SSP& operator*=( SSP& left, const SSP& right );
883
890     friend SSP& operator/=( SSP& left, const SSP& right );
891
892
899     friend SSP& operator+=( SSP& left, const double right );
900
907     friend SSP& operator-=( SSP& left, const double right );
908
915     friend SSP& operator/=( SSP& left, const double right );
916
923     friend SSP& operator*=( SSP& left, const double right );
924
925
932     friend ::std::ostream& operator«( ::std::ostream& os, const SSP& instance );
933
940     friend ::std::ostream& operator»( ::std::istream& is, const SSP& instance );
941
942
947     static void setDebug( bool flag ) { debug = flag; }
948
949
950     protected:
951
955     SSPEqType ssp_eq_type;
956
960     double min_ssp_value;
961
965     double max_ssp_value;
966
970     long double depth_precision;
971
972
976     static bool debug;
977
978
982     DepthMap ssp_map;
983
987     DepthMap pressure_map;
988
992     DepthMap salinity_map;
993
997     DepthMap temperature_map;
998
999
1007     virtual double calculateSSP( double temperature, double salinity, double pressure ) const;
1008
1009
1010     private:
1011
1012
1018     bool isCanonOcean( const Coord& coordinates )const { return( coordinates.getLatitude() >= -40.0 &&
      coordinates.getLatitude() <= 60.0 ); }
1019
1025     bool isNEAtlanticOcean( const Coord& coordinates )const { return( coordinates.getLatitude() >= 30.0
      && coordinates.getLatitude() <= 60.0
1026                                                                       && coordinates.getLongitude() >=
      -43.0 && coordinates.getLongitude() <= -5.0 ); }
1027
1033     bool isAntarcticOcean( const Coord& coordinates )const { return( coordinates.getLatitude() <= -55.0
      ); }
1034
1040     bool isMediterraneanSea( const Coord& coordinates )const { return( coordinates.getLatitude() >=
      30.1 && coordinates.getLatitude() <= 46.0
1041                                                                       && coordinates.getLongitude() >=
      -4.0 && coordinates.getLongitude() <= 37.0 ); }
1042
1048     bool isRedSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 12.0 &&
      coordinates.getLatitude() <= 30.0
1049                                                    && coordinates.getLongitude() >= 32.0 &&
      coordinates.getLongitude() <= 44.0 ); }
1050
1056     bool isArcticOcean( const Coord& coordinates )const { return( coordinates.getLatitude() >= 65.0 );
      }
1057
1063     bool isJapanSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 33.0 &&
```

```
    coordinates.getLatitude() <= 52.0
1064                                                    && coordinates.getLongitude() >= 127.0 &&
    coordinates.getLongitude() <= 142.0 ); }
1065
1071    bool isSuluSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 5.0 &&
    coordinates.getLatitude() <= 13.0
1072                                                    && coordinates.getLongitude() >= 117.0 &&
    coordinates.getLongitude() <= 123.0 ); }
1073
1079    bool isHalmaheraSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= -2.0 &&
    coordinates.getLatitude() <= 1.0
1080                                                       && coordinates.getLongitude() >= 127.0
    && coordinates.getLongitude() <= 130.0 ); }
1081
1087    bool isCelebesSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 0.0 &&
    coordinates.getLatitude() <= 7.0
1088                                                      && coordinates.getLongitude() >= 115.0 &&
    coordinates.getLongitude() <= 125.0 ); }
1089
1095    bool isBlackSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 41.0 &&
    coordinates.getLatitude() <= 48.0
1096                                                    && coordinates.getLongitude() >= 28.0 &&
    coordinates.getLongitude() <= 42.0 ); }
1097
1103    bool isBalticSea( const Coord& coordinates )const { return( coordinates.getLatitude() >= 53.0 &&
    coordinates.getLatitude() <= 66.0
1104                                                      && coordinates.getLongitude() >= 20.0 &&
    coordinates.getLongitude() <= 26.0 ); }
1105
1110    double thyh( double z ) const ;
1111
1116    double g( double lat ) const ;
1117
1123    double k( double z, double lat ) const ;
1124
1129    double hq( double z ) const ;
1130
1136    double h( double z, double lat ) const ;
1137
1143    double getPressureCorreptions( const Coord& coordinates, double depth ) const ;
1144
1151    double getPressureFromDepth( const Coord& coordinates, double depth ) const ;
1152
1153
1158    double g_z( double lat ) const ;
1159
1165    double getDepthCorreptions( const Coord& coordinates, double pressure ) const ;
1166
1173    double getDepthfromPressure( const Coord& coordinates, double pressure ) const ;
1174
1175
1181    double d( double t, double p ) const ;
1182
1188    double b( double t, double p ) const ;
1189
1195    double a( double t, double p  ) const ;
1196
1202    double cw( double t, double p ) const ;
1203
1213    double gibbs(int ns, int nt, int np, double sa, double t, double p) const ;
1214
1215  };
1216
1217  //non-inline operator declarations
1219
1220  SSP& operator+=( SSP& left, const SSP& right );
1221
1222  SSP& operator-=( SSP& left, const SSP& right );
1223
1224  SSP& operator*=( SSP& left, const SSP& right );
1225
1226  SSP& operator/=( SSP& left, const SSP& right );
1227
1228
1229  SSP& operator+=( SSP& left, const double right );
1230
1231  SSP& operator-=( SSP& left, const double right );
1232
1233  SSP& operator/=( SSP& left, const double right );
1234
1235  SSP& operator*=( SSP& left, const double right );
1236
1237
1238  //inline functions
1240  inline bool operator==( const SSP& left, const SSP& right ) {
1241    if ( &left == &right ) return true;
1242    return( left.ssp_map == right.ssp_map && left.temperature_map == right.temperature_map &&
```

```
      left.pressure_map == right.pressure_map
1243         && left.salinity_map == right.salinity_map );
1244   }
1245
1246
1247   inline bool operator!=( const SSP& left, const SSP& right ) {
1248     if ( &left == &right ) return false;
1249     return( left.ssp_map != right.ssp_map || left.temperature_map != right.temperature_map ||
      left.pressure_map != right.pressure_map
1250         || left.salinity_map == right.salinity_map ); }
1251
1252
1253   inline const SSP operator+( const SSP& left, const SSP& right ) {
1254     SSP ret_val ( left );
1255     ret_val += right;
1256     return ret_val;
1257   }
1258
1259
1260   inline const SSP operator-( const SSP& left, const SSP& right ) {
1261     SSP ret_val ( left );
1262     ret_val -= right;
1263     return ret_val;
1264   }
1265
1266
1267   inline const SSP operator*( const SSP& left, const SSP& right ) {
1268     SSP ret_val ( left );
1269     ret_val *= right;
1270
1271     return ret_val;
1272   }
1273
1274
1275   inline const SSP operator/( const SSP& left, const SSP& right ) {
1276     SSP ret_val ( left );
1277     ret_val /= right;
1278     return ret_val;
1279   }
1280
1281
1282   inline const SSP operator+( const SSP& left, const double right ) {
1283     SSP ret_val ( left );
1284     ret_val += right;
1285     return ret_val;
1286   }
1287
1288
1289   inline const SSP operator-( const SSP& left, const double right ) {
1290     SSP ret_val ( left );
1291     ret_val -= right;
1292     return ret_val;
1293   }
1294
1295
1296   inline const SSP operator/( const SSP& left, const double right ) {
1297     SSP ret_val ( left );
1298     ret_val /= right;
1299     return ret_val;
1300   }
1301
1302
1303   inline const SSP operator*( const SSP& left, const double right ) {
1304     SSP ret_val ( left );
1305     ret_val *= right;
1306     return ret_val;
1307   }
1308
1309
1310   inline const SSP operator+( const double left, const SSP& right ) {
1311     SSP ret_val ( right );
1312     ret_val += left;
1313     return ret_val;
1314   }
1315
1316
1317   inline const SSP operator-( const double left, const SSP& right ) {
1318     SSP ret_val ( right );
1319     ret_val -= left;
1320     return ret_val;
1321   }
1322
1323
1324   inline const SSP operator/( const double left, const SSP& right ) {
1325     SSP ret_val ( right );
1326     ret_val /= left;
1327     return ret_val;
```

```
1328   }
1329
1330
1331   inline const SSP operator*( const double left, const SSP& right ) {
1332     SSP ret_val ( right );
1333     ret_val *= left;
1334     return ret_val;
1335   }
1336
1337
1338   inline ::std::ostream& operator«( ::std::ostream& os, const SSP& instance ) {
1339     if ( !instance.write( os ) ) os.setstate( ::std::ios_base::failbit );
1340     return os;
1341   }
1342
1343
1344   inline ::std::istream& operator»( ::std::istream& is, SSP& instance ) {
1345     if ( !instance.import( is ) ) is.setstate( ::std::ios_base::failbit );
1346     return is;
1347   }
1348
1349
1350   inline double SSP::thyh( double z )const {
1351     return (1.0e-2 * z/(z+100) + 6.2e-6 * z);
1352   }
1353
1354
1355   inline double SSP::g( double lat )const {
1356     return( 9.7803*(1.0 + 5.3e-3 * pow(sin(lat),2.0)) );
1357   }
1358
1359
1360   inline double SSP::k( double z, double lat )const {
1361     return( (g(lat) - 2e-5 * z) / (9.80612 - 2e-5 * z) );
1362   }
1363
1364
1365   inline double SSP::hq( double z )const {
1366     return( 1.00818e-2 * z + 2.465e-8 * pow(z,2.0) - 1.25e-13 * pow(z,3.0) + 2.8e-19 * pow(z,4.0) );
1367   }
1368
1369
1370   inline double SSP::h( double z, double lat )const {
1371     return( hq(z) * k(z,lat) );
1372   }
1373
1374
1375   inline double SSP::getPressureFromDepth( const Coord& coordinates, double depth )const {
1376     // conversion in bar from MPa
1377     return( 10.0 * ( h(depth,coordinates.getLatitude()) - thyh(depth) -
     getPressureCorreptions(coordinates, depth) ) );
1378   }
1379
1380
1381   inline double SSP::g_z( double lat )const {
1382     return( 9.780318 * ( 1.0 + 5.2788 * 10e-3 * pow(sin(lat), 2.0) + 2.36 * 10e-5 * pow(sin(lat),4.0) )
     );
1383   }
1384
1385
1386   inline double SSP::getDepthfromPressure( const Coord& coordinates, double pressure )const {
1387     // conversion from bar to MPa
1388     pressure /= 10.0;
1389     return( ( (9.72659e2 * pressure - 2.2512e-1 * pow(pressure, 2.0) + 2.279e-4 * pow(pressure, 3.0) -
     1.82e-7 * pow(pressure, 4.0) )
1390     / ( g_z(coordinates.getLatitude()) + 1.092e-4 * pressure ) ) + getDepthCorreptions(coordinates,
     pressure)  );
1391   }
1392
1393
1394
1395   inline double SSP::calculateSSP(double temperature, double salinity, double pressure)const {
1396     if (ssp_eq_type == SSP_EQ_CHEN_MILLERO)
1397     {
1398       return( cw(temperature, pressure) + a(temperature, pressure)*salinity
1399         + b(temperature, pressure) * ::std::sqrt(salinity*salinity*salinity)
1400         + d(temperature, pressure) * (salinity*salinity) );
1401     }
1402     else if (ssp_eq_type == SSP_EQ_TEOS_10)
1403     {
1404       double  v, v_p, xs, ys, z;
1405
1406       pressure *= 10; // formula requires dbar instead of bar
1407
1408       xs  = ::std::sqrt(TEO_10_gsw_sfac*salinity + TEO_10_offset);
1409       ys  = temperature*0.025;
1410       z = pressure*1e-4;
```

```
1411
1412        v = TEO_10_v000
1413          + xs*(TEO_10_v010 + xs*(TEO_10_v020 + xs*(TEO_10_v030 + xs*(TEO_10_v040 + xs*(TEO_10_v050
1414          + TEO_10_v060*xs))))) + ys*(TEO_10_v100 + xs*(TEO_10_v110 + xs*(TEO_10_v120 + xs*(TEO_10_v130 +
      xs*(TEO_10_v140
1415          + TEO_10_v150*xs)))) + ys*(TEO_10_v200 + xs*(TEO_10_v210 + xs*(TEO_10_v220 + xs*(TEO_10_v230 +
      TEO_10_v240*xs)))
1416          + ys*(TEO_10_v300 + xs*(TEO_10_v310 + xs*(TEO_10_v320 + TEO_10_v330*xs)) + ys*(TEO_10_v400 +
      xs*(TEO_10_v410
1417          + TEO_10_v420*xs) + ys*(TEO_10_v500 + TEO_10_v510*xs + TEO_10_v600*ys))))) + z*(TEO_10_v001 +
      xs*(TEO_10_v011
1418          + xs*(TEO_10_v021 + xs*(TEO_10_v031 + xs*(TEO_10_v041 + TEO_10_v051*xs)))) + ys*(TEO_10_v101 +
      xs*(TEO_10_v111
1419          + xs*(TEO_10_v121 + xs*(TEO_10_v131 + TEO_10_v141*xs))) + ys*(TEO_10_v201 + xs*(TEO_10_v211 +
      xs*(TEO_10_v221
1420          + TEO_10_v231*xs)) + ys*(TEO_10_v301 + xs*(TEO_10_v311 + TEO_10_v321*xs) + ys*(TEO_10_v401 +
      TEO_10_v411*xs
1421          + TEO_10_v501*ys)))) + z*(TEO_10_v002 + xs*(TEO_10_v012 + xs*(TEO_10_v022 + xs*(TEO_10_v032 +
      TEO_10_v042*xs)))
1422          + ys*(TEO_10_v102 + xs*(TEO_10_v112 + xs*(TEO_10_v122 + TEO_10_v132*xs)) + ys*(TEO_10_v202 +
      xs*(TEO_10_v212
1423          + TEO_10_v222*xs) + ys*(TEO_10_v302 + TEO_10_v312*xs + TEO_10_v402*ys))) + z*(TEO_10_v003 +
      xs*(TEO_10_v013
1424          + TEO_10_v023*xs) + ys*(TEO_10_v103 + TEO_10_v113*xs + TEO_10_v203*ys) + z*(TEO_10_v004 +
      TEO_10_v014*xs + TEO_10_v104*ys
1425          + z*(TEO_10_v005 + TEO_10_v006*z))));
1426
1427        v_p = TEO_10_c000
1428          + xs*(TEO_10_c100 + xs*(TEO_10_c200 + xs*(TEO_10_c300 + xs*(TEO_10_c400 + TEO_10_c500*xs))))
1429          + ys*(TEO_10_c010 + xs*(TEO_10_c110 + xs*(TEO_10_c210 + xs*(TEO_10_c310 + TEO_10_c410*xs))) +
      ys*(TEO_10_c020
1430          + xs*(TEO_10_c120 + xs*(TEO_10_c220 + TEO_10_c320*xs)) + ys*(TEO_10_c030 + xs*(TEO_10_c130 +
      TEO_10_c230*xs)
1431          + ys*(TEO_10_c040 + TEO_10_c140*xs + TEO_10_c050*ys)))) + z*(TEO_10_c001 + xs*(TEO_10_c101 +
      xs*(TEO_10_c201
1432          + xs*(TEO_10_c301 + TEO_10_c401*xs))) + ys*(TEO_10_c011 + xs*(TEO_10_c111 + xs*(TEO_10_c211 +
      TEO_10_c311*xs))
1433          + ys*(TEO_10_c021 + xs*(TEO_10_c121 + TEO_10_c221*xs) + ys*(TEO_10_c031 + TEO_10_c131*xs +
      TEO_10_c041*ys)))
1434          + z*( TEO_10_c002 + xs*(TEO_10_c102 + TEO_10_c202*xs) + ys*(TEO_10_c012 + TEO_10_c112*xs +
      TEO_10_c022*ys)
1435          + z*(TEO_10_c003 + TEO_10_c103*xs + TEO_10_c013*ys + z*(TEO_10_c004 + TEO_10_c005*z))));
1436
1437        return (10000.0*::std::sqrt(-v*v/v_p));
1438      }
1439      else if (ssp_eq_type == SSP_EQ_TEOS_10_EXACT)
1440      {
1441        int n0=0, n1=1, n2=2;
1442        double  g_tt, g_tp;
1443
1444        pressure *= 10; // formulas require dbar instead of bar
1445
1446        g_tt  = gibbs(n0, n2, n0, salinity, temperature, pressure);
1447        g_tp  = gibbs(n0, n1, n1, salinity, temperature, pressure);
1448
1449        return (gibbs(n0, n0, n1, salinity, temperature, pressure) * ::std::sqrt(g_tt/(g_tp*g_tp -
      g_tt*gibbs(n0, n0, n2, salinity, temperature, pressure))));
1450      }
1451      else return -HUGE_VAL;
1452    }
1453
1454    inline double SSP::gibbs(int ns, int nt, int np, double sa, double t, double p)const
1455 {
1456      double x2, x, y, z, g03, g08, return_value = 0.0;
1457
1458      x2  = TEO_10_gsw_sfac*sa;
1459      x = sqrt(x2);
1460      y = t*0.025;
1461      z = p*1e-4;
1462
1463      if (ns == 0  && nt == 0  && np == 0) {
1464        g03 = 101.342743139674 + z*(100015.695367145 +
1465            z*(-2544.5765420363 + z*(284.517778446287 +
1466            z*(-33.3146754253611 + (4.20263108803084 -
1467                0.546428511471039*z)*z)))) +
1468            y*(5.90578347909402 + z*(-270.983805184062 +
1469            z*(776.153611613101 + z*(-196.51255088122 +
1470                (28.9796526294175 - 2.13290083518327*z)*z))) +
1471            y*(-12357.785933039 + z*(1455.0364540468 +
1472            z*(-756.558385769359 + z*(273.479662323528 +
1473                z*(-55.5604063817218 + 4.34420671917197*z)))) +
1474            y*(736.741204151612 + z*(-672.50778314507 +
1475            z*(499.360390819152 + z*(-239.545330654412 +
1476                (48.8012518593872 - 1.66307106208905*z)*z))) +
1477            y*(-148.185936433658 + z*(397.968445406972 +
1478            z*(-301.815380621876 + (152.196371733841 -
1479                26.3748377232802*z)*z)) +
```

```
1480            y*(58.0259125842571 + z*(-194.618310617595 +
1481            z*(120.520654902025 + z*(-55.2723052340152 +
1482                6.48190668077221*z))) +
1483            y*(-18.9843846514172 + y*(3.05081646487967 -
1484                9.63108119393062*z) +
1485            z*(63.5113936641785 + z*(-22.2897317140459 +
1486                8.17060541818112*z))))))));
1487
1488        g08 = x2*(1416.27648484197 + z*(-3310.49154044839 +
1489            z*(384.794152978599 + z*(-96.5324320107458 +
1490              (15.8408172766824 - 2.62480156590992*z)*z))) +
1491            x*(-2432.14662381794 + x*(2025.80115603697 +
1492            y*(543.835333000098 + y*(-68.5572509204491 +
1493            y*(49.3667694856254 + y*(-17.1397577419788 +
1494                2.49697009569508*y))) - 22.6683558512829*z) +
1495            x*(-1091.66841042967 - 196.028306689776*y +
1496            x*(374.60123787784 - 48.5891069025409*x +
1497                36.7571622995805*y) + 36.028419561086*z) +
1498            z*(-54.7919133532887 + (-4.08193978912261 -
1499                30.1755111971161*z)*z)) +
1500            z*(199.459603073901 + z*(-52.2940909281335 +
1501                (68.0444942726459 - 3.41251932441282*z)*z)) +
1502            y*(-493.407510141682 + z*(-175.292041186547 +
1503                (83.1923927801819 - 29.483064349429*z)*z) +
1504            y*(-43.0664675978042 + z*(383.058066002476 +
1505                z*(-54.1917262517112 + 25.6398487389914*z)) +
1506            y*(-10.0227370861875 - 460.319931801257*z +
1507                y*(0.875600661808945 + 234.565187611355*z))))) +
1508            y*(168.072408311545 + z*(729.116529735046 +
1509            z*(-343.956902961561 + z*(124.687671116248 +
1510                z*(-31.656964386073 + 7.046588033154 49*z)))) +
1511            y*(880.031352997204 + y*(-225.267649263401 +
1512            y*(91.4260447751259 + y*(-21.6603240875311 +
1513                2.13016970847183*y) +
1514            z*(-297.728741987187 + (74.726141138756 -
1515                36.4872919001588*z)*z)) +
1516            z*(694.244814133268 + z*(-204.889641964903 +
1517                (113.561697840594 - 11.1282734326413*z)*z))) +
1518            z*(-860.764303783977 + z*(337.409530269367 +
1519            z*(-178.314556207638 + (44.2040358308 -
1520                7.92001547211682*z)*z))))));
1521
1522        if (sa > 0.0)
1523          g08 = g08 + x2*(5812.81456626732 +
1524          851.226734946706*y)*log(x);
1525
1526        return_value      = g03 + g08;
1527      }
1528      else if (ns == 1  && nt == 0  && np == 0) {
1529        g08       = 8645.36753595126 + z*(-6620.98308089678 +
1530            z*(769.588305957198 + z*(-193.0648640214916 +
1531                (31.6816345533648 - 5.249603131819 84*z)*z))) +
1532            x*(-7296.43987145382 + x*(8103.20462414788 +
1533            y*(2175.341332000392 + y*(-274.2290036817964 +
1534            y*(197.4670779425016 + y*(-68.5590309679152 +
1535                9.98788038278032*y))) - 90.6734234051316*z) +
1536            x*(-5458.34205214835 - 980.14153344888*y +
1537            x*(2247.60742726704 - 340.1237483177863*x +
1538                220.542973797483*y) + 180.142097805543*z) +
1539            z*(-219.1676534131548 + (-16.32775915649044 -
1540                120.7020447884644*z)*z)) +
1541            z*(598.378809221703 + z*(-156.8822727844005 +
1542                (204.1334828179377 - 10.23755797323846*z)*z)) +
1543            y*(-1480.222530425046 + z*(-525.876123559641 +
1544                (249.57717834054571 - 88.449193048287*z)*z) +
1545            y*(-129.1994027934126 + z*(1149.174198007428 +
1546                z*(-162.5751787551336 + 76.9195462169742*z)) +
1547            y*(-30.0682112585625 - 1380.9597954037708*z +
1548                y*(2.626801985426835 + 703.695562834065*z))))) +
1549            y*(1187.3715515697959 + z*(1458.233059470092 +
1550            z*(-687.913805923122 + z*(249.375342232496 +
1551                z*(-63.313928772146 + 14.09317606630898*z)))) +
1552            y*(1760.062705994408 + y*(-450.535298526802 +
1553            y*(182.8520895502518 + y*(-43.3206481750622 +
1554                4.26033941694366*y) +
1555            z*(-595.457483974374 + (149.452282277512 -
1556                72.9745838003176*z)*z)) +
1557            z*(1388.489628266536 + z*(-409.779283929806 +
1558                (227.123395681188 - 22.2565468652826*z)*z))) +
1559            z*(-1721.528607567954 + z*(674.819060538734 +
1560            z*(-356.629112415276 + (88.4080716616 -
1561                15.84003094423364*z)*z))))));
1562
1563        if (sa > 0.0)
1564          g08 = g08 + (11625.62913253464 + 1702.453469893412*y)*log(x);
1565        else
1566          g08 = 0.0;
```

```
1567
1568        return_value      = 0.5*TEO_10_gsw_sfac*g08;
1569      }
1570      else if (ns == 0  && nt == 1  && np == 0) {
1571        g03 = 5.90578347909402 + z*(-270.983805184062 +
1572            z*(776.153611613101 + z*(-196.51255088122 +
1573                (28.9796526294175 - 2.13290083518327*z)*z))) +
1574            y*(-24715.571866078 + z*(2910.0729080936 +
1575            z*(-1513.116771538718 + z*(546.959324647056 +
1576                z*(-111.1208127634436 + 8.68841343834394*z)))) +
1577            y*(2210.2236124548363 + z*(-2017.52334943521 +
1578            z*(1498.081172457456 + z*(-718.6359919632359 +
1579                (146.4037555781616 - 4.9892131862671505*z)*z))) +
1580            y*(-592.743745734632 + z*(1591.873781627888 +
1581            z*(-1207.261522487504 + (608.785486935364 -
1582                105.4993508931208*z)*z)) +
1583            y*(290.12956292128547 + z*(-973.091553087975 +
1584            z*(602.603274510125 + z*(-276.361526170076 +
1585                32.409533403861O5*z))) +
1586            y*(-113.90630790850321 + y*(21.35571525415769 -
1587                67.41756835751434*z) +
1588            z*(381.06836198507096 + z*(-133.7383902842754 +
1589                49.023632509086724*z)))))));
1590
1591        g08 = x2*(168.072408311545 + z*(729.116529735046 +
1592            z*(-343.956902961561 + z*(124.687671116248 +
1593                z*(-31.656964386073 + 7.046588033154 49*z)))) +
1594            x*(-493.407510141682 + x*(543.835333000098 +
1595                x*(-196.028306689776 + 36.7571622995805*x) +
1596            y*(-137.1145018408982 + y*(148.10030845687618 +
1597                y*(-68.5590309679152 + 12.4848504784754*y))) -
1598                22.6683558512829*z) + z*(-175.292041186547 +
1599                (83.1923927801819 - 29.483064349429*z)*z) +
1600            y*(-86.1329351956084 + z*(766.116132004952 +
1601                z*(-108.3834525034224 + 51.2796974779828*z)) +
1602            y*(-30.0682112585625 - 1380.9597954037708*z +
1603                y*(3.50240264723578 + 938.26075044542*z)))) +
1604            y*(1760.062705994408 + y*(-675.802947790203 +
1605            y*(365.7041791005036 + y*(-108.30162043765552 +
1606                12.78101825083098*y) +
1607            z*(-1190.914967948748 + (298.904564555024 -
1608                145.94916760063 52*z)*z)) +
1609            z*(2082.7344423998043 + z*(-614.668925894709 +
1610                (340.685093521782 - 33.3848202979239*z)*z))) +
1611            z*(-1721.528607567954 + z*(674.819060538734 +
1612            z*(-356.629112415276 + (88.4080716616 -
1613                15.840030944233 64*z)*z)))));
1614
1615        if (sa > 0.0)
1616          g08 = g08 + 851.226734946706*x2*log(x);
1617
1618        return_value      = (g03 + g08)*0.025;
1619      }
1620      else if (ns == 0  && nt == 0  && np == 1) {
1621        g03 = 100015.695367145 + z*(-5089.1530840726 +
1622            z*(853.5533353388611 + z*(-133.2587017014444 +
1623                (21.0131554401542 - 3.278571068826234*z)*z))) +
1624            y*(-270.983805184062 + z*(1552.307223226202 +
1625            z*(-589.53765264366 + (115.91861051767 -
1626                10.6645041759163 49*z)*z)) +
1627            y*(1455.0364540468 + z*(-1513.116771538718 +
1628            z*(820.438986970584 + z*(-222.2416255268872 +
1629                21.721033595859 85*z))) +
1630            y*(-672.50778314507 + z*(998.720781638304 +
1631            z*(-718.6359919632359 + (195.2050074375488 -
1632                8.31535531044525*z)*z)) +
1633            y*(397.968445406972 + z*(-603.630761243752 +
1634                (456.589115201523 - 105.4993508931208*z)*z) +
1635            y*(-194.618310617595 + y*(63.5113936641785 -
1636                9.63108119393062*y +
1637            z*(-44.5794634280918 + 24.511816254543362*z)) +
1638            z*(241.04130980405 + z*(-165.8169157020456 +
1639            25.927626723088 84*z)))))));
1640
1641        g08 = x2*(-3310.49154044839 + z*(769.588305957198 +
1642            z*(-289.5972960322374 + (63.3632691067296 -
1643                13.1240078295496*z)*z)) +
1644            x*(199.459603073901 + x*(-54.7919133532887 +
1645                36.0284195611086*x - 22.6683558512829*y +
1646            (-8.16387957824522 - 90.52653359134831*z)*z) +
1647            z*(-104.588181856267 + (204.1334828179377 -
1648                13.65007729765128*z)*z) +
1649            y*(-175.292041186547 + (166.3847855603638 -
1650                88.449193048287*z)*z +
1651            y*(383.058066002476 + y*(-460.319931801257 +
1652                234.565187611355*y) +
1653            z*(-108.3834525034224 + 76.9195462169742*z)))) +
```

```
1654            y*(729.116529735046 + z*(-687.913805923122 +
1655            z*(374.063013348744 + z*(-126.627857544292 +
1656                35.23294016577245*z))) +
1657            y*(-860.764303783977 + y*(694.244814133268 +
1658            y*(-297.728741987187 + (149.452282277512 -
1659                109.46187570047641*z)*z) +
1660            z*(-409.779283929806 + (340.685093521782 -
1661                44.5130937305652*z)*z)) +
1662            z*(674.819060538734 + z*(-534.943668622914 +
1663                (176.8161433232 - 39.600077360584095*z)*z))))));
1664
1665      return_value      = (g03 + g08)*1.0e-8;
1666    }
1667    else if (ns == 0  && nt == 2  && np == 0) {
1668      g03 = -24715.571866078 + z*(2910.0729080936 + z*
1669          (-1513.116771538718 + z*(546.959324647056 +
1670            z*(-111.1208127634436 + 8.68841343834394*z)))) +
1671          y*(4420.4472249096725 + z*(-4035.04669887042 +
1672          z*(2996.162344914912 + z*(-1437.2719839264719 +
1673              (292.8075111563232 - 9.978426372534301*z)*z))) +
1674          y*(-1778.231237203896 + z*(4775.621344883664 +
1675          z*(-3621.784567462512 + (1826.356460806092 -
1676              316.498052679362 44*z)*z)) +
1677          y*(1160.5182516851419 + z*(-3892.3662123519 +
1678          z*(2410.4130980405 + z*(-1105.446104680304 +
1679              129.6381336154442*z))) +
1680          y*(-569.531539542516 + y*(128.13429152494615 -
1681              404.50541014508605*z) +
1682          z*(1905.341809925355 + z*(-668.691951421377 +
1683              245.11816254543362*z))))));
1684
1685      g08 = x2*(1760.062705994408 + x*(-86.1329351956084 +
1686          x*(-137.1145018408982 + y*(296.20061691375236 +
1687              y*(-205.67709290374563 + 49.9394019139016*y))) +
1688          z*(766.116132004952 + z*(-108.3834525034224 +
1689              51.2796974779828*z)) +
1690          y*(-60.136422517125 - 2761.9195908075417*z +
1691              y*(10.50720794170734 + 2814.78225133626*z))) +
1692          y*(-1351.605895580406 + y*(1097.1125373015109 +
1693              y*(-433.20648175062206 + 63.9050912541549 04*y) +
1694          z*(-3572.7449038462437 + (896.713693665072 -
1695              437.84750280190565*z)*z)) +
1696          z*(4165.4688847996085 + z*(-1229.337851789418 +
1697              (681.370187043564 - 66.7696405958478*z)*z))) +
1698          z*(-1721.528607567954 + z*(674.819060538734 +
1699          z*(-356.629112415276 + (88.4080716616 -
1700              15.840030944233 64*z)*z))));
1701
1702      return_value = (g03 + g08)*0.000625;
1703    }
1704    else if (ns == 1  && nt == 0  && np == 1) {
1705      g08 = -6620.98308089678 + z*(1539.176611914396 +
1706          z*(-579.1945920644748 + (126.7265382134592 -
1707              26.2480156590992*z)*z)) +
1708          x*(598.378809221703 + x*(-219.1676534131548 +
1709              180.142097805543*x - 90.6734234051316*y +
1710          (-32.65551831298088 - 362.10613436539325*z)*z) +
1711          z*(-313.764545568801 + (612.4004484538132 -
1712              40.95023189295384*z)*z) +
1713          y*(-525.876123559641 + (499.15435668109143 -
1714              265.347579144861*z)*z +
1715          y*(1149.174198007428 + y*(-1380.9597954037708 +
1716              703.695562834065*y) +
1717          z*(-325.1503575102672 + 230.7586386509226*z)))) +
1718          y*(1458.233059470092 + z*(-1375.827611846244 +
1719          z*(748.126026697488 + z*(-253.255715088584 +
1720              70.4658803315449*z))) +
1721          y*(-1721.528607567954 + y*(1388.489628266536 +
1722          y*(-595.457483974374 + (298.904564555024 -
1723              218.92375140095282*z)*z) +
1724          z*(-819.558567859612 + (681.370187043564 -
1725              89.0261874611304*z)*z)) +
1726          z*(1349.638121077468 + z*(-1069.887337245828 +
1727              (353.6322866464 - 79.20015472116819*z)*z))));
1728
1729      return_value = g08*TEO_10_gsw_sfac*0.5e-8;
1730    }
1731    else if (ns == 0  && nt == 1  && np == 1) {
1732      g03 = -270.983805184062 + z*(1552.307223226202 +
1733          z*(-589.53765264366 + (115.91861051767 -
1734              10.6645041759163 49*z)*z)) +
1735          y*(2910.0729080936 + z*(-3026.233543077436 +
1736          z*(1640.877973941168 + z*(-444.4832510537744 +
1737              43.4420671917197*z))) +
1738          y*(-2017.52334943521 + z*(2996.162344914912 +
1739          z*(-2155.907975889708 + (585.6150223126464 -
1740              24.946065931335752*z)*z)) +
```

```
1741            y*(1591.873781627888 + z*(-2414.523044975008 +
1742                (1826.356460806092 - 421.9974035724832*z)*z) +
1743            y*(-973.091553087975 + z*(1205.20654902025 +
1744                z*(-829.084578510228 + 129.6381336154442*z)) +
1745            y*(381.06836198507096 - 67.41756835751434*y +
1746                z*(-267.4767805685508 + 147.070897526017*z))))));
1747
1748        g08 = x2*(729.116529735046 + z*(-687.913805923122 +
1749            z*(374.063013348744 + z*(-126.627857544292 +
1750                35.23294016577245*z))) +
1751            x*(-175.292041186547 - 22.6683558512829*x +
1752                (166.3847855603638 - 88.449193048287*z)*z +
1753            y*(766.116132004952 + y*(-1380.9597954037708 +
1754                938.26075044542*y) +
1755            z*(-216.7669050068448 + 153.8390924339484*z))) +
1756            y*(-1721.528607567954 + y*(2082.7344423998043 +
1757            y*(-1190.914967948748 + (597.809129110048 -
1758                437.84750280190565*z)*z) +
1759            z*(-1229.337851789418 + (1022.055280565346 -
1760                133.53928119169 56*z)*z)) +
1761            z*(1349.638121077468 + z*(-1069.887337245828 +
1762                (353.6322866464 - 79.20015472116819*z)*z))));
1763
1764        return_value = (g03 + g08)*2.5e-10;
1765      }
1766      else if (ns == 1  && nt == 1  && np == 0) {
1767        g08 = 1187.3715515697959 + z*(1458.233059470092 +
1768            z*(-687.913805923122 + z*(249.375342232496 +
1769            z*(-63.313928772146 + 14.09317606630898*z)))) +
1770            x*(-1480.222530425046 + x*(2175.341332000392 +
1771            x*(-980.14153344888 + 220.542973797483*x) +
1772            y*(-548.4580073635929 + y*(592.4012338275047 +
1773            y*(-274.2361238716608 + 49.9394019139016*y))) -
1774            90.6734234051316*z) +
1775            z*(-525.876123559641 + (249.57717834054571 -
1776            88.449193048287*z)*z) +
1777                  y*(-258.3988055868252 + z*(2298.348396014856 +
1778            z*(-325.1503575102672 + 153.8390924339484*z)) +
1779            y*(-90.2046337756875 - 4142.87938621131 25*z +
1780            y*(10.50720794170734 + 2814.78225133626*z)))) +
1781            y*(3520.125411988816 + y*(-1351.605895580406 +
1782            y*(731.4083582010072 + y*(-216.60324087531103 +
1783            25.56203650166196*y) +
1784            z*(-2381.829935897496 + (597.809129110048 -
1785            291.8983352012704*z)*z)) +
1786            z*(4165.4688847996085 + z*(-1229.337851789418 +
1787            (681.370187043564 - 66.7696405958478*z)*z))) +
1788            z*(-3443.057215135908 + z*(1349.638121077468 +
1789            z*(-713.258224830552 + (176.8161433232 -
1790            31.68006188846728*z)*z))));
1791
1792        if (sa > 0.0)
1793          g08 = g08 + 1702.453469893412*log(x);
1794
1795        return_value = 0.5*TEO_10_gsw_sfac*0.025*g08;
1796      }
1797      else if (ns == 2  && nt == 0  && np == 0) {
1798        g08 = 2.0*(8103.20462414788 +
1799            y*(2175.341332000392 + y*(-274.2290036817964 +
1800            y*(197.4670779425016 + y*(-68.5590309679152 +
1801            9.98788038278032*y))) - 90.6734234051316*z) +
1802            1.5*x*(-5458.34205214835 - 980.14153344888*y +
1803            (4.0/3.0)*x*(2247.60742726704 - 340.1237483177863*1.25*x +
1804            220.542973797483*y) + 180.142097805543*z) +
1805            z*(-219.1676534131548 + (-16.32775915649044 -
1806            120.7020447884644*z)*z));
1807
1808        if (x > 0.0) {
1809          g08 += (-7296.43987145382 + z*(598.378809221703 +
1810              z*(-156.8822727844005 + (204.1334828179377 -
1811              10.23755797323846*z)*z)) +
1812              y*(-1480.222530425046 + z*(-525.876123559641 +
1813              (249.57717834054571 - 88.449193048287*z)*z) +
1814              y*(-129.1994027934126 + z*(1149.174198007428 +
1815              z*(-162.5751787551336 + 76.9195462169742*z)) +
1816              y*(-30.0682112585625 - 1380.9597954037708*z +
1817              y*(2.626801985426835 + 703.695562834065*z)))))/x +
1818              (11625.62913253464 + 1702.453469893412*y)/x2;
1819        } else
1820          g08 = 0.0;
1821
1822        return_value = 0.25*TEO_10_gsw_sfac*TEO_10_gsw_sfac*g08;
1823      }
1824      else if (ns == 0  && nt == 0  && np == 2) {
1825        g03 = -5089.1530840726 + z*(1707.1066706777221 +
1826            z*(-399.7761051043332 + (84.0526217606168 -
1827                16.39285534413117*z)*z)) +
```

```
1828            y*(1552.307223226202 + z*(-1179.07530528732 +
1829                (347.75583155301 - 42.658016703665396*z)*z) +
1830            y*(-1513.116771538718 + z*(1640.877973941168 +
1831                z*(-666.7248765806615 + 86.8841343834394*z)) +
1832            y*(998.720781638304 + z*(-1437.2719839264719 +
1833                (585.6150223126464 - 33.261421241781*z)*z) +
1834            y*(-603.630761243752 + (913.178230403046 -
1835                316.49805267936244*z)*z +
1836            y*(241.04130980405 + y*(-44.5794634280918 +
1837                49.023632509086724*z) +
1838            z*(-331.6338314040912 + 77.78288016926652*z))))));
1839
1840        g08 = x2*(769.588305957198 + z*(-579.1945920644748 +
1841            (190.08980732018878 - 52.4960313181984*z)*z) +
1842            x*(-104.588181856267 + x*(-8.16387957824522 -
1843                181.05306718269662*z) +
1844            (408.2669656358754 - 40.95023189295384*z)*z +
1845            y*(166.3847855603638 - 176.898386096574*z +
1846                y*(-108.3834525034224 + 153.8390924339484*z))) +
1847            y*(-687.913805923122 + z*(748.126026697488 +
1848                z*(-379.883572632876 + 140.9317606630898*z)) +
1849            y*(674.819060538734 + z*(-1069.887337245828 +
1850                (530.4484299696 - 158.40030944233638*z)*z) +
1851            y*(-409.779283929806 + y*(149.452282277512 -
1852                218.92375140095282*z) +
1853            (681.370187043564 - 133.5392811916956*z)*z))));
1854
1855        return_value = (g03 + g08)*1e-16 ;
1856      }
1857      else
1858        return_value = -HUGE_VAL;
1859
1860      return (return_value);
1861    }
1862
1863    inline double SSP::d(double t, double p)const {
1864      return( cf_D00 + cf_D10*p );
1865    }
1866
1867
1868    inline double SSP::b( double t, double p)const {
1869      return( cf_B00 + cf_B01*t + (cf_B10 + cf_B11*t)*p );
1870    }
1871
1872
1873    inline double SSP::a(double t, double p)const {
1874      return( (cf_A00 + cf_A01*t + cf_A02*pow(t,2.0) + cf_A03*pow(t,3.0) + cf_A04*pow(t,4.0))
1875      + (cf_A10 + cf_A11*t + cf_A12*pow(t,2.0) + cf_A13*pow(t,3.0) + cf_A14*pow(t,4.0))*p
1876      + (cf_A20 + cf_A21*t + cf_A22*pow(t,2.0) + cf_A23*pow(t,3.0))*pow(p,2.0)
1877      + (cf_A30 + cf_A31*t + cf_A32*pow(t,2.0))*pow(p,3.0) );
1878    }
1879
1880
1881    inline double SSP::cw( double t, double p)const {
1882      return( (cf_C00 + cf_C01*t + cf_C02*pow(t,2.0) + cf_C03*pow(t,3.0) + cf_C04*pow(t,4.0) +
       cf_C05*pow(t,5.0))
1883      + (cf_C10 + cf_C11*t + cf_C12*pow(t,2.0) + cf_C13*pow(t,3.0) + cf_C14*pow(t,4.0))*p
1884      + (cf_C20 + cf_C21*t + cf_C22*pow(t,2.0) + cf_C23*pow(t,3.0) + cf_C24*pow(t,4.0))*pow(p,2.0)
1885      + (cf_C30 + cf_C31*t + cf_C32*pow(t,2.0))*pow(p,3.0) );
1886    }
1887
1888 }
1889
1890
1891 #endif /* WOSS_SSP_DEFINITIONS_H */
1892
1893
```

14.137 woss/woss_def/time-arrival-definitions.cpp File Reference

Implementations and library for woss::TimeArr class.

14.137.1 Detailed Description

Implementations and library for woss::TimeArr class.

Author

Federico Guerra

Implementations and library for woss::TimeArr class

14.138 woss/woss_def/time-arrival-definitions.h File Reference

Definitions and library for woss::TimeArr class.

Classes

- class woss::TimeArr
  Channel power delay profile class.

Typedefs

- typedef std::map< PDouble, std::complex< double > > woss::TimeArrMap
- typedef TimeArrMap::iterator **woss::TimeArrIt**
- typedef TimeArrMap::const_iterator **woss::TimeArrCIt**
- typedef TimeArrMap::reverse_iterator **woss::TimeArrRIt**
- typedef TimeArrMap::const_reverse_iterator **woss::TimeArrCRIt**

Functions

- const TimeArr woss::operator+ (const TimeArr &left, const TimeArr &right)
- const TimeArr woss::operator- (const TimeArr &left, const TimeArr &right)
- const TimeArr woss::operator+ (const TimeArr &left, const double right)
- const TimeArr woss::operator- (const TimeArr &left, const double right)
- const TimeArr woss::operator/ (const TimeArr &left, const double right)
- const TimeArr woss::operator* (const TimeArr &left, const double right)
- const TimeArr woss::operator+ (const double left, const TimeArr &right)
- const TimeArr woss::operator- (const double left, const TimeArr &right)
- const TimeArr woss::operator/ (const double left, const TimeArr &right)
- const TimeArr woss::operator* (const double left, const TimeArr &right)
- TimeArr & woss::operator+= (TimeArr &left, const TimeArr &right)
- TimeArr & woss::operator-= (TimeArr &left, const TimeArr &right)
- TimeArr & woss::operator+= (TimeArr &left, double right)
- TimeArr & woss::operator-= (TimeArr &left, double right)
- TimeArr & woss::operator/= (TimeArr &left, double right)
- TimeArr & woss::operator*= (TimeArr &left, double right)
- std::ostream & woss::operator<< (std::ostream &os, const TimeArr &instance)
- bool woss::operator== (const TimeArr &left, const TimeArr &right)
- bool woss::operator!= (const TimeArr &left, const TimeArr &right)

14.138.1 Detailed Description

Definitions and library for woss::TimeArr class.

Author

Federico Guerra

Definitions and library for woss::TimeArr class

14.138.2 Typedef Documentation

14.138.2.1 TimeArrMap `typedef std::map< PDouble , std::complex<double> > woss::TimeArrMap`

Map that links a PDouble delay [s] to a complex Pressure

14.138.3 Function Documentation

14.138.3.1 operator"!=() `bool woss::operator!= ( const TimeArr & left, const TimeArr & right ) [inline]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.138.3.2 operator*() [1/2] `const TimeArr woss::operator* ( const double left, const TimeArr & right )`

Multiplication operator

Parameters

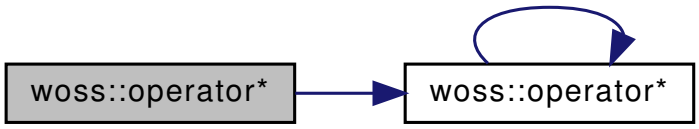
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator∗().

Here is the call graph for this function:



14.138.3.3 operator∗() [2/2]

```
const TimeArr woss::operator* (
    const TimeArr & left,
    const double right )
```

Multiplication operator

Parameters

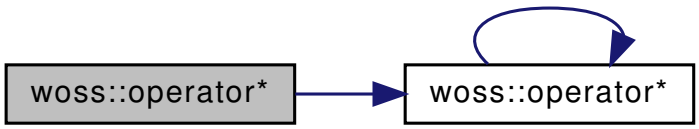
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator∗().

Here is the call graph for this function:



14.138.3.4 operator*=() `TimeArr & woss::operator*= ( TimeArr & left, double right )`

Compound assignment multiplication operator

Parameters

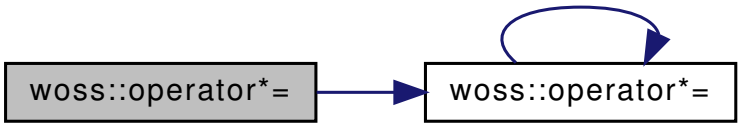
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator*=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.5 operator+() [1/3] `const TimeArr woss::operator+ ( const double left, const TimeArr & right )`

Sum operator

Parameters

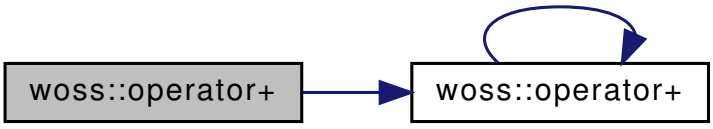
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator+().

Here is the call graph for this function:



14.138.3.6 operator+() [2/3]

```
const TimeArr woss::operator+ (
    const TimeArr & left,
    const double right )
```

Sum operator

Parameters

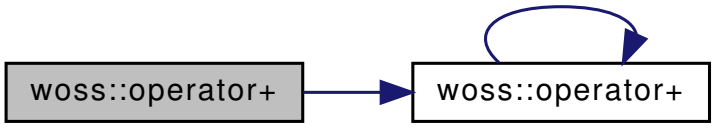
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator+().

Here is the call graph for this function:



14.138.3.7 operator+() [3/3]

```
const TimeArr woss::operator+ (
    const TimeArr & left,
    const TimeArr & right )
```

Sum operator

Parameters

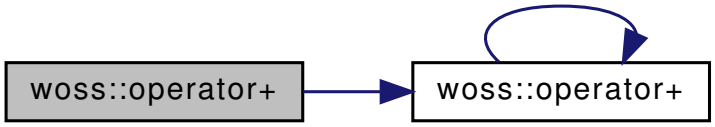
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator+().

Here is the call graph for this function:



14.138.3.8 operator+=() [1/2]

```
TimeArr & woss::operator+= (
    TimeArr & left,
    const TimeArr & right )
```

Compound assignment sum operator

Parameters

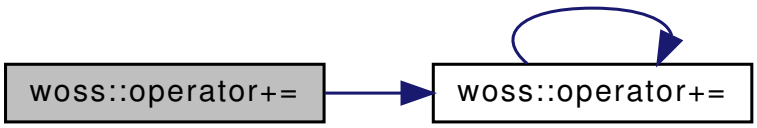
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator+=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.9 operator+=() [2/2]

```
TimeArr & woss::operator+= (
    TimeArr & left,
    double right )
```

Compound assignment sum operator

Parameters

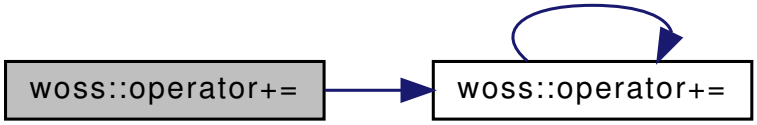
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator+=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.10 operator-() [1/3]

```
const TimeArr woss::operator- (
    const double left,
    const TimeArr & right )
```

Subtraction operator

Parameters

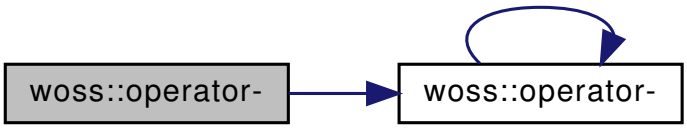
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator-().

Here is the call graph for this function:



14.138.3.11 operator-() [2/3]

```
const TimeArr woss::operator- (
    const TimeArr & left,
    const double right )
```

Subtraction operator

Parameters

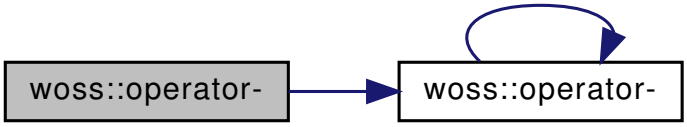
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator-().

Here is the call graph for this function:



14.138.3.12 operator-() [3/3]

```
const TimeArr woss::operator- (
    const TimeArr & left,
    const TimeArr & right )
```

Subtraction operator

Parameters

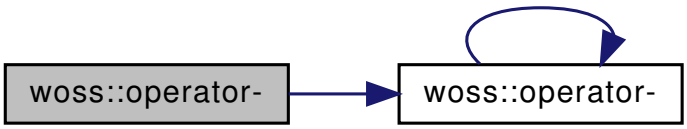
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator-().

Here is the call graph for this function:



14.138.3.13 operator-=() [1/2]

```
TimeArr & woss::operator-= (
    TimeArr & left,
    const TimeArr & right )
```

Compound assignment subtraction operator

Parameters

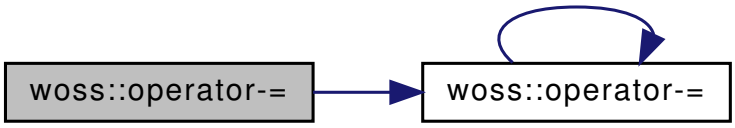
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator-=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.14 operator-=() [2/2]

```
TimeArr & woss::operator-= (
    TimeArr & left,
    double right )
```

Compound assignment subtraction operator

Parameters

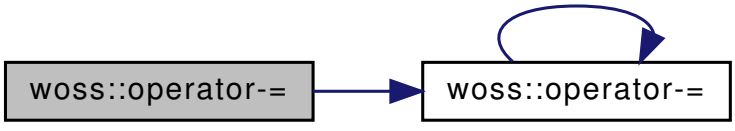
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator-=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.15 operator/() [1/2]

```
const TimeArr woss::operator/ (
    const double left,
    const TimeArr & right )
```

Division operator

Parameters

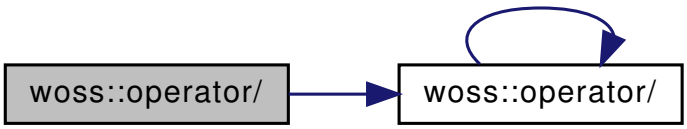
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator/().

Here is the call graph for this function:



14.138.3.16 operator/() [2/2]

```
const TimeArr woss::operator/ (
    const TimeArr & left,
    const double right )
```

Division operator

Parameters

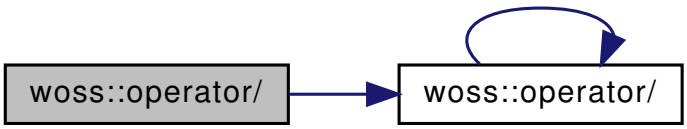
left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

References woss::operator/().

Here is the call graph for this function:



14.138.3.17 operator/=() TimeArr & woss::operator/= (TimeArr & *left*, double *right*)

Compound assignment division operator

Parameters

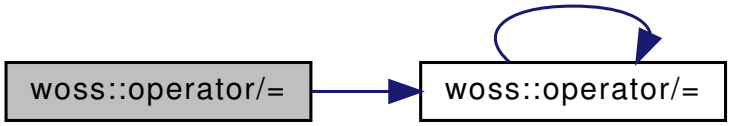
left	left operand reference
right	right operand const reference

Returns

left reference after the operation

References woss::operator/=(), and woss::TimeArr::time_arr_map.

Here is the call graph for this function:



14.138.3.18 operator<<() std::ostream & woss::operator<< (std::ostream & *os*, const TimeArr & *instance*) [inline]

<< operator

Parameters

os	left operand ostream reference
instance	right operand const Pressure reference

Returns

os reference after the operation

14.138.3.19 operator==() bool woss::operator== (const TimeArr & *left*, const TimeArr & *right*) [inline]

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.139 time-arrival-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef TIME_ARRIVAL_DEFINITIONS_H
41 #define TIME_ARRIVAL_DEFINITIONS_H
42
43
44 #include <cassert>
45 #include <climits>
46 #include <map>
47 #include "pressure-definitions.h"
48 #include "custom-precision-double.h"
49
50
51 namespace woss {
52
53
57   typedef std::map < PDouble , std::complex<double> > TimeArrMap;
58   typedef TimeArrMap::iterator TimeArrIt;
59   typedef TimeArrMap::const_iterator TimeArrCIt;
60   typedef TimeArrMap::reverse_iterator TimeArrRIt;
61   typedef TimeArrMap::const_reverse_iterator TimeArrCRIt;
62
63
67 #define TIMEARR_CUSTOM_DELAY_PRECISION (1.0e-7)
68
69
73 #define TIMEARR_PRESSURE_CONVERSION_DELAY (-INT_MAX)
74
75
82   class TimeArr {
83
84
85     public:
86
87
92     TimeArr( long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION );
93
99     TimeArr( TimeArrMap& map, long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION );
```

```
100
107     TimeArr( const Pressure& pressure, double delay = TIMEARR_PRESSURE_CONVERSION_DELAY, long double
    custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION );
108
113     TimeArr( const TimeArr& copy );
114
115     virtual ~TimeArr() { }
116
117
122     virtual operator std::complex<double>() const;
123
124
130     virtual TimeArr* create( long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION )const {
    return new TimeArr(custom_delay_prec); }
131
138     virtual TimeArr* create( TimeArrMap& map, long double custom_delay_prec =
    TIMEARR_CUSTOM_DELAY_PRECISION )const {
139       return new TimeArr( map, custom_delay_prec); }
140
148     virtual TimeArr* create( const Pressure& pressure, double delay = TIMEARR_PRESSURE_CONVERSION_DELAY,
    long double custom_delay_prec = TIMEARR_CUSTOM_DELAY_PRECISION )const {
149       return new TimeArr(pressure, delay, custom_delay_prec ); }
150
156     virtual TimeArr* create( const TimeArr& copy )const { return new TimeArr( copy ); }
157
162     virtual TimeArr* clone()const { return new TimeArr(*this); }
163
169     virtual TimeArr* createArray( unsigned int array_size )const { return new TimeArr[array_size]; }
170
175     static TimeArrMap& createNotValid();
176
181     static TimeArrMap& createImpulse();
182
183
190     TimeArr& insertValue( double delay, const Pressure& pressure );
191
197     void sumValue( double delay, const Pressure& pressure );
198
199
205     TimeArrCIt findValue( double delay )const { return( time_arr_map.find(delay) ); }
206
207
213     TimeArr& eraseValue( double delay ) { time_arr_map.erase(delay); return *this; }
214
215
222     virtual TimeArr* coherentSumSample( double time_delay );
223
230     virtual TimeArr* incoherentSumSample( double time_delay );
231
239     virtual TimeArr* crop( double time_start, double time_end );
240
248     virtual bool checkPressureAttenuation( double distance, double frequency );
249
250
255     TimeArrCIt begin()const { return time_arr_map.begin(); }
256
261     TimeArrCIt end()const { return time_arr_map.end(); }
262
267     TimeArrCRIt rbegin()const { return time_arr_map.rbegin(); }
268
273     TimeArrCRIt rend()const { return time_arr_map.rend(); }
274
280     TimeArrCIt at( const int i ) const ;
281
287                 TimeArrCIt lowerBoundTxLoss( double threshold_db ) const;
288
293     int size()const { return time_arr_map.size(); }
294
299     bool empty()const { return time_arr_map.empty(); }
300
301
305     void clear() { time_arr_map.clear(); }
306
307
314     TimeArr& setDelayPrecision( long double precision );
315
316
321     static void setDebug( bool flag ) { debug = flag; }
322
323
328     double getMaxDelayValue()const { return( time_arr_map.rbegin()->first ); }
329
334     double getMinDelayValue()const { return( time_arr_map.begin()->first ); }
335
340     long double getDelayPrecision()const { return delay_precision; }
341
342
```

```
347     virtual bool isValid() const;
348
349
354     virtual bool isConvertedFromPressure()const { return( time_arr_map.size() == 1 && (
    time_arr_map.begin()->first == PDouble(TIMEARR_PRESSURE_CONVERSION_DELAY) ) ); }
355
356
362     TimeArr& operator=( const TimeArr& copy ) ;
363
364
371     friend bool operator==( const TimeArr& left, const TimeArr& right ) ;
372
379     friend bool operator!=( const TimeArr& left, const TimeArr& right );
380
381
388     friend const TimeArr operator+( const TimeArr& left, const TimeArr& right );
389
396     friend const TimeArr operator-( const TimeArr& left, const TimeArr& right );
397
398
405     friend const TimeArr operator+( const TimeArr& left, const double right );
406
413     friend const TimeArr operator-( const TimeArr& left, const double right );
414
421     friend const TimeArr operator/( const TimeArr& left, const double right );
422
429     friend const TimeArr operator*( const TimeArr& left, const double right );
430
431
438     friend const TimeArr operator+( const double left, const TimeArr& right );
439
446     friend const TimeArr operator-( const double left, const TimeArr& right );
447
454     friend const TimeArr operator/( const double left, const TimeArr& right );
455
462     friend const TimeArr operator*( const double left, const TimeArr& right );
463
464
471     friend TimeArr& operator+=( TimeArr& left, const TimeArr& right );
472
479     friend TimeArr& operator-=( TimeArr& left, const TimeArr& right );
480
481
488     friend TimeArr& operator+=( TimeArr& left, double right );
489
496     friend TimeArr& operator-=( TimeArr& left, double right );
497
504     friend TimeArr& operator/=( TimeArr& left, double right );
505
512     friend TimeArr& operator*=( TimeArr& left, double right );
513
514
521     friend std::ostream& operator«( std::ostream& os, const TimeArr& instance );
522
523
524     protected:
525
526
530     static bool debug;
531
532
536     long double delay_precision;
537
538
542     TimeArrMap time_arr_map;
543
544
545   };
546
547   //non-inline operator declarations
549   const TimeArr operator+( const TimeArr& left, const TimeArr& right );
550
551   const TimeArr operator-( const TimeArr& left, const TimeArr& right );
552
553
554   const TimeArr operator+( const TimeArr& left, const double right );
555
556   const TimeArr operator-( const TimeArr& left, const double right );
557
558   const TimeArr operator/( const TimeArr& left, const double right );
559
560   const TimeArr operator*( const TimeArr& left, const double right );
561
562
563   const TimeArr operator+( const double left, const TimeArr& right );
564
565   const TimeArr operator-( const double left, const TimeArr& right );
```

```
566
567   const TimeArr operator/( const double left, const TimeArr& right );
568
569   const TimeArr operator*( const double left, const TimeArr& right );
570
571
572   TimeArr& operator+=( TimeArr& left, const TimeArr& right );
573
574   TimeArr& operator-=( TimeArr& left, const TimeArr& right );
575
576
577   TimeArr& operator+=( TimeArr& left, double right );
578
579   TimeArr& operator-=( TimeArr& left, double right );
580
581   TimeArr& operator/=( TimeArr& left, double right );
582
583   TimeArr& operator*=( TimeArr& left, double right );
584
585
587   //inline functions
588   inline TimeArrMap& TimeArr::createNotValid() {
589     static TimeArrMap time_arr_map;
590     time_arr_map.clear();
591     time_arr_map[0.0] = Pressure::createNotValid();
592     return time_arr_map;
593   }
594
595
596   inline TimeArrMap& TimeArr::createImpulse() {
597     static TimeArrMap time_arr_map;
598     time_arr_map.clear();
599     time_arr_map[0.0] = std::complex<double> (1.0 , 0.0);
600     return time_arr_map;
601   }
602
603
604   inline TimeArr& TimeArr::insertValue( double delay, const Pressure& pressure ) {
605     assert( pressure.isValid() );
606     assert( delay >= 0.0 );
607
608     time_arr_map.insert( std::make_pair( delay, pressure ) );
609
610     return *this;
611   }
612
613
614   inline void TimeArr::sumValue( double delay, const Pressure& pressure ) {
615     assert( pressure.isValid() );
616     assert( delay >= 0.0 );
617
618     TimeArrIt it = time_arr_map.find( delay );
619     if ( it == time_arr_map.end() ) time_arr_map.insert( std::make_pair( delay, pressure ) );
620     else it->second += std::complex<double>( pressure );
621   }
622
623
624   inline std::ostream& operator«( std::ostream& os, const TimeArr& instance ) {
625     os « "size = " « instance.time_arr_map.size() « "; min time_arr = " «
    instance.time_arr_map.begin()->first
626                    « "; pressure db = " « Pressure::getTxLossDb(instance.begin()->second)
627                    « "; max time_arr = " « instance.time_arr_map.rbegin()->first
628                    « "; pressure db = " «
    Pressure::getTxLossDb(instance.time_arr_map.rbegin()->second);
629     return os;
630   }
631
632
633   inline bool operator==( const TimeArr& left, const TimeArr& right ) {
634     if ( &left == &right ) return true;
635     return( left.time_arr_map == right.time_arr_map );
636   }
637
638
639   inline bool operator!=( const TimeArr& left, const TimeArr& right ) {
640     if ( &left == &right ) return false;
641     return( left.time_arr_map != right.time_arr_map );
642   }
643
644 }
645
646
647 #endif /* TIME_ARRIVAL_DEFINITIONS_H */
648
649
650
```

14.140 woss/woss_def/time-definitions.cpp File Reference

Implementation of woss::Time class.

14.140.1 Detailed Description

Implementation of woss::Time class.

Author

Federico Guerra

Implementation of woss::Time class

14.141 woss/woss_def/time-definitions.h File Reference

Definitions and library for woss::Time, woss::SimTime, woss::TimeReference and woss::TimeReferenceTcl classes.

Classes

- class woss::TimeReference
  Class for simulation time reference purposes.
- class woss::Time
  a class for time date manipulation
- struct woss::SimTime
  Struct that stores start and end Time.

Functions

- double woss::operator- (const Time &left, const Time &right)
- bool woss::operator== (const Time &left, const Time &right)
- bool woss::operator!= (const Time &left, const Time &right)
- bool woss::operator> (const Time &left, const Time &right)
- bool woss::operator< (const Time &left, const Time &right)
- bool woss::operator<= (const Time &left, const Time &right)
- bool woss::operator>= (const Time &left, const Time &right)
- std::ostream & **woss::operator**<< (std::ostream &os, const Time &time)
- const Time woss::operator+ (const Time &left, const time_t right)
- const Time woss::operator- (const Time &left, const time_t right)
- Time & woss::operator+= (Time &left, time_t right)
- Time & woss::operator-= (Time &left, time_t right)

14.141.1 Detailed Description

Definitions and library for woss::Time, woss::SimTime, woss::TimeReference and woss::TimeReferenceTcl classes.

Author

Federico Guerra

Definitions and library for woss::Time, woss::SimTime, woss::TimeReference and woss::TimeReferenceTcl classes

14.141.2 Function Documentation

14.141.2.1 operator"!=() bool woss::operator!= (const Time & *left*, const Time & *right*) [inline]

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left != right*, false otherwise

14.141.2.2 operator+() const Time woss::operator+ (const Time & *left*, const time_t *right*) [inline]

Sum operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.141.2.3 operator+=() Time & woss::operator+= (Time & *left*, time_t *right*)

Compound assignment sum operator

Parameters

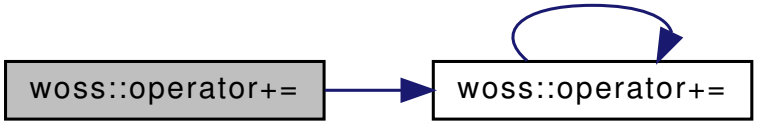
left	left operand reference
right	right operand const time_t representing seconds

Returns

left reference after the operation

References woss::operator+=(), woss::Time::raw_time, and woss::Time::timeinfo.

Here is the call graph for this function:



14.141.2.4 operator-() [1/2] double woss::operator- (const Time & *left*, const Time & *right*)

Subtraction operator

Parameters

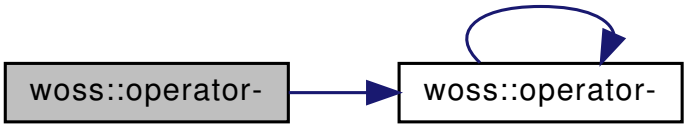
left	left operand const reference
right	right operand const reference

Returns

time difference in seconds

References woss::operator-(), and woss::Time::timeinfo.

Here is the call graph for this function:



14.141.2.5 operator-() [2/2] const Time woss::operator- (const Time & *left*, const time_t *right*) [inline]

Subtraction operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

a new const instance holding the operation result

14.141.2.6 operator-=() `Time & woss::operator-= ( Time & left, time_t right )`

Compound assignment subtraction operator

Parameters

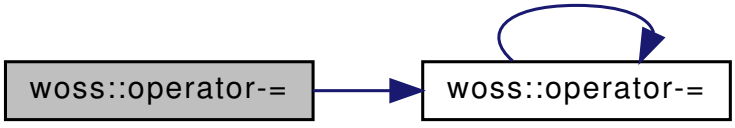
left	left operand reference
right	right operand const time_t representing seconds

Returns

left reference after the operation

References woss::operator-=(), woss::Time::raw_time, and woss::Time::timeinfo.

Here is the call graph for this function:



14.141.2.7 operator<() `bool woss::operator< ( const Time & left, const Time & right ) [inline]`

Less than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* < *right*, false otherwise

14.141.2.8 operator<=() `bool woss::operator<= (`

```
const Time & left,
const Time & right )  [inline]
```

Less than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* <= *right*, false otherwise

14.141.2.9 operator==() `bool woss::operator== (`

```
const Time & left,
const Time & right )  [inline]
```

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* == *right*, false otherwise

14.141.2.10 operator>() `bool woss::operator> (`

```
const Time & left,
const Time & right )  [inline]
```

Greater than operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* > *right*, false otherwise

14.141.2.11 operator>=() bool woss::operator>= (

```
const Time & left,
const Time & right )  [inline]
```

Greater than or equal to operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left* >= *right*, false otherwise

14.142 time-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_TIME_DEFINITIONS_H
41 #define WOSS_TIME_DEFINITIONS_H
42
43
44 #include <iostream>
45 #include <ctime>
46 #include <cassert>
47 #include <cmath>
48 #include <climits>
49
50
51 namespace woss {
52
53
54 #define TIME_NOT_SET_VALUE (LONG_MIN)
```

```
55
56
63   class TimeReference {
64
65
66     public:
67
68
69     virtual ~TimeReference() = 0;
70
75     virtual TimeReference* clone() = 0;
76
81     virtual double getTimeReference() const = 0;
82
83
84   };
85
86   inline TimeReference::~TimeReference() { }
87
95   class Time {
96
97
98     public:
99
100
104    Time();
105
110    Time( struct tm* time );
111
121    Time( int day, int month, int year, int hours = 0, int mins = 0, int seconds = 1 );
122
127    Time( const Time& copy );
128
133    ~Time() { }
134
135
141    Time& setMonth( int m ) { assert( m >= 1 && m <= 12 ); timeinfo.tm_mon = m - 1; raw_time =
     mktime(&timeinfo);
142                              return *this; }
143
149    Time& setDay( int d ) { assert( d >= 1 && d <= 31 ); timeinfo.tm_mday = d; raw_time =
     mktime(&timeinfo); return *this; }
150
156    Time& setYear( int y ) { assert( y >= 1900 ); timeinfo.tm_year = y - 1900; raw_time =
     mktime(&timeinfo); return *this; }
157
163    Time& setHours( int h ) { assert( h >= 0 && h <= 23 ); timeinfo.tm_hour = h; raw_time =
     mktime(&timeinfo); return *this; }
164
170    Time& setMinutes( int m ) { assert( m >= 0 && m <= 59 ); timeinfo.tm_min = m; raw_time =
     mktime(&timeinfo); return *this; }
171
177    Time& setSeconds( int s ) { assert( s >= 0 && s <= 59 ); timeinfo.tm_sec = s; raw_time =
     mktime(&timeinfo); return *this; }
178
179
184    static void setDebug(bool flag) { debug = flag; }
185
186
191    bool isValid()const { return ( raw_time != TIME_NOT_SET_VALUE ); }
192
193
198    int getMonth()const { return timeinfo.tm_mon; }
199
204    int getDay()const { return timeinfo.tm_mday; }
205
210    int getHours()const { return timeinfo.tm_hour; }
211
216    int getYear()const { return timeinfo.tm_year; }
217
222    int getMinutes()const { return timeinfo.tm_min; }
223
228    int getSeconds()const { return timeinfo.tm_sec; }
229
230
235    operator time_t() const;
236
242    Time& operator=( const Time& copy );
243
244
251    friend const Time operator+( const Time& left, const time_t right );
252
259    friend const Time operator-( const Time& left, const time_t right );
260
267    friend double operator-( const Time& left, const Time& right );
268
269
```

```
276     friend Time& operator+=( Time& left, time_t right );
277
284     friend Time& operator-=( Time& left, time_t right );
285
286
293     friend bool operator==( const Time& left, const Time& right );
294
301     friend bool operator!=( const Time& left, const Time& right );
302
303
310     friend bool operator>( const Time& left, const Time& right );
311
318     friend bool operator<( const Time& left, const Time& right );
319
326     friend bool operator<=( const Time& left, const Time& right );
327
334     friend bool operator>=( const Time& left, const Time& right );
335
336
343     friend ::std::ostream& operator«( ::std::ostream& os, const Time& instance );
344
345
346     protected:
347
348
352     mutable struct tm timeinfo;
353
357     time_t raw_time;
358
359
363     static bool debug;
364
365   };
366
367
373   struct SimTime {
374
375
376     SimTime( Time time1 = Time(), Time time2 = Time() ) :  start_time(time1), end_time(time2) { }
377
378
379     friend std::ostream& operator«( std::ostream& os, const SimTime& instance ) {
380       os « "start time = " « instance.start_time « "; end time = " « instance.end_time;
381       return os;
382     }
383
384
385     Time start_time;
386
387     Time end_time;
388
389
390   };
391
392
393   // non-inline operator declarations
394   double operator-( const Time& left, const Time& right );
395
396
397   //inline functions
399   inline Time::operator time_t()const {
400     return( mktime(&timeinfo) );
401   }
402
403
404   inline bool operator==( const Time& left, const Time& right ) {
405     if ( &left == &right ) return true;
406     return( mktime(const_cast<tm*>(&left.timeinfo)) == mktime(const_cast<tm*>(&right.timeinfo)) );
407   }
408
409
410   inline bool operator!=( const Time& left, const Time& right ) {
411     if ( &left == &right ) return false;
412     return( mktime(const_cast<tm*>(&left.timeinfo)) != mktime(const_cast<tm*>(&right.timeinfo)) );
413   }
414
415
416   inline bool operator>( const Time& left, const Time& right ) {
417     if ( &left == &right ) return false;
418     return( mktime(const_cast<tm*>(&left.timeinfo)) > mktime(const_cast<tm*>(&right.timeinfo)) );
419   }
420
421
422   inline bool operator<( const Time& left, const Time& right ) {
423     if ( &left == &right ) return false;
424     return( mktime(const_cast<tm*>(&left.timeinfo)) < mktime(const_cast<tm*>(&right.timeinfo)) );
425   }
```

```
426
427
428   inline bool operator<=( const Time& left, const Time& right ) {
429     return( mktime(const_cast<tm*>(&left.timeinfo)) <= mktime(const_cast<tm*>(&right.timeinfo)) );
430   }
431
432
433   inline bool operator>=( const Time& left, const Time& right ) {
434     return( mktime(const_cast<tm*>(&left.timeinfo)) >= mktime(const_cast<tm*>(&right.timeinfo)) );
435   }
436
437
438   inline std::ostream& operator«( std::ostream& os, const Time& time ) {
439     os « asctime(const_cast<tm*>(&time.timeinfo)) ;
440     return os;
441   }
442
443
444   inline const Time operator+( const Time& left, const time_t right ) {
445     time_t sum_time = mktime(const_cast<tm*>(&left.timeinfo)) + right;
446     return( Time( localtime( &sum_time ) ) );
447   }
448
449
450   inline const Time operator-( const Time& left, const time_t right ) {
451     time_t diff_time = mktime(const_cast<tm*>(&left.timeinfo)) - right;
452     return( Time( localtime( &diff_time ) ) );
453   }
454
455
456   Time& operator+=( Time& left, time_t right );
457
458
459   Time& operator-=( Time& left, time_t right );
460
461 }
462
463
464 #endif /* WOSS_TIME_DEFINITIONS_H */
465
466
```

14.143 woss/woss_def/transducer-definitions.cpp File Reference

Provides the implementation of the the woss::Transducer class.

14.143.1 Detailed Description

Provides the implementation of the the woss::Transducer class.

Author

Federico Guerra

Provides the implementation of the the woss::Transducer class

14.144 woss/woss_def/transducer-definitions.h File Reference

Provides the interface for the woss::Transducer class.

Classes

- class woss::Transducer

  Transducer class.

Functions

- bool woss::operator== (const Transducer &left, const Transducer &right)
- bool woss::operator!= (const Transducer &left, const Transducer &right)
- inline ::std::ostream & **woss::operator**<< (::std::ostream &os, const Transducer &instance)
- inline ::std::istream & **woss::operator**>> (::std::istream &is, Transducer &instance)

14.144.1 Detailed Description

Provides the interface for the woss::Transducer class.

Author

Federico Guerra

Provides the interface for the woss::Transducer class

14.144.2 Function Documentation

14.144.2.1 operator"!=() `bool woss::operator!= ( const Transducer & left, const Transducer & right ) [inline]`

Inequality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.144.2.2 operator==() `bool woss::operator== ( const Transducer & left, const Transducer & right ) [inline]`

Equality operator

Parameters

left	left operand const reference
right	right operand const reference

Returns

true if *left == right*, false otherwise

14.145 transducer-definitions.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_TRANSDUCER_DEFINITIONS_H
41 #define WOSS_TRANSDUCER_DEFINITIONS_H
42
43
44 #include <iostream>
45 #include <complex>
46 #include <map>
47 #include <cmath>
48 #include <cassert>
49 #include "custom-precision-double.h"
50
51
52 namespace woss {
53
54
55   class CoordZ;
56
57
58 #define BEAM_PATTERN_CUSTOM_BEAM_PRECISION (1.0)
59 #define CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION (1.0)
60 #define TVR_CUSTOM_FREQUENCY_PRECISION (1.0)
61 #define OCV_CUSTOM_FREQUENCY_PRECISION (1.0)
62 #define TRANSDUCER_NOT_SET (-1000)
63
64
70   class Transducer {
71
72
73     protected:
74
75
79     typedef ::std::map< PDouble, double > BeamPowerMap;
80     typedef BeamPowerMap::iterator BPMIter;
81     typedef BeamPowerMap::reverse_iterator BPMRIter;
82     typedef BeamPowerMap::const_iterator BPMCIter;
83     typedef BeamPowerMap::const_reverse_iterator BPMCRIter;
84
85
89     typedef ::std::map< PDouble, double > ConductanceMap;
90     typedef ConductanceMap::iterator CMIter;
91     typedef ConductanceMap::reverse_iterator CMRIter;
92     typedef ConductanceMap::const_iterator CMCIter;
93     typedef ConductanceMap::const_reverse_iterator CMCRIter;
94
95
99     typedef ::std::map< PDouble, double > TVRMap;
100     typedef TVRMap::iterator TVRMIter;
```

```
101     typedef TVRMap::reverse_iterator TVRMRIter;
102     typedef TVRMap::const_iterator TVRMCIter;
103     typedef TVRMap::const_reverse_iterator TVRMCRIter;
104
105
109     typedef ::std::map< PDouble, double > OCVMap;
110     typedef OCVMap::iterator OCVMIter;
111     typedef OCVMap::reverse_iterator OCVMRIter;
112     typedef OCVMap::const_iterator OCVMCIter;
113     typedef OCVMap::const_reverse_iterator OCVMCRIter;
114
115
116     public:
117
118
126     Transducer( long double beam_precision = BEAM_PATTERN_CUSTOM_BEAM_PRECISION, long double
    conduct_precison = CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION,
127                 long double tvr_precision = TVR_CUSTOM_FREQUENCY_PRECISION, long double ocv_precision =
    OCV_CUSTOM_FREQUENCY_PRECISION );
128
133     Transducer( const Transducer& copy );
134
146     Transducer( BeamPowerMap& beam_map, ConductanceMap& conductance_map, TVRMap& tvr_map, OCVMap&
    ocv_map );
147
148
149     virtual ~Transducer() { }
150
151
164     virtual Transducer* create( long double beam_precision = BEAM_PATTERN_CUSTOM_BEAM_PRECISION, long
    double conduct_precison = CONDUCTANCE_CUSTOM_FREQUENCY_PRECISION,
165                                 long double tvr_precision = TVR_CUSTOM_FREQUENCY_PRECISION, long double
    ocv_precision = OCV_CUSTOM_FREQUENCY_PRECISION ) const;
166
167
180     virtual Transducer* create( BeamPowerMap& beam_map, ConductanceMap& conductance_map, TVRMap&
    tvr_map, OCVMap& ocv_map ) const;
181
187     virtual Transducer* create( const Transducer& copy ) const;
188
193     virtual Transducer* clone() const;
194
195
200     virtual bool isValid() const;
201
202
208     Transducer& beampattern_rotate( double angle );
209
215     Transducer& beampattern_sum( double value );
216
222     Transducer& beampattern_multiply( double value );
223
224
231     virtual double getSPL( double frequency, double power ) const;
232
238     double getMaxSPL( double frequency ) const;
239
246     virtual double getPowerFromSPL( double frequency, double spl ) const;
247
254     bool beampattern_insert( double angle, double power );
255
256
263     Transducer& beampattern_replace( double angle, double power );
264
265
271     BPMCIter beampattern_find( double angle ) const;
272
273
279     Transducer& beampattern_erase( double angle );
280
281
286     int beampattern_size() const;
287
292     bool beampattern_empty() const;
293
294
299     Transducer& beampattern_clear();
300
301
306     BPMCIter beampattern_begin() const;
307
312     BPMCIter beampattern_end() const;
313
318     BPMCRIter beampattern_rbegin() const;
319
324     BPMCRIter beampattern_rend() const;
325
```

```
331     BPMCIter beampattern_lower_bound( double angle ) const;
332
338     BPMCIter beampattern_upper_bound( double angle ) const;
339
340
347     bool conductance_insert( double frequency, double conductance );
348
355     bool conductance_insert( double frequency, const ::std::complex<double>& impedance );
356
357
364     Transducer& conductance_replace( double frequency, double conductance );
365
372     Transducer& conductance_replace( double frequency, const ::std::complex<double>& impedance );
373
374
380     CMCIter conductance_find( double frequency ) const;
381
382
388     Transducer& conductance_erase( double frequency );
389
390
395     int conductance_size() const;
396
401     bool conductance_empty() const;
402
403
408     Transducer& conductance_clear();
409
410
415     CMCIter conductance_begin() const;
416
421     CMCIter conductance_end() const;
422
427     CMCRIter conductance_rbegin() const;
428
433     CMCRIter conductance_rend() const;
434
440     CMCIter conductance_lower_bound( double frequency ) const;
441
447     CMCIter conductance_upper_bound( double frequency ) const;
448
449
456     bool tvr_insert( double frequency, double tvr );
457
464     Transducer& tvr_replace( double frequency, double tvr );
465
466
472     TVRMCIter tvr_find( double frequency ) const;
473
474
480     Transducer& tvr_erase( double frequency );
481
482
487     int tvr_size() const;
488
493     bool tvr_empty() const;
494
495
500     Transducer& tvr_clear();
501
502
507     TVRMCIter tvr_begin() const;
508
513     TVRMCIter tvr_end() const;
514
519     TVRMCRIter tvr_rbegin() const;
520
525     TVRMCRIter tvr_rend() const;
526
532     TVRMCIter tvr_lower_bound( double frequency ) const;
533
539     TVRMCIter tvr_upper_bound( double frequency ) const;
540
541
548     bool ocv_insert( double frequency, double ocv );
549
556     Transducer& ocv_replace( double frequency, double ocv );
557
558
564     OCVMCIter ocv_find( double frequency ) const;
565
566
572     Transducer& ocv_erase( double frequency );
573
574
579     int ocv_size() const;
580
```

```
585     bool ocv_empty() const;
586
587
592     Transducer& ocv_clear();
593
594
599     OCVMCIter ocv_begin() const;
600
605     OCVMCIter ocv_end() const;
606
611     OCVMCRIter ocv_rbegin() const;
612
617     OCVMCRIter ocv_rend() const;
618
624     OCVMCIter ocv_lower_bound( double frequency ) const;
625
631     OCVMCIter ocv_upper_bound( double frequency ) const;
632
633
638     Transducer& clearAll();
639
640
646     virtual bool import( ::std::istream& stream_in );
647
653     virtual bool importBinary( ::std::fstream& stream_in );
654
655
664     bool writeVertBeamPattern( ::std::ostream& stream_out, const CoordZ& tx, const CoordZ& rx, double
        init_bearing, double vert_rot = 0, double horiz_rot = 0, double mult_costant = 1, double add_costant =
        0 ) const;
665
666
672     bool writeSPL( ::std::ostream& stream_out, double frequency_step, double power ) const;
673
674
680     virtual bool write( ::std::ostream& stream_out ) const;
681
687     virtual bool writeBinary( ::std::fstream& file_out ) const;
688
689
690     Transducer& setMaxPower( double power );
691
692     Transducer& setDutyCycle( double cycle );
693
694     Transducer& setResonanceFrequency( double frequency );
695
696     Transducer& setBandwith3dB( double frequency );
697
698     Transducer& setTypeName( const ::std::string& name );
699
700
706     virtual Transducer& setBeamPrecision( long double prec );
707
713     virtual Transducer& setTVRPrecision( long double prec );
714
720     virtual Transducer& setOCVPrecision( long double prec );
721
727     virtual Transducer& setConductancePrecision( long double prec );
728
729
730     bool hasToroidalSymmetry() const;
731
732     bool hasConicalSymmetry() const;
733
734
739     double getMaxPower() const;
740
745     double getDutyCycle() const;
746
751     double getResonanceFrequency() const;
752
757     double getBandwith3dB() const;
758
763     ::std::string getTypeName() const;
764
765
770     long double getBeamPrecision() const;
771
776     long double getTVRPrecision() const;
777
782     long double getOCVPrecision() const;
783
788     long double getConductancePrecision() const;
789
790
796     Transducer& operator=( const Transducer& x ) ;
797
```

```
798
805     friend bool operator==( const Transducer& left, const Transducer& right );
806
813     friend bool operator!=( const Transducer& left, const Transducer& right );
814
815
822     friend ::std::ostream& operator«( ::std::ostream& os, const Transducer& instance );
823
830     friend ::std::ostream& operator»( ::std::istream& is, const Transducer& instance );
831
832
837     static void setDebug( bool flag ) { debug = flag; }
838
839
840     protected:
841
842
843     static const ::std::string conical_string;
844
845     static const ::std::string toroidal_string;
846
847
851     static bool debug;
852
853
857     bool has_conical_symmetry;
858
859
863     double resonance_frequency;
864
868     double bandwith_3db;
869
873     double max_power;
874
878     double duty_cycle;
879
880
884     long double beam_precision;
885
889     long double conductance_precision;
890
894     long double tvr_precision;
895
899     long double ocv_precision;
900
904     ::std::string type_name;
905
906
910     BeamPowerMap beam_power_map;
911
915     ConductanceMap conductance_map;
916
920     TVRMap tvr_map;
921
925     OCVMap ocv_map;
926
927
933     virtual double normalizeAngle( double angle ) const;
934
935
943     virtual double getValue( double frequency, const ::std::map< PDouble, double >& map, long double
      precision, bool use_linear = false, double costant = 20.0 ) const;
944
945
950     virtual void beampattern_sum( double value, BeamPowerMap& map );
951
956     virtual void beampattern_multiply( double value, BeamPowerMap& map );
957
962     virtual void beampattern_rotate( double angle, BeamPowerMap& map );
963
964
973     virtual bool import( ::std::istream& stream_in, ::std::map< PDouble, double >& map, long double
      precision, bool is_angle = false );
974
983     virtual bool importBinary( ::std::fstream& file_in, ::std::map< PDouble, double >& map, long double
      precision, bool is_angle = false );
984
985
992     virtual bool write( ::std::ostream& stream_out, const ::std::map< PDouble, double >& map ) const;
993
1000     virtual bool writeBinary( ::std::fstream& file_out, const ::std::map< PDouble, double >& map )
      const;
1001
1002
1003     virtual const ::std::string& getSymmetryString() const;
1004
1005
```

```
1006  };
1007
1008
1010  //inline functions
1011  inline Transducer& Transducer::setMaxPower( double power ) {
1012    max_power = power;
1013    return *this;
1014  }
1015
1016
1017  inline Transducer& Transducer::setDutyCycle( double cycle ) {
1018    duty_cycle = cycle;
1019    return *this;
1020  }
1021
1022
1023  inline Transducer& Transducer::setResonanceFrequency( double frequency ) {
1024    resonance_frequency = frequency;
1025    return *this;
1026  }
1027
1028
1029  inline Transducer& Transducer::setBandwith3dB( double bw ) {
1030    bandwith_3db = bw;
1031    return *this;
1032  }
1033
1034
1035  inline Transducer& Transducer::setTypeName( const ::std::string& name ) {
1036    type_name = name;
1037    return *this;
1038  }
1039
1040
1041  inline bool Transducer::hasToroidalSymmetry()const {
1042    return !has_conical_symmetry;
1043  }
1044
1045
1046  inline bool Transducer::hasConicalSymmetry()const {
1047    return has_conical_symmetry;
1048  }
1049
1050
1051  inline double Transducer::getMaxPower()const {
1052    return max_power;
1053  }
1054
1055
1056  inline double Transducer::getDutyCycle()const {
1057    return duty_cycle;
1058  }
1059
1060
1061  inline double Transducer::getResonanceFrequency()const {
1062    return resonance_frequency;
1063  }
1064
1065
1066  inline double Transducer::getBandwith3dB()const {
1067    return bandwith_3db;
1068  }
1069
1070
1071  inline ::std::string Transducer::getTypeName()const {
1072    return type_name;
1073  }
1074
1075
1076  inline double Transducer::getMaxSPL( double frequency )const {
1077    return getSPL( frequency, max_power );
1078  }
1079
1080
1081  inline Transducer& Transducer::beampattern_rotate( double angle ) {
1082    beampattern_rotate( angle, beam_power_map );
1083    return *this;
1084  }
1085
1086
1087  inline Transducer& Transducer::beampattern_sum( double value ) {
1088    beampattern_sum( value, beam_power_map );
1089    return *this;
1090  }
1091
1092
1093  inline Transducer& Transducer::beampattern_multiply( double value ) {
```

```
1094     beampattern_multiply( value, beam_power_map );
1095     return *this;
1096   }
1097
1098
1099   inline bool Transducer::beampattern_insert( double angle, double power ) {
1100     return beam_power_map.insert( ::std::make_pair( PDouble( normalizeAngle(angle), beam_precision),
      power ) ).second;
1101   }
1102
1103
1104   inline Transducer& Transducer::beampattern_replace( double angle, double power ) {
1105     normalizeAngle(angle);
1106     beam_power_map[PDouble(angle, beam_precision)] = power;
1107     return *this;
1108   }
1109
1110
1111   inline Transducer::BPMCIter Transducer::beampattern_find( double angle )const {
1112     return beam_power_map.find(angle);
1113   }
1114
1115
1116   inline Transducer& Transducer::beampattern_erase( double angle ) {
1117     beam_power_map.erase(angle);
1118     return *this;
1119   }
1120
1121   inline int Transducer::beampattern_size()const {
1122     return beam_power_map.size();
1123   }
1124
1125
1126   inline bool Transducer::beampattern_empty()const {
1127     return beam_power_map.empty();
1128   }
1129
1130
1131   inline Transducer& Transducer::beampattern_clear() {
1132     beam_power_map.clear();
1133     return *this;
1134   }
1135
1136
1137   inline Transducer::BPMCIter Transducer::beampattern_begin()const {
1138     return beam_power_map.begin();
1139   }
1140
1141
1142   inline Transducer::BPMCIter Transducer::beampattern_end()const {
1143     return beam_power_map.end();
1144   }
1145
1146
1147   inline Transducer::BPMCRIter Transducer::beampattern_rbegin()const  {
1148     return beam_power_map.rbegin();
1149   }
1150
1151
1152   inline Transducer::BPMCRIter Transducer::beampattern_rend()const  {
1153     return beam_power_map.rend();
1154   }
1155
1156
1157   inline Transducer::BPMCIter Transducer::beampattern_lower_bound( double angle )const  {
1158     return beam_power_map.lower_bound(PDouble(angle, beam_precision));
1159   }
1160
1161
1162   inline Transducer::BPMCIter Transducer::beampattern_upper_bound( double angle )const  {
1163     return beam_power_map.upper_bound(PDouble(angle, beam_precision));
1164   }
1165
1166
1167
1168   inline bool Transducer::conductance_insert( double frequency, double conductance ) {
1169     return conductance_map.insert( ::std::make_pair( PDouble( frequency, conductance_precision ),
      conductance ) ).second;
1170   }
1171
1172
1173   inline bool Transducer::conductance_insert( double frequency, const ::std::complex< double >&
      impedance ) {
1174     return conductance_map.insert( ::std::make_pair( PDouble( frequency, conductance_precision ),
      impedance.real()
1175                                                      / ( ::std::pow( ::std::abs(impedance),
      2.0 ) ) ) ).second;
```

```
1176   }
1177
1178
1179   inline Transducer& Transducer::conductance_replace( double frequency, double conductance ) {
1180     conductance_map[PDouble(frequency, conductance_precision)] = conductance;
1181     return *this;
1182   }
1183
1184
1185   inline Transducer& Transducer::conductance_replace( double frequency, const ::std::complex< double >&
       impedance ) {
1186     conductance_map[PDouble(frequency, conductance_precision)] = impedance.real()
1187                                                                  / ( ::std::pow( ::std::abs(impedance),
       2.0 ) ) ;
1188     return *this;
1189   }
1190
1191
1192   inline Transducer::CMCIter Transducer::conductance_find( double frequency )const {
1193     return conductance_map.find(frequency);
1194   }
1195
1196
1197   inline Transducer& Transducer::conductance_erase( double frequency ) {
1198     conductance_map.erase(frequency);
1199     return *this;
1200   }
1201
1202   inline int Transducer::conductance_size()const {
1203     return conductance_map.size();
1204   }
1205
1206
1207   inline bool Transducer::conductance_empty()const {
1208     return conductance_map.empty();
1209   }
1210
1211
1212   inline Transducer& Transducer::conductance_clear() {
1213     conductance_map.clear();
1214     return *this;
1215   }
1216
1217
1218   inline Transducer::CMCIter Transducer::conductance_begin()const {
1219     return conductance_map.begin();
1220   }
1221
1222
1223   inline Transducer::CMCIter Transducer::conductance_end()const {
1224     return conductance_map.end();
1225   }
1226
1227
1228   inline Transducer::CMCRIter Transducer::conductance_rbegin()const  {
1229     return conductance_map.rbegin();
1230   }
1231
1232
1233   inline Transducer::CMCRIter Transducer::conductance_rend()const  {
1234     return conductance_map.rend();
1235   }
1236
1237
1238   inline Transducer::CMCIter Transducer::conductance_lower_bound( double frequency )const  {
1239     return conductance_map.lower_bound(PDouble(frequency, conductance_precision));
1240   }
1241
1242
1243   inline Transducer::CMCIter Transducer::conductance_upper_bound( double frequency )const  {
1244     return conductance_map.upper_bound(PDouble(frequency, conductance_precision));
1245   }
1246
1247
1248
1249
1250   inline bool Transducer::tvr_insert( double frequency, double tvr ) {
1251     return tvr_map.insert( ::std::make_pair( PDouble( frequency, tvr_precision ), tvr ) ).second;
1252   }
1253
1254
1255   inline Transducer& Transducer::tvr_replace( double frequency, double tvr ) {
1256     tvr_map[PDouble(frequency, tvr_precision)] = tvr;
1257     return *this;
1258   }
1259
1260
```

```
1261  inline Transducer::CMCIter Transducer::tvr_find( double frequency )const {
1262    return tvr_map.find(frequency);
1263  }
1264
1265
1266  inline Transducer& Transducer::tvr_erase( double frequency ) {
1267    tvr_map.erase(frequency);
1268    return *this;
1269  }
1270
1271  inline int Transducer::tvr_size()const {
1272    return tvr_map.size();
1273  }
1274
1275
1276  inline bool Transducer::tvr_empty()const {
1277    return tvr_map.empty();
1278  }
1279
1280
1281  inline Transducer& Transducer::tvr_clear() {
1282    tvr_map.clear();
1283    return *this;
1284  }
1285
1286
1287  inline Transducer::CMCIter Transducer::tvr_begin()const {
1288    return tvr_map.begin();
1289  }
1290
1291
1292  inline Transducer::CMCIter Transducer::tvr_end()const {
1293    return tvr_map.end();
1294  }
1295
1296
1297  inline Transducer::CMCRIter Transducer::tvr_rbegin()const  {
1298    return tvr_map.rbegin();
1299  }
1300
1301
1302  inline Transducer::CMCRIter Transducer::tvr_rend()const  {
1303    return tvr_map.rend();
1304  }
1305
1306
1307  inline Transducer::CMCIter Transducer::tvr_lower_bound( double frequency )const  {
1308    return tvr_map.lower_bound(PDouble(frequency, tvr_precision));
1309  }
1310
1311
1312  inline Transducer::CMCIter Transducer::tvr_upper_bound( double frequency )const  {
1313    return tvr_map.upper_bound(PDouble(frequency, tvr_precision));
1314  }
1315
1316
1317
1318  inline bool Transducer::ocv_insert( double frequency, double ocv ) {
1319    return ocv_map.insert( ::std::make_pair( PDouble( frequency, ocv_precision ), ocv ) ).second;
1320  }
1321
1322
1323  inline Transducer& Transducer::ocv_replace( double frequency, double ocv ) {
1324    ocv_map[PDouble(frequency, ocv_precision)] = ocv;
1325    return *this;
1326  }
1327
1328
1329  inline Transducer::CMCIter Transducer::ocv_find( double frequency )const {
1330    return ocv_map.find(frequency);
1331  }
1332
1333
1334  inline Transducer& Transducer::ocv_erase( double frequency ) {
1335    ocv_map.erase(frequency);
1336    return *this;
1337  }
1338
1339  inline int Transducer::ocv_size()const {
1340    return ocv_map.size();
1341  }
1342
1343
1344  inline bool Transducer::ocv_empty()const {
1345    return ocv_map.empty();
1346  }
1347
```

```
1348
1349   inline Transducer& Transducer::ocv_clear() {
1350     ocv_map.clear();
1351     return *this;
1352   }
1353
1354
1355   inline Transducer::CMCIter Transducer::ocv_begin()const {
1356     return ocv_map.begin();
1357   }
1358
1359
1360   inline Transducer::CMCIter Transducer::ocv_end()const {
1361     return ocv_map.end();
1362   }
1363
1364
1365   inline Transducer::CMCRIter Transducer::ocv_rbegin()const  {
1366     return ocv_map.rbegin();
1367   }
1368
1369
1370   inline Transducer::CMCRIter Transducer::ocv_rend()const  {
1371     return ocv_map.rend();
1372   }
1373
1374
1375   inline Transducer::CMCIter Transducer::ocv_lower_bound( double frequency )const  {
1376     return ocv_map.lower_bound(PDouble(frequency, ocv_precision));
1377   }
1378
1379
1380   inline Transducer::CMCIter Transducer::ocv_upper_bound( double frequency )const  {
1381     return ocv_map.upper_bound(PDouble(frequency, ocv_precision));
1382   }
1383
1384
1385   inline Transducer& Transducer::clearAll() {
1386     beam_power_map.clear();
1387     ocv_map.clear();
1388     tvr_map.clear();
1389     conductance_map.clear();
1390     return *this;
1391   }
1392
1393
1394   inline long double Transducer::getBeamPrecision()const {
1395     return beam_precision;
1396   }
1397
1398
1399   inline long double Transducer::getConductancePrecision()const {
1400     return conductance_precision;
1401   }
1402
1403
1404   inline long double Transducer::getTVRPrecision()const {
1405     return tvr_precision;
1406   }
1407
1408
1409   inline long double Transducer::getOCVPrecision()const {
1410     return ocv_precision;
1411   }
1412
1413
1414   inline bool operator==( const Transducer& left, const Transducer& right ) {
1415     if ( &left == &right ) return true;
1416     return( left.beam_power_map == right.beam_power_map && left.conductance_map ==
     right.conductance_map
1417          && left.tvr_map == right.tvr_map && left.ocv_map == right.ocv_map );
1418   }
1419
1420
1421   inline bool operator!=( const Transducer& left, const Transducer& right ) {
1422     if ( &left == &right ) return false;
1423     if ( &left == &right ) return true;
1424     return( left.beam_power_map != right.beam_power_map || left.conductance_map !=
     right.conductance_map
1425          || left.tvr_map != right.tvr_map || left.ocv_map != right.ocv_map );
1426   }
1427
1428
1429   inline ::std::ostream& operator«( ::std::ostream& os, const Transducer& instance ) {
1430     if ( !instance.write( os ) ) os.setstate( ::std::ios_base::failbit );
1431     return os;
1432   }
```

```
1433
1434
1435   inline ::std::istream& operator»( ::std::istream& is, Transducer& instance ) {
1436     if ( !instance.import( is ) ) is.setstate( ::std::ios_base::failbit );
1437     return is;
1438   }
1439
1440
1441 }
1442
1443
1444 #endif // WOSS_TRANSDUCER_DEFINITIONS_H
1445
```

14.146 woss/woss_def/transducer-handler.cpp File Reference

Provides the implementation of the the woss::TransducerHandler class.

14.146.1 Detailed Description

Provides the implementation of the the woss::TransducerHandler class.

Author

Federico Guerra

Provides the implementation of the the woss::TransducerHandler class

14.147 woss/woss_def/transducer-handler.h File Reference

Provides the interface for the woss::transducer::TransducerHandler class.

Classes

- class woss::TransducerHandler

  Transducer creator and handler class.

14.147.1 Detailed Description

Provides the interface for the woss::transducer::TransducerHandler class.

Author

Federico Guerra

Provides the interface for the woss::trasducer::TransducerHandler class

14.148 transducer-handler.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef WOSS_TRANSDUCER_HANDLER_DEFINITIONS_H
41 #define WOSS_TRANSDUCER_HANDLER_DEFINITIONS_H
42
43
44 // #include <cmath>
45 // #include <cassert>
46 // #include <iostream>
47 #include <string>
48 #include <map>
49
50
51 namespace woss {
52
53
54   class Transducer;
55
56
63   class TransducerHandler {
64
65
66     protected:
67
68
72     typedef ::std::map< ::std::string, Transducer* > TransducerMap;
73     typedef TransducerMap::iterator TMIter;
74     typedef TransducerMap::reverse_iterator TMRIter;
75     typedef TransducerMap::const_iterator TMCIter;
76     typedef TransducerMap::const_reverse_iterator TMCRIter;
77
78
79     public:
80
81
85     TransducerHandler();
86
91     TransducerHandler( const TransducerHandler& copy );
92
97     TransducerHandler( TransducerMap& transduc_map );
98
99
100     virtual ~TransducerHandler();
101
102
110     bool insertValue( const ::std::string& name, Transducer* const transducer );
111
112
119     TransducerHandler& replaceValue( const ::std::string& name, Transducer* const transducer );
120
121
127     const Transducer* const getValue( const ::std::string& name ) const;
128
129
135     TransducerHandler& eraseValue( const ::std::string& name );
136
```

```
137
142     int size() const;
143
148     bool empty() const;
149
150
155     TransducerHandler& clear();
156
157
162     TMCIter begin() const;
163
168     TMCIter end() const;
169
174     TMCRIter rbegin() const;
175
180     TMCRIter rend() const;
181
188     virtual bool importValueAscii( const ::std::string& type_name, const ::std::string& file_name );
189
196     virtual bool importValueBinary( const ::std::string& type_name, const ::std::string& file_name );
197
198
205     virtual bool writeValueAscii( const ::std::string& type_name, const ::std::string& file_name );
206
213     virtual bool writeValueBinary( const ::std::string& type_name, const ::std::string& file_name );
214
215
221     TransducerHandler& operator=( const TransducerHandler& x ) ;
222
223
228     TransducerHandler& setDebug( bool flag );
229
230                 bool getDebug() { return debug; }
231
232
233     protected:
234
235
239     static const ::std::string TRANSDUCER_NOT_VALID;
240
241
245     bool debug;
246
247
251     TransducerMap transducer_map;
252
253
254   };
255
256
258
259   inline int TransducerHandler::size()const {
260     return transducer_map.size();
261   }
262
263
264   inline bool TransducerHandler::empty()const {
265     return transducer_map.empty();
266   }
267
268
269   inline TransducerHandler::TMCIter TransducerHandler::begin()const {
270     return transducer_map.begin();
271   }
272
273
274   inline TransducerHandler::TMCIter TransducerHandler::end()const {
275     return transducer_map.end();
276   }
277
278
279   inline TransducerHandler::TMCRIter TransducerHandler::rbegin()const {
280     return transducer_map.rbegin();
281   }
282
283   inline TransducerHandler::TMCRIter TransducerHandler::rend()const {
284     return transducer_map.rend();
285   }
286
287
288   inline TransducerHandler& TransducerHandler::setDebug( bool flag ) {
289     debug = flag;
290     return *this;
291   }
292
293
294 }
```

```
295
296
297 #endif // WOSS_TRANSDUCER_HANDLER_DEFINITIONS_H
298
```

14.149 woss_phy/uw-woss-bpsk.cpp File Reference

Provides the implementation of WossMPhyBpsk class.

Classes

- class **UwMPhyBpskTransducerClass**
- class **WossMPhyBpskClass**

14.149.1 Detailed Description

Provides the implementation of WossMPhyBpsk class.

Author

Federico Guerra

Provides the implementation of WossMPhyBpsk class

14.150 woss_phy/uw-woss-bpsk.h File Reference

Provides the interface for WossMPhyBpsk class.

Classes

- class UwMPhyBpskTransducer

  BPSK modulation class with woss::Transducer tx power control.
- class WossMPhyBpsk

  BPSK modulation class with woss::Transducer power control.

14.150.1 Detailed Description

Provides the interface for WossMPhyBpsk class.

Author

Federico Guerra

Provides the interface for WossMPhyBpsk class

14.151 uw-woss-bpsk.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
30 #ifndef UNDERWATER_WOSS_BPSK_H
31 #define UNDERWATER_WOSS_BPSK_H
32
33
44 #include <underwater-bpsk.h>
45
46
47 #define UWWOSSMPHYBPSK_MODNAME "UWWOSSBPSK"
48
49
50 namespace woss {
51
52   class Transducer;
53
54 }
55
56
62 class UwMPhyBpskTransducer :    public UnderwaterMPhyBpsk {
63
64
65   public:
66
67
68   UwMPhyBpskTransducer();
69
70
71   virtual ~UwMPhyBpskTransducer() { }
72
73
74   virtual int command( int argc, const char*const* argv );
75
76
77   protected:
78
79
80   virtual double consumedEnergyTx( double Ptx, double duration );
81
82
83   virtual const woss::Transducer* const getTransducer( double frequency ) const;
84
85
86   typedef ::std::map< ::std::pair< double, double >, const woss::Transducer* > FreqTransducerMap;
87   typedef FreqTransducerMap::iterator FTMIter;
88   typedef FreqTransducerMap::const_iterator FTMCIter;
89   typedef FreqTransducerMap::reverse_iterator FTMRIter;
90   typedef FreqTransducerMap::const_reverse_iterator FTMRCIter;
91
92
93   FreqTransducerMap freq_transd_map;
94
95
96 };
97
98
```

```
104 class WossMPhyBpsk :    public UwMPhyBpskTransducer {
105
106
107   public:
108
109
110   WossMPhyBpsk();
111
112
113   virtual ~WossMPhyBpsk() { }
114
115
116   protected:
117
118
126   virtual double getTxPower( Packet* p );
127
128
129 };
130
131
132 #endif /* UNDERWATER_WOSS_BPSK_H */
133
134
135
```

14.152 woss_phy/uw-woss-cbr.cpp File Reference

Provides the implementation of WossCbrModule class.

Classes

- class **WossCbrModuleClass**

14.152.1 Detailed Description

Provides the implementation of WossCbrModule class.

Author

Federico Guerra

Provides the implementation of WossCbrModule class

14.153 woss_phy/uw-woss-cbr.h File Reference

Provides the interface for WossCbrModule class.

Classes

- class WossCbrModule

14.153.1 Detailed Description

Provides the interface for WossCbrModule class.

Author

Federico Guerra

Provides the interface for WossCbrModule class

14.154 uw-woss-cbr.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_CBR_H
41 #define UNDERWATER_WOSS_CBR_H
42
43
44 #include <cbr-module.h>
45
46
47 class WossCbrModule :    public CbrModule {
48
49
50   public:
51
52
53   WossCbrModule();
54
55   virtual ~WossCbrModule() { }
56
57
58   virtual void recv(Packet*);
59
60
61   virtual int command(int argc, const char*const* argv);
62
63
64   protected:
65
66
67   double first_time_rx;
68
69   double last_time_rx;
70
71
72   void updateFirstTimeRx(double time);
73
74   void updateLastTimeRx(double time);
```

```
75
76
77   double getFirstTimeRx(){ return first_time_rx; }
78
79   double getLastTimeRx() { return last_time_rx; }
80
81
82 };
83
84
85 #endif /* UNDERWATER_WOSS_CBR_H */
86
87
88
89
```

14.155 woss_phy/uw-woss-channel-estimator.cpp File Reference

Provides the interface for ChannelEstimator class.

Classes

- class **ChannelEstimatorClass**
- class **ChEstimatorPlugInClass**

14.155.1 Detailed Description

Provides the interface for ChannelEstimator class.

Author

Federico Guerra

Provides the interface for ChannelEstimator class

14.156 woss_phy/uw-woss-channel-estimator.h File Reference

Provides the interfaces for ChannelEstimator and ChEstimatorPlugIn classes.

Classes

- class ChannelEstimator

  Class for channel estimation and averaging.
- class ChEstimatorPlugIn

  Service class for attaching a ChannelEstimator to the node bus.

14.156.1 Detailed Description

Provides the interfaces for ChannelEstimator and ChEstimatorPlugIn classes.

Author

Federico Guerra

Provides the interfaces for ChannelEstimator and ChEstimatorPlugIn classes

14.157 uw-woss-channel-estimator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef CHANNEL_ESTIMATOR_H
41 #define CHANNEL_ESTIMATOR_H
42
43
44 #include <plugin.h>
45 #include <coordinates-definitions.h>
46 #include <map>
47 #include "uw-woss-position.h"
48
49
50 namespace woss {
51   class TimeArr;
52 }
53
54
60 class ChannelEstimator :  public TclObject {
61
62
63   public:
64
65
66   typedef ::std::map< woss::CoordZ, woss::TimeArr*, woss::CoordComparator< ChannelEstimator, woss::CoordZ
      > > RxMap;
67   typedef RxMap::iterator RxMIter;
68   typedef RxMap::const_iterator RxMCIter;
69   typedef RxMap::reverse_iterator RxMRIter;
70
71   typedef ::std::map< woss::CoordZ, RxMap, woss::CoordComparator< ChannelEstimator, woss::CoordZ > >
      ChannelMap;
72   typedef ChannelMap::iterator ChMapIter;
73   typedef ChannelMap::reverse_iterator ChMapRIter;
74
75   typedef std::map< int, WossPosition* > MacToPosMap;
76   typedef MacToPosMap::iterator MacMapIter;
77   typedef MacToPosMap::reverse_iterator MacMapRIter;
78
79 //   typedef std::map< std::pair< woss::CoordZ, woss::CoordZ > , woss::TimeArr* > ChannelMap;
80 //   typedef ChannelMap::iterator ChMapIter;
81 //   typedef ChannelMap::reverse_iterator ChMapRIter;
82
83
84   ChannelEstimator();
85
86   virtual ~ChannelEstimator();
87
88
89   virtual int command(int argc, const char*const* argv);
90
91
92   static double getSpaceSampling() { return space_sampling; }
93
94
102    virtual void updateEstimation( const woss::CoordZ& tx, const woss::CoordZ& rx, woss::TimeArr*
      curr_channel );
```

```
103
111   woss::TimeArr* getEstimation( const woss::CoordZ& tx, const woss::CoordZ& rx );
112
113
114   void addMacAddress( int addr, WossPosition* pos ) { mac_to_pos_map[addr] = pos; }
115
116   virtual WossPosition* findMacAddress( int addr );
117
118
119   virtual bool resetEstimator();
120
121
122   protected:
123
124
125   static double space_sampling;
126
127
128   ChannelMap channel_map;
129
130   MacToPosMap mac_to_pos_map;
131
132
133   double debug_;
134
135   double avg_coeff;
136
137 };
138
139
146 class ChEstimatorPlugIn :  public PlugIn {
147
148
149   public:
150
151
152   ChEstimatorPlugIn();
153
154   virtual ~ChEstimatorPlugIn() { }
155
156
157   virtual int command(int argc, const char*const* argv);
158
159   virtual int recvSyncClMsg(ClMessage* m);
160
161
162   protected:
163
164
165   ChannelEstimator* channel_estimator;
166
167   double debug_;
168
169 };
170
171
172 #endif  // CHANNEL_ESTIMATOR_H
173
```

14.158 woss_phy/uw-woss-channel.cpp File Reference

Provides the implementation of WossChannelModule class.

Classes

- class **WossChannelModuleClass**

14.158.1 Detailed Description

Provides the implementation of WossChannelModule class.

Author

Federico Guerra

Provides the implementation of WossChannelModule class

14.159 woss_phy/uw-woss-channel.h File Reference

Provides the interface for WossChannelModule class.

Classes

- class WossChannelModule

  WossChannelModule class for channel calculations with WOSS.

14.159.1 Detailed Description

Provides the interface for WossChannelModule class.

Author

Federico Guerra

Provides the interface for WossChannelModule class

14.160 uw-woss-channel.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UW_WOSS_CHANNEL_H
41 #define UW_WOSS_CHANNEL_H
42
43
44 #include <channel-module.h>
45 #include <underwater.h>
46 #include <woss-manager.h>
47
48
49 namespace woss {
50   class TimeArr;
51   class WossManager;
52 }
53
54
55 class ChannelEstimator;
```

```
56
57
63 class  WossChannelModule :  public ChannelModule
64 {
65
66
67   public:
68
69
70   WossChannelModule();
71
72   virtual ~ WossChannelModule();
73
74
75   virtual int command(int argc, const char*const* argv);
76
77
78   virtual void recv(Packet* p, ChSAP* chsap);
79
80
81   protected:
82
83
84   typedef ::std::vector< ChSAP* > ChSAPVector;
85
86
87   double getPropDelay( const woss::CoordZ& s, const woss::CoordZ& d);
88
89
90   woss::CoordZPairVect computeCoordZPairVect( ChSAP* dest );
91
92   woss::SimFreq computeSimFreq( Packet* p );
93
94   woss::TimeArrVector computeTimeArrVector( const woss::CoordZPairVect& coords, const woss::SimFreq&
      sim_freq );
95
96   void checkTimeArrVector( const woss::CoordZPairVect& coords, woss::TimeArrVector& channels );
97
98   void schedulePacketCopies( const woss::CoordZPairVect& coords, const woss::TimeArrVector& channels,
      const woss::SimFreq& sim_freq, Packet* p );
99
100   void deleteChannels( woss::TimeArrVector& channels );
101
102
103   double channel_eq_snr_threshold_db;
104
105   double channel_symbol_resolution;
106
107   double channel_eq_time;
108
109   double channel_eq_attenuation_db;
110
111   double channel_max_distance; // [m]
112
113
114   woss::WossManager* woss_manager;
115
116   ChannelEstimator* channel_estimator;
117
118
119   Underwater uw;
120
121   ChSAPVector chsap_vector;
122
123 };
124
125
126 #endif /* UW_WOSS_CHANNEL_H */
127
```

14.161 woss_phy/uw-woss-clmsg-channel-estimation.cpp File Reference

Provides the implementation of ClMsgChannelEstimation class.

Variables

- ClMessage_t **CLMSG_CHANNEL_ESTIMATION**

14.161.1 Detailed Description

Provides the implementation of ClMsgChannelEstimation class.

Author

Federico Guerra

Provides the implementation of ClMsgChannelEstimation class

14.162 woss_phy/uw-woss-clmsg-channel-estimation.h File Reference

Provides the interface for ClMsgChannelEstimation class.

Classes

- class ClMsgChannelEstimation
  Class for channel estimation synchronous cross-layer messaging.

Variables

- ClMessage_t **CLMSG_CHANNEL_ESTIMATION**

14.162.1 Detailed Description

Provides the interface for ClMsgChannelEstimation class.

Author

Federico Guerra

Provides the interface for ClMsgChannelEstimation class

14.163 uw-woss-clmsg-channel-estimation.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UW_WOSS_CLMSG_CHANNEL_ESTIMATION_H
41 #define UW_WOSS_CLMSG_CHANNEL_ESTIMATION_H
42
43
44 #include <clmessage.h>
45 // #include <coordinates-definitions.h>
46
47 #define CLMSG_CH_ESTIMATION_VERBOSITY 2  // verbosity of this message
48
49
50 extern ClMessage_t CLMSG_CHANNEL_ESTIMATION;
51
52
53 namespace woss {
54   class TimeArr;
55 }
56
57
64 class ClMsgChannelEstimation :  public ClMessage {
65
66
67   public:
68
69
70   ClMsgChannelEstimation( int i, int j, woss::TimeArr* time_arr = NULL );
71
72   virtual ~ClMsgChannelEstimation();
73
74   virtual ClMessage* copy();  // copy the message
75
76
77   const int getTx() { return tx; }
78
79   const int getRx() { return rx; }
80
81
82   void setTimeArr( woss::TimeArr* time_arr ) { ch_estimation = time_arr; }
83
84   woss::TimeArr* getTimeArr() { return ch_estimation; }
85
86
87   void setDeletable() { deletable = true; }
88
89   void unsetDeletable() { deletable = false; }
90
91
92   bool isQuery() {return query; }
93
94   bool isDeletable() { return deletable; }
95
96
97   protected:
98
99
```

```
100   bool query;
101
102   bool deletable;
103
104   int tx;
105
106   int rx;
107
108   woss::TimeArr* ch_estimation;
109
110
111 };
112
113
114 #endif //UW_WOSS_CLMSG_CHANNEL_ESTIMATION_H
115
```

14.164 woss_phy/uw-woss-mpropagation.cpp File Reference

Provides the implementation of WossMPropagation class.

Classes

- class **WossMPropagationClass**

14.164.1 Detailed Description

Provides the implementation of WossMPropagation class.

Author

Federico Guerra

Provides the implementation of WossMPropagation class

14.165 woss_phy/uw-woss-mpropagation.h File Reference

Provides the interface for WossMPropagation class.

Classes

- class WossMPropagation

  UnderwaterMPropagation class for channel calculations with WOSS.

Typedefs

- typedef std::map< woss::CoordZ, std::map< woss::CoordZ, double > > **GainMatrix**
- typedef GainMatrix::iterator **GMIter**
- typedef GainMatrix::reverse_iterator **GMRIter**

14.165.1 Detailed Description

Provides the interface for WossMPropagation class.

Author

Federico Guerra

Provides the interface for WossMPropagation class

14.166 uw-woss-mpropagation.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_PROPAGATION_H
41 #define UNDERWATER_WOSS_PROPAGATION_H
42
43
44 #include <fstream>
45 #include <map>
46 #include <underwater-mpropagation.h>
47 #include <underwater.h>
48 #include <mphy.h>
49 #include "uw-woss-pkt-hdr.h"
50 #include <coordinates-definitions.h>
51
52
53 namespace woss {
54   class WossManager;
55 }
56
57
58 typedef std::map< woss::CoordZ , std::map < woss::CoordZ , double > > GainMatrix;
59 typedef GainMatrix::iterator GMIter;
60 typedef GainMatrix::reverse_iterator GMRIter;
61
62
68 class WossMPropagation :  public UnderwaterMPropagation {
69
70
71   public:
72
73
74   WossMPropagation();
75
76   virtual ~WossMPropagation() { }
77
78
```

```
85   virtual double getGain(Packet* p);
86
87
88   virtual int command(int argc, const char*const* argv);
89
90
91   private:
92
93
94   woss::WossManager* woss_manager;
95
96
97   GainMatrix std_gain_map;
98
99
100   bool write_gain_matrix;
101
102
103   fstream std_gain_out;
104
105
106   string std_gain_matrix_name;
107
108
109   void insertStdGainMatrix( Position* sp, Position* rp, double gain );
110
111
118   double computeGain(Packet* p);
119
120
121   void writeStdGainMatrix();
122
123
124 };
125
126
127 //inline functions
129
130 inline void WossMPropagation::insertStdGainMatrix( Position* sp, Position* rp, double gain ) {
131   std_gain_map[ woss::CoordZ( sp->getLatitude(), sp->getLongitude(), abs( sp->getZ() ) ) ]
132               [ woss::CoordZ( rp->getLatitude(), rp->getLongitude(), abs( rp->getZ() ) ) ] = gain;
133 }
134
135 #endif /* UNDERWATER_WOSS_PROPAGATION_H */
```

14.167 woss_phy/uw-woss-pkt-hdr.h File Reference

Definition of hdr_woss, WOSS pkt header.

Classes

- struct hdr_woss
  WOSS packet header.

Typedefs

- typedef struct hdr_woss hdr_woss
  WOSS packet header.

14.167.1 Detailed Description

Definition of hdr_woss, WOSS pkt header.

Author

Federico Guerra

Definition of hdr_woss, WOSS pkt header

14.167.2 Typedef Documentation

14.167.2.1 hdr_woss typedef struct hdr_woss hdr_woss

WOSS packet header.

struct hdr_woss extends Miracle MPhy header adding new information

14.168 uw-woss-pkt-hdr.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UW_WOSS_PKT_H
41 #define UW_WOSS_PKT_H
42
43
44 #define HDR_WOSS(P)       (hdr_woss::access(P))
45
46
47 #include <packet.h>
48 #include <complex>
49
50
56 typedef struct hdr_woss {
57
58
62   std::complex<double> attenuation;
63
67   double frequency;
68
72   bool already_processed;
73
74   static int offset_;
75   inline static int& offset() { return offset_; }
76   inline static struct hdr_woss* access(const Packet* p) {
77     return (struct hdr_woss*)p->access(offset_);
78   }
79
80 } hdr_woss;
81
82
83 #endif /* UW_WOSS_PKT_H */
```

14.169 woss_phy/uw-woss-position.cpp File Reference

Provides the implementation of WossPosition class.

Classes

- class **WossPositionClass**

14.169.1 Detailed Description

Provides the implementation of WossPosition class.

Author

Federico Guerra

Provides the implementation of WossPosition class

14.170 woss_phy/uw-woss-position.h File Reference

Provides the interface for WossWpPosition class.

Classes

- class WossPosition

14.170.1 Detailed Description

Provides the interface for WossWpPosition class.

Author

Federico Guerra

Provides the interface for WossWpPosition class

14.171 uw-woss-position.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_POSITION_H
41 #define UNDERWATER_WOSS_POSITION_H
42
43
44 #include <location-definitions.h>
45 #include <node-core.h>
46
47
48 class WossPosition :  public Position, public woss::Location {
49
50
51   public:
52
53
54   WossPosition( double latitude, double longitude, double depth = 0, double dist =
      LOCATION_COMPARISON_DISTANCE );
55
56   WossPosition( const woss::CoordZ& coordz = woss::CoordZ(), double dist = LOCATION_COMPARISON_DISTANCE
      );
57
58
59   virtual ~WossPosition() {  }
60
61
62   virtual WossPosition* create( double latitude, double longitude, double depth = 0, double dist =
      LOCATION_COMPARISON_DISTANCE )const {
63     return new WossPosition(latitude, longitude, depth); }
64
65   virtual WossPosition* create( const woss::CoordZ& coordz = woss::CoordZ(), double dist =
      LOCATION_COMPARISON_DISTANCE )const {
66     return new WossPosition( coordz, dist ); }
67
68   virtual WossPosition* clone()const { return new WossPosition(*this); }
69
70
71   virtual int command(int argc, const char*const* argv);
72
73
74   virtual void setLatitude( double val );
75
76   virtual void setLongitude( double val );
77
78   virtual void setDepth( double val );
79
80   virtual void setAltitude( double val );
81
82
83   virtual void setMinVerticalOrientation( double val );
84
85   virtual void setMaxVerticalOrientation( double val );
86
87
88   virtual void setX( double val );
89
```

```
90   virtual void setY( double val );
91
92   virtual void setZ( double val );
93
94
95   virtual double getX();
96
97   virtual double getY();
98
99   virtual double getZ();
100
101
102   virtual double getLatitude();
103
104   virtual double getLongitude();
105
106   virtual double getDepth();
107
108   virtual double getAltitude();
109
110
111   virtual double getMinVerticalOrientation();
112
113   virtual double getMaxVerticalOrientation();
114
115
116   protected:
117
118
119   double min_vertical_orientation;
120
121   double max_vertical_orientation;
122
123
124 };
125
126
127 #endif // UNDERWATER_WOSS_POSITION_H
128
129
```

14.172 woss_phy/uw-woss-random-generator.cpp File Reference

Provides the implementation of WossRandomGenerator class.

Classes

- class **WossRandomGeneratorTclClass**

14.172.1 Detailed Description

Provides the implementation of WossRandomGenerator class.

Author

Federico Guerra

Provides the implementation of WossRandomGenerator class

14.173 woss_phy/uw-woss-random-generator.h File Reference

Provides the interface for WossRandomGenerator and WossRandomGeneratorTcl classes.

Classes

- class WossRandomGenerator
- class WossRandomGeneratorTcl

14.173.1 Detailed Description

Provides the interface for WossRandomGenerator and WossRandomGeneratorTcl classes.

Author

Federico Guerra

Provides the interface for WossRandomGenerator and WossRandomGenerator classes

14.174 uw-woss-random-generator.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_RANDOM_GENERATOR_H
41 #define UNDERWATER_WOSS_RANDOM_GENERATOR_H
42
43
44 #include <random-generator-definitions.h>
45 #include <rng.h>
46 #include <tclcl.h>
47
48
49 class WossRandomGenerator :  public woss::RandomGenerator {
50
51
52     public:
53
54
55     WossRandomGenerator( int seed = 0 );
56
57
58     WossRandomGenerator( const WossRandomGenerator& copy );
59
60     WossRandomGenerator& operator=( const WossRandomGenerator& copy );
61
62
63     virtual ~WossRandomGenerator();
64
```

```
65
66      virtual WossRandomGenerator* create( int s ) { return new WossRandomGenerator(s); }
67
68      virtual WossRandomGenerator* clone()const { return new WossRandomGenerator(*this); }
69
70
71      virtual double getRand() const;
72
73      virtual int getRandInt() const;
74
75      virtual void initialize();
76
77
78      protected:
79
80
81      RNG* rng;
82
83
84  };
85
86
87  class WossRandomGeneratorTcl :  public WossRandomGenerator, public TclObject {
88
89
90    public:
91
92
93    WossRandomGeneratorTcl();
94
95
96    virtual int command(int argc, const char*const* argv);
97
98
99  };
100
101
102  #endif // UNDERWATER_WOSS_RANDOM_GENERATOR_H
103
```

14.175 woss_phy/uw-woss-time-reference.cpp File Reference

Provides the implementation of WossTimeReferenceTcl class.

Classes

- class **WossTimeReferenceTclClass**

14.175.1 Detailed Description

Provides the implementation of WossTimeReferenceTcl class.

Author

Federico Guerra

Provides the implementation of WossTimeReferenceTcl class

14.176 woss_phy/uw-woss-time-reference.h File Reference

Provides the interface for WossTimeReferenceTcl class.

Classes

- class WossTimeReference
- class WossTimeReferenceTcl

14.176.1 Detailed Description

Provides the interface for WossTimeReferenceTcl class.

Author

Federico Guerra

Provides the interface for WossTimeReferenceTcl class

14.177 uw-woss-time-reference.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_TIME_REFERENCE_H
41 #define UNDERWATER_WOSS_TIME_REFERENCE_H
42
43
44 #include <time-definitions.h>
45 #include <tclcl.h>
46 #include <scheduler.h>
47
48
49 class WossTimeReference :  public woss::TimeReference {
50
51
52   public:
53
54   virtual ~WossTimeReference() { }
55
56   virtual double getTimeReference()const { return Scheduler::instance().clock(); }
57
58   virtual WossTimeReference* clone() { return new WossTimeReference(*this); }
59
60
61 };
62
63
64 class WossTimeReferenceTcl :  public TclObject, public WossTimeReference { };
65
66
67 #endif // UNDERWATER_WOSS_TIME_REFERENCE_H
```

14.178 woss_phy/uw-woss-waypoint-position.cpp File Reference

Provides the implementation of WossWpPosition class.

Classes

- class **WossWpPositionClass**

14.178.1 Detailed Description

Provides the implementation of WossWpPosition class.

Author

Federico Guerra

Provides the implementation of WossWpPosition class

14.179 woss_phy/uw-woss-waypoint-position.h File Reference

Provides the interface for WossWpPosition class.

Classes

- class WossWpPosition
- class WossWpPosition::WayPoint

14.179.1 Detailed Description

Provides the interface for WossWpPosition class.

Author

Federico Guerra

Provides the interface for WossWpPosition class

14.180 uw-woss-waypoint-position.h

Go to the documentation of this file.

```
1 /* WOSS - World Ocean Simulation System -
2 *
3 * Copyright (C) 2009 Federico Guerra
4 * and regents of the SIGNET lab, University of Padova
5 *
6 * Author:  Federico Guerra - federico@guerra-tlc.com
7 *
8 * This program is free software; you can redistribute it and/or modify
9 * it under the terms of the GNU General Public License version 2 as
10 * published by the Free Software Foundation;
11 *
12 * This program is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
15 * GNU General Public License for more details.
16 *
17 * You should have received a copy of the GNU General Public License
18 * along with this program; if not, write to the Free Software
19 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
20 */
21
22 /*
23 * This software has been developed by Federico Guerra and SIGNET lab,
24 * University of Padova, in collaboration with the NATO Centre for
25 * Maritime Research and Experimentation (http://www.cmre.nato.int ;
26 * E-mail:  pao@cmre.nato.int), whose support is gratefully acknowledged.
27 */
28
29
40 #ifndef UNDERWATER_WOSS_WAYPOINT_POSITION_H
41 #define UNDERWATER_WOSS_WAYPOINT_POSITION_H
42
43
44 #include <map>
45 #include <utility>
46 #include <climits>
47 #include "uw-woss-position.h"
48
49
50 class WossWpPosition :  public WossPosition {
51
52
53   public:
54
55
56   WossWpPosition();
57
58   virtual ~WossWpPosition() { }
59
60
61   virtual int command(int argc, const char*const* argv);
62
63
64   virtual bool isEquivalentTo( const woss::CoordZ& coordz );
65
66
67   virtual woss::CoordZ getLocation();
68
69   virtual double getVerticalOrientation();
70
71   virtual double getBearing();
72
73   virtual double getSpeed();
74
75
76   protected:
77
78
79   class WayPoint {
80
81
82     public:
83
84
85     WayPoint( const woss::CoordZ& destination, double speed = 0.0, double time = HUGE_VAL, int loop_id =
      INT_MAX, int total_loops = 0 );
86
87
88     void setDestination( const woss::CoordZ& dest ) { destination = dest; }
89
90     void setSpeed( double s ) { speed = s; }
91
92     void setTimeToWait( double t ) { time_to_wait = t; }
```

```
93
94      void setLoopId( int id ) { loop_to_waypoint_id = id; }
95
96      void setTotalLoops( int loops ) { total_loops = loops; }
97
98
99      const woss::CoordZ& getDestination()const { return destination; }
100
101      // speed used to reach this node
102      double getSpeed()const { return speed; }
103
104      double getTimeToWait()const { return time_to_wait; }
105
106      int getLoopId()const { return loop_to_waypoint_id; }
107
108      int getTotalLoops()const { return total_loops; }
109
110
111      bool isValid()const { return( destination.isValid() && !( speed == 0.0 && time_to_wait == 0.0 ) ); }
112
113      bool hasToLoop()const { return( loop_to_waypoint_id != INT_MAX && total_loops != 0.0 ); }
114
115      bool hasToWait()const { return( speed == 0.0 && time_to_wait > 0 && time_to_wait != HUGE_VAL ); }
116
117      bool hasToStop()const { return( speed == 0.0 && time_to_wait == HUGE_VAL ); }
118
119
120      virtual double getTimeOfArrival( const WayPoint& dest_waypoint ) const;
121
122      virtual woss::CoordZ getCurrentPosition( const WayPoint& dest_waypoint, double time_elapsed ) const;
123
124
125      friend std::ostream& operator«( std::ostream& os, const WayPoint& instance ) {
126        os « instance.destination « "; speed = " « instance.speed « "; time to wait = " «
     instance.time_to_wait
127           « "; loop id = " « instance.loop_to_waypoint_id «"; total loops = " « instance.total_loops;
128        return os;
129      }
130
131      protected:
132
133
134      woss::CoordZ destination;
135
136      double speed;
137
138      double time_to_wait;
139
140      int loop_to_waypoint_id;
141
142      int total_loops;
143
144
145    };
146
147    typedef ::std::vector< WayPoint > WayPointVect;
148
149    typedef ::std::map< double, int > TimeIdMap;
150    typedef TimeIdMap::iterator TIMIter;
151    typedef TimeIdMap::reverse_iterator TIMRIter;
152
153
154    double time_threshold;
155
156    double last_time_update;
157
158    double current_speed;
159
160    WayPointVect waypoint_vect;
161
162    TimeIdMap timeid_map;
163
164
165    virtual void update( double now );
166
167    virtual void updateVerticalOrientation( const woss::CoordZ& prev, const woss::CoordZ& curr );
168
169    virtual void updateBearing( const woss::CoordZ& prev, const woss::CoordZ& curr );
170
171
172    virtual double addWayPoint( const WayPoint& waypoint );
173
174    virtual double addLoopPoint( const WayPoint& waypoint );
175
176
177  };
178
```

```
179
180 #endif // UNDERWATER_WOSS_WAYPOINT_POSITION_H
181
```

Index