A data format for tomography

OCTOPUS Group
Rev. 1.1 : 08/06/2001

June 8, 2001

A glossary of the numerous variables defined in the various netCDF files of the OCTOPUS project is given here. If some variables are rather self descriptive, others require some comments which are not always possible to write in the *long_name* attribute. This also gives the defined files and their contents.

Description and technical characteristics of the tomographic instruments involved in an experiment are stored together with some raw data (long baseline positionning data, raw temperature and pressure time series, clock checking data) in a file called **instrument** file (**i_** type). The available pool of instrument is described in a data base of instrument **dbi** file). Each experiment requires to be defined through an **experiment** file (**e_** type) where are stored the geometric and geographic information. Some processed data (mooring motions, clock drifts, calibrated and corrected pressure and temperature time series) are collected in that file too. A priori knowledge on the ocean is stored in the **ocean** directory in **o_** files. Acoustic predictions describing the rays and modes parameters and geometry are saved in the so-called **acous** directory in **a_** files (related to direct problem) or **z_** files (related to inversion). Depending on their state of processing, the acoustic data for each pair are stored in a directory called either **Level-0, Level-1, Level-2 or Level-3**. At each stage of the process a format is defined. This paper gives the format and describes the contents of each file.

Contents

Document history

Version	Updated by	Date	History
0.1.0	T.Terre	30/01/1998	Title = Bilan des informations nécessaires
0.1.1	T.Terre	30/06/1998	Description in netCDF CDL of the data
0.2.0	G.Maudire	10/1998	Title = Octopus : File formats Description of formats after 2nd meeting (Heraklion)
0.3.0	T.Terre	12/03/1999	Title = Glossary of variable used in OCTOPUS netCDF files. Revision of formats after 3rd meeting (Grenoble)
0.3.1	T.Terre	08/11/1999	Revision of formats after 4th meeting (Kiel)
0.3.2	T.Terre	21/12/1999	Revision of formats for alpha release of toolboxes
0.4.0	T.Terre	15/02/1999	Title = Octopus : File formats Revision for beta release of toolboxes
0.5.0	T.Terre	15/03/2000	Revision of formats after 5th meeting (Brest) Set all positions in double Add Level-2 specifications
0.6.0	T.Terre	26/11/2000	Revision of formats after 6th meeting (Heraklion) Add format for CTD and reduced databases Set all travel times in double
0.6.1	T.Terre	14/12/2000	Add new variable and dimension for clock drift corrections in experiment file
0.6.2	T.Terre	31/01/2001	Add new variables for X,Y Z offsets of interrogation transducer for transponders survey.
0.6.3	T.Terre	01/02/2001	Change definition of Sect_flags in Ocean data file
0.6.4	T.Terre	26/02/2001	Change definitions of Mode_mean_par_descr and Mode_mean_par in Ocean data file
0.6.5	T.Terre	15/03/2001	Add figure for directories structures
0.7.0	T.Terre	29/03/2001	Add Level-3 specifications
0.7.1	T.Terre	02/04/2001	Modify Level-3 specifications (add some units)
0.7.2	T.Terre	02/05/2001	Add Sect_center in o_file and set Fillvalue attribute of Sect_flag to 0
0.7.3	T.Terre	09/05/2001	Modify M_actual_depth definition
0.7.4	T.Terre	23/05/2001	Add Experiment_history variable
0.7.5	T.Terre	31/05/2001	Update for final versions of acoustic files and level 2 and 3 files
1.0	T.Terre	01/06/2001	Title = A data format for tomography Author = OCTOPUS Group
1.1	T.Terre	08/06/2001	Change naming conventions for files

File formats status

File	Status	last update
Experiment	stabilized	03/2000
Instrument	stabilized	03/1999
Ocean data	stabilized	03/2000
Direct acoustic	stabilized	03/2000
Inverse acoustic	stabilized	03/2000
Level-0	stabilized	10/1999
Level-1	stabilized	10/1999
Level-2	stabilized	11/2000
Level-3	stabilized	05/2001

Stabilized : format is used and only small changes could happen

Defined : format is used but some points are still in discussions

Described : format is in discussion

1 Conventions

1.1 Sign conventions

The following sign conventions are used :

- depths are positive downward,
- corrections are always added i.e.

  `True/best value = Uncorrected value + correction or delay`

1.2 File name conventions

The number of characters to name files is not limited but the filenames should follow the general and particular conventions described thereafter.

- **Data base of instrument** is a unique file named `dbi.nc`. It should be located in TOMOLAB databases path.
- **Instrument file :** the naming convention for the instrument file is **i_ccccc.nc** where
  - **i_** is the reserved prefix for this file type.
  - **ccccc** One or more alphanumeric characters to name the experiment and/or to discriminate between different versions of the file (example : **i_thetis2_a** for Thetis 2 version a).
- **Experiment file :** the naming convention for the experiment file is **e_ccccc.nc** where
  - **e_** is the reserved prefix for this file type.
  - **ccccc** One or more characters to name the experiment and/or to discriminate between different versions of the file (example : **e_thetis2_a** for Thetis 2 version a).
- **Ocean data file :** the naming convention for the ocean data file is **o_ccccc.nc** where
  - **o_** is the reserved prefix for this file type.
  - **ccccc** One or more alphanumeric characters to name the experiment and/or to discriminate between different versions of the file (example : **o_thetis2_a** for Thetis 2 version a)

All files described below are named according to the same convention. The first part of the name describes the pair and, except for level-0, the second part identifies the version.

- **Level-0 file :** all the level-0 data files are grouped in a directory named **level0** and the naming convention for the data files is **xxxyyy.lv0** where
  - **xxx** One or more alphanumeric characters for the receiver mooring name,
  - **yyy** Same number of alphanumeric characters for the source mooring name,
  - **.lv0** the reserved extension for the level
- **Level-1 file :** the naming convention for the data file is **xxxyyy[.ccc].lv1** where
  - **xxx** One or more alphanumeric characters for the first mooring name,
  - **yyy** Same number of alphanumeric characters for the second mooring name,
  - **.lv1** the reserved extension for the level
  - **[.ccc]** Optional unlimited number of characters to build different versions of the file. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.
- **Level-2 file :** the naming convention for the data file is **xxxyyy.lv2** where
  - **xxx** One or more alphanumeric characters for the first mooring name,
  - **yyy** Same number of alphanumeric characters for the second mooring name,
  - **.lv2** the reserved extension for the level
  - **[.ccc]** Optional unlimited number of characters to build different versions of the file. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.
- **Level-3 file :** the naming convention for the data file is **xxxyyy[.ccc].lv3** where

  - **xxx** One or more alphanumeric characters for the first mooring name,
  - **yyy** Same number of alphanumeric characters for the second mooring name,
  - **.lv3** the reserved extension for the level
  - **[.ccc]** Optional unlimited number of characters to build different versions of the file. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

- **Acoustic data file :** the naming convention for the data file is **a_xxxyyy[.ccc].nc** or **z_xxxyyy[.ccc].nc** where
  - **xxx** One or more alphanumeric characters for the 1rst mooring name,
  - **yyy** Same number of alphanumeric characters for the 2nd mooring name,
  - **[.ccc]** One or more optional alphanumeric characters for distinguishing different file versions. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

2 Instrument file associated to an experiment

This file is described by the **i_model.cdl** model and contains all the technical data regarding the instruments deployed in an experiment. There is a single instrument file for each experiment. The file content is for a large part extracted from the data base file **dbi.nc**. Specific instrumental data relative to the experiment are written here too. In particular, this file contains :

- the technical characteristics of the acoustic transponders of the long baseline positionning systems;
- the sensors calibrations attached to the experiment;
- the status data acquired along the experiment for all modules;
- the raw navigation data i.e. unfiltered, uncorrected travel times as measured by the instruments navigator;
- the raw pressure and temperature data i.e. uncorrected and unconverted as delivered by the sensors electronic.

A naming convention is used to distinguish between the dimensions which depend only on the instrument characteristics and those which are dependant on the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_SENSORS** for example, while all experiment related dimensions are prefixed with **e** like **eN_TOMO**.
The naming convention for the instrument file is **i_ccccc.nc** where

- **i_** is the reserved prefix for this file type.
- **ccccc** One or more alphanumeric characters to name the experiment and/or to discriminate between different versions of the file (example : **i_thetis2_a** for Thetis 2 version a).

2.1 Dimensions

- **N_NAME** This is the maximum number of characters in an instrument or module name.
- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_STRING** The maximum number of characters in a short string.

- **N_LSTR** The maximum number of characters in a long string.
- **N_TASK** Maximum number of bytes in tasks.
- **TWO** Used to set 2 dimensions to arrays.
- **iN_BEACON** Number of navigation channels.
- **iN_POW_SUP** The maximum number of power supplies in the instruments.
- **iN_IR** Maximum number of points used to describe the source frequency response of each instrument.
- **iN_COEF** Maximum number of coefficients for the sensors conversion polynomials.
- **iN_VAL** Maximum number of internal parameters.
- **eN_TOMO** Number of tomographic instruments deployed in the experiment.
- **eN_FREQ** Maximum number of frequencies used in the experiment.
- **eN_CHAN** Maximum number of receiver channels used in the experiment.
- **eN_PING** Maximum number of ping sent per navigation.
- **eN_CAL** Maximum numbers of sensors calibrations done for the experiment.
- **eN_VALP** Maximum numbers of experiment related calibration points for pressure.
- **eN_VALT** Maximum numbers of experiment related calibration points for temperature.
- **eN_RCK** Maximum number of clock/reference comparisons.
- **eN_TX** Maximum number of transmissions during the experiment.
- **eN_RX** Maximum number of receptions during the experiment.
- **eN_NV** Maximum number of navigations during the experiment.

- **eN_IP** Maximum number of internal parameters measurements during the experiment.
- **eN_PT** Maximum number of environmental parameters during the experiment.
- **eN_ST** Maximum number of storage done during the experiment.

2.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of the last update of the file.
- **comments(N_LSTR)** *char* Comments on the update, file content...

Instrument

- **I_id(eN_TOMO, N_NAME)** *char* Instrument identifier.
- **I_owner(eN_TOMO, N_STRING)** *char* Instrument owner.
- **I_type(eN_TOMO, N_STRING)** *char* Instrument type.
- **I_function(eN_TOMO, N_STRING)** *char* Instrument function.
- **I_max_depth(eN_TOMO)** *short* Maximum allowed depth for the instrument. *Unit = m, Range = [0, 2000].*
- **I_bat_cap(eN_TOMO, iN_POW_SUP)** *short* A number which indicates the capacity of each battery block dedicated to each power supply. The convention for the power supplies is : 1- emission function; 2- Navigation/storage; 3- General electronics; 4- Clock; *Unit = Ah.*
- **I_nb_bat(eN_TOMO)** *short* A number which gives the maximum number of batteries blocks dedicated to the emission function for each instrument. *Range = [1, 60].*

Controller

- **CX_id(eN_TOMO, N_NAME)** *char* Controller identifier.
- **CX_ver_soft(eN_TOMO, N_STRING)** *char* Controller software version.

- **CX_gen_task(eN_TOMO, N_STRING)** *char* Task generator version.
- **CX_comp_task(eN_TOMO, N_STRING)** *char* Task compiler version.
- **CX_cons_sleep(eN_TOMO)** *float* Controller consumption in sleep mode. *Unit = Ah.*
- **CX_cons_awake(eN_TOMO)** *float* Controller consumption in run mode. *Unit = Ah.*
- **CX_ip_n(eN_TOMO)** *short* Number of measured internal parameters.
- **CX_ip_run(eN_TOMO)** *short* Execution time of internal parameters measure. *Unit = s.*
- **CX_task(eN_TOMO, N_TASK)** *byte* Binary of the task description. This is the image of the task downloaded to the instrument controller.
- **CX_date(eN_TOMO, eN_IP)** *int* Date and time of internal parameters measurements expressed in seconds since the experiment reference time. *Unit = s.*
- **CX_param(eN_TOMO, TWO, iN_VAL, eN_IP)** *short* Raw and internally converted values of the internal parameters.

Clock

- **CK_id(eN_TOMO, N_NAME)** *char* Clock identifier.
- **CK_type(eN_TOMO, N_STRING)** *char* Clock type.
- **CK_set_fr(eN_TOMO)** *short* Setting time to obtain a stabilized 1 MHz frequency. *Unit = s.*
- **CK_cons_sleep(eN_TOMO)** *float* Consumption of the clock in sleep mode. *Unit = Ah.*
- **CK_cons_awake(eN_TOMO)** *float* Consumption of the clock in run mode. *Unit = Ah.*
- **CK_aging_date(eN_TOMO, N_DATE_TIME)** *char* Aging correction date.

- **CK_aging_temp(eN_TOMO)** *float* Aging correction temperature.
- **CK_aging_coef(eN_TOMO)** *short* Aging correction coefficient.
- **CK_date_ante(eN_TOMO, N_DATE_TIME)** *char* Date of clock checking before the deployment.
- **CK_utc_ante(eN_TOMO)** *float* Offset with UTC reference before the deployment. *Unit = s.*
- **CK_date_post(eN_TOMO, N_DATE_TIME)** *char* Date of clock checking after the recovery.
- **CK_utc_post(eN_TOMO)** *float* Offset with UTC reference after the recovery. *Unit = s.*
- **CK_drift(eN_TOMO)** *float* Drift measured during the experiment.
- **CK_date_check(eN_TOMO, eN_RCK)** *int* Date of extra clock checking (if any).
- **CK_offset(eN_TOMO, eN_RCK)** *float* Offset with UTC reference for the extra clock checking. *Unit = s.*

Transducer

- **TR_id(eN_TOMO, N_NAME)** *char* Transducer identifier.
- **TR_type(eN_TOMO, N_STRING)** *char* Transducer type.
- **TR_frequency(eN_TOMO)** *short* Transducer frequency. *Unit = Hz.*
- **TR_bandwidth(eN_TOMO)** *short* Transducer bandwidth. *Unit = Hz.*
- **TR_level(eN_TOMO)** *short* Nominal acoustic level for the source. *Unit = dB re 1μPa @ 1 m.*
- **TR_ir_fr(eN_TOMO, iN_IR)** *float* Frequency points for the transducer frequency response (if measured). *Unit = Hz.*
- **TR_ir_lv(eN_TOMO, iN_IR)** *float* Measured levels of the transducer frequency response (if measured). *Unit = dB.*

Power amplifier

- **PA_id(eN_TOMO, N_NAME)** *char* Power amplifier identifier.
- **PA_type(eN_TOMO, N_NAME)** *char* Power amplifier type.
- **PA_power(eN_TOMO)** *short* Electrical power. *Unit = VA.*
- **PA_cons_awake(eN_TOMO)** *floatt* Consumption of the power amplifier in run mode. *Unit = Ah.*

Transmitter

- **TX_id(eN_TOMO, N_NAME)** *char* Transmitter identifier.
- **TX_ver_soft(eN_TOMO, N_STRING)** *char* Transmitter software version.
- **TX_frequency(eN_TOMO)** *short* Source frequency. *Unit = Hz.*
- **TX_signal_code(eN_TOMO)** *short* Signal code (Shift register loop for PSK signal). The initial state is 1 (must first go all registers before coming out).
- **TX_delay(eN_TOMO)** *float* Delay between the SYNC pulse and the effective sound. *Unit = s.*
- **TX_cons_awake(eN_TOMO)** *float* Consumption of the source in run mode. *Unit = Ah.*
- **TX_arm(eN_TOMO)** *short* Time needed by the controller to prepare a transmission : power on and initialization for the transmission. *Unit = s.*
- **TX_run(eN_TOMO)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the transmission. *Unit = s.*
- **TX_end(eN_TOMO)** *short* Time needed by the controller to cleanly terminate the transmission and retrieve the transmitter status. *Unit = s.*
- **TX_n(eN_TOMO)** *short* Number of transmissions done per instrument during the experiment.
- **TX_date(eN_TOMO, eN_TX)** *int* Transmission dates. Those are expressed in seconds since the experiment reference date. *Unit = s.*

- **TX_sail(eN_TOMO, eN_TX)** *byte* Emitter Sail status.
- **TX_carrier(eN_TOMO, eN_TX)** *short* Emitted carrier frequency. *Unit = Hz.*
- **TX_seq(eN_TOMO, eN_TX)** *short* Number of emitted sequences.
- **TX_pres(eN_TOMO, eN_TX)** *short* Raw external pressure at transmission time. *Unit = arbitrary unit.*
- **TX_mes1(eN_TOMO, eN_TX)** *short* Batteries voltage. *Unit = arbitrary unit.*
- **TX_mes2(eN_TOMO, eN_TX)** *short* Duty cycle or transducer current. *Unit = arbitrary unit.*
- **TX_mes3(eN_TOMO, eN_TX)** *short* Duty cycle or transducer voltage. *Unit = arbitrary unit.*

Receiver

- **RX_id(eN_TOMO, N_NAME)** *char* Receiver identifier.
- **RX_ver_soft(eN_TOMO, N_STRING)** *char* Receiver software version.
- **RX_frequencies(eN_TOMO, eN_FREQ)** *short* Received frequencies during the experiment. *Unit = Hz.*
- **RX_channel(eN_TOMO)** *short* Number of received channels during the experiment.
- **RX_gain(eN_TOMO, eN_FREQ, eN_CHAN)** *short* Gain of each channel at each frequency. *Unit = dB.*
- **RX_agc_min(eN_TOMO)** *short* Minimum automatic gain control value. *Unit = dB.*
- **RX_agc_max(eN_TOMO)** *short* Maximum automatic gain control value. *Unit = dB.*
- **RX_agc_step(eN_TOMO)** *float* Step for automatic gain control setting. *Unit = dB.*
- **RX_delay(eN_TOMO, eN_FREQ)** *float* Delay between the SYNC pulse and the first acquired sample at each frequency. *Unit = s.*

- **RX_offset_acq(eN_TOMO, eN_FREQ, eN_CHAN)** *float* Delays between the first samples of each channel. This is relevant when the channels are not simultaneously sampled. *Unit = s.*
- **RX_cons_awake(eN_TOMO)** *float* Consumption of the receiver module in run mode. *Unit = Ah.*
- **RX_arm(eN_TOMO)** *short* Time needed by the controller to initiate a reception. *Unit = s.*
- **RX_run(eN_TOMO)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the reception. *Unit = s.*
- **RX_end(eN_TOMO)** *short* Time needed by the controller to terminate a reception and get back the receiver status. *Unit = s.*
- **RX_n(eN_TOMO)** *short* Number of receptions done per instrument during the experiment.
- **RX_date(eN_TOMO, eN_RX)** *int* Reception dates. Those are expressed in seconds since the experiment reference date. *Unit = s.*
- **RX_sail(eN_TOMO, eN_RX)** *byte* Receiver Sail status.
- **RX_carrier(eN_TOMO, eN_RX)** *short* Programmed reception carrier frequency. *Unit = Hz.*
- **RX_seq(eN_TOMO, eN_RX)** *short* Number of programmed sequences/reception.
- **RX_samples(eN_TOMO, eN_RX)** *int* Number of samples/reception.
- **RX_pointer(eN_TOMO, eN_RX)** *int* Pointer for reception.
- **RX_pres(eN_TOMO, eN_RX)** *short* Raw external pressure at reception time. *Unit = arbitrary units.*
- **RX_agc_gain(eN_TOMO, eN_CHAN, eN_RX)** *short* AGC gain setting. The gain in dB is obtained by multiplying this value with the number in **RX_agc_step**. *Unit = arbitrary units.*
- **RX_agc_rms(eN_TOMO, eN_CHAN, eN_RX)** *short* Measured RMS noise with AGC settings. Absolute value can be determined with the global gain of the acquisition. *Unit = arbitrary units.*

Receiving array

- **AR_id(eN_TOMO, N_NAME)** *char* Array identifier.
- **AR_type(eN_TOMO, N_STRING)** *char* Array type.
- **AR_channel(eN_TOMO)** *int* Number of channels.
- **AR_length(eN_TOMO)** *int* Array length. *Unit = m.*
- **AR_offset_depth(eN_TOMO)** *float* Offset of depth for the array center relatively to the pressure sensor position. *Unit = m.*
- **AR_geom(eN_TOMO, eN_CHAN)** *float* Position of acoustic center of each channel relatively to the array connector. *Unit = m.*
- **AR_type_hydro(eN_TOMO, N_STRING)** *char* Type of hydrophones.
- **AR_sh(eN_TOMO, eN_CHAN)** *short* Channel sensitivity. *Unit = dB re 1V/μPa.*

Navigator

- **NV_id(eN_TOMO, N_NAME)** *char* Navigator identifier.
- **NV_ver_soft(eN_TOMO, N_STRING)** *char* Navigator software version.
- **NV_freq_ping(eN_TOMO)** *short* Navigator ping frequency. *Unit = Hz.*
- **NV_time_ping(eN_TOMO)** *float* Navigator ping duration. *Unit = s.*
- **NV_counting(eN_TOMO)** *byte* 1 if the measurement starts at the pulse end.
- **NV_delay_ping(eN_TOMO)** *float* Navigator ping delay between SYNC pulse and effective ping. *Unit = s.*
- **NV_blank_ping(eN_TOMO)** *float* Blanking imposed after the emitted ping. *Unit = s.*
- **NV_freq_rcv(eN_TOMO, iN_BEACON))** *short* Navigator received frequencies. *Unit = Hz.*

- **NV_delay_rcv(eN_TOMO, iN_BEACON)** *float* Navigator delay for each received frequency. *Unit = s.*
- **NV_res_rcv(eN_TOMO)** *float* Navigator resolution. *Unit = s.*
- **NV_offset_depth(eN_TOMO)** *short* Correction of depth to set the navigation transducer at the depth of the pressure sensor. Negative value if the transducer is above the pressure sensor. *Unit = m.*
- **NV_cons_awake(eN_TOMO)** *float* Consumption of the navigator module in run mode. *Unit = Ah.*
- **NV_arm(eN_TOMO)** *short* Time needed by the controller to initiate a navigation. *Unit = s.*
- **NV_run(eN_TOMO)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the navigation. *Unit = s.*
- **NV_end(eN_TOMO)** *short* Time needed by the controller to terminate a navigation and get back the navigator status. *Unit = s.*
- **NV_n(eN_TOMO)** *int* Number of navigation per instrument during the experiment.
- **NV_date(eN_TOMO, eN_NV)** *int* Navigation ping dates during the experiment. Those are expressed in seconds since the experiment reference date. *Unit = s.*
- **NV_travel_time(eN_TOMO, iN_BEACON, eN_PING, eN_NV)** *float* Raw travel times measured during the experiment. *Unit = s.*
- **NV_sail(eN_TOMO, eN_NV)** *byte* Navigator Sail status.
- **NV_int(eN_TOMO, eN_NV)** *byte* Number of interrogation at each navigation.
- **NV_agc(eN_TOMO, eN_NV)** *byte* Navigator AGC. *Unit = arbitrary units.*
- **NV_pres(eN_TOMO, eN_NV)** *short* Raw external pressure at navigation time. *Unit = arbitrary units.*

Navigation transponders

- **NB_id(eN_TOMO, iN_BEACON, N_NAME)** *char* Acoustic transponder identifier. Set to *exp.* when expandable ones are used.
- **NB_type(eN_TOMO, iN_BEACON, N_STRING)** *char* Acoustic transponder type of the deployed transponders.
- **NB_sernum(eN_TOMO, iN_BEACON, N_STRING)** *char* Acoustic transponder serial number of the deployed transponders.
- **NB_inter_freq(eN_TOMO)** *short* Interrogation frequency of the transponders. *Unit = Hz.*
- **NB_resp_freq(eN_TOMO, iN_BEACON)** *short* Frequency of each transponder. *Unit = Hz.*
- **NB_resp_delay(eN_TOMO, iN_BEACON)** *float* Internal delay of each transponder. *Unit = s.*

Pressure and temperature module

- **PT_id(eN_TOMO, N_NAME)** *char* Pression and temperature measurement module identifier.
- **PT_Ptype(eN_TOMO, N_STRING)** *char* Pressure sensor type.
- **PT_Psernum(eN_TOMO, N_STRING)** *char* Pressure sensor serial number.
- **PT_Prange(eN_TOMO, TWO)** Pressure sensor range. *Unit = db* $= 10^4$ *Pa.*
- **PT_Ttype(eN_TOMO, N_STRING)** *char* Temperature sensor type.
- **PT_Tsernum(eN_TOMO, N_STRING)** *char* Temperature sensor serial number.
- **PT_Trange(eN_TOMO, TWO)** *int* Temperature sensor range. *Unit* $= {}^oC$.
- **PT_cons_awake(eN_TOMO)** *float* Consumption of the pressure and temperature measurement module in run mode. *Unit = Ah.*
- **PT_arm(eN_TOMO)** *short* Time needed by the controller to prepare a measure. *Unit = s.*

- **PT_run(eN_TOMO)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the measure. $Unit = s$.
- **PT_end(eN_TOMO)** *short* Time needed by the controller to cleanly terminate the measure and retrieve the status. $Unit = s$.
- **PT_dates_cal_P(eN_TOMO, eN_CAL, N_DATE_TIME)** *char* Pressure calibration dates. $Unit = s$.
- **PT_cal_Pref(eN_TOMO, eN_CAL, eN_VALP)** *float* Pressure reference. $Unit = db = 10^4\ Pa$.
- **PT_cal_Praw(eN_TOMO, eN_CAL, eN_VALP)** *float* Raw sensor outputs calibration values.
- **PT_conv_Pcoefs(eN_TOMO, eN_CAL, iN_COEF)** *float* Pressure conversion coefficients computed from the calibration points of the experiment.
- **PT_drift_Pcoefs(eN_TOMO, eN_CAL, iN_COEF)** *float* Pressure time drift coefficients.
- **PT_Perror(eN_TOMO, eN_CAL)** *float* Pressure sensor measurement error. $Unit = db = 10^4\ Pa$.
- **PT_dates_cal_T(eN_TOMO, eN_CAL, N_DATE_TIME)** *char* Temperature calibration dates. $Unit = s$.
- **PT_cal_Tref(eN_TOMO, eN_CAL, eN_VALT)** *float* Temperature reference. $Unit = {}^oC$.
- **PT_cal_Traw(eN_TOMO, eN_CAL, eN_VALT)** *float* Raw sensor outputs calibration values.
- **PT_conv_Tcoefs(eN_TOMO, eN_CAL, iN_COEF)** *float* Temperature conversion coefficients computed from the calibration points of the experiment.
- **PT_drift_Tcoefs(eN_TOMO, eN_CAL, iN_COEF)** *float* Temperature time drift coefficients.
- **PT_Terror(eN_TOMO, eN_CAL)** *float* Temperature sensor measurement error. $Unit = {}^oC$.

- **PT_n(eN_TOMO)** *int* Number of environmental measurements during the experiment.
- **PT_date(eN_TOMO, eN_PT)** *int* P and T measurement dates.Those are expressed in seconds since the experiment reference date. *Unit = s.*
- **PT_Pvalues(eN_TOMO, TWO, eN_PT)** *short* Raw and converted external P values as delivered by the instrument. Converted values are expressed in decibar.
- **PT_Tvalues(eN_TOMO, TWO, eN_PT)** *short* Raw and converted external T values as delivered by the instrument. Converted values are expressed im centidegree Celsius.

Storage unit

- **ST_id(eN_TOMO, N_NAME)** *char* Storage unit identifier
- **ST_type(eN_TOMO, N_STRING)** *char* Storage unit type.
- **ST_sernum(eN_TOMO, N_STRING)** *char* Storage unit serial number.
- **ST_capacity(eN_TOMO)** *short* Storage unit capacity. *Unit = Mb.*
- **ST_cons_awake(eN_TOMO)** *float* Consumption of the storage module in run mode. *Unit = Ah.*
- **ST_arm(eN_TOMO)** *short* Time needed by the controller to initiate a storage. *Unit = s.*
- **ST_run(eN_TOMO)** *short* Time needed by the unit to perform the storage. *Unit = s.*
- **ST_end(eN_TOMO)** *short* Time needed by the controller to terminate a storage and get back the status. *Unit = s.*
- **ST_n(eN_TOMO)** *short* Number of storages during the experiment.
- **ST_status(eN_TOMO, eN_ST)** *int* Storage status acquired during the experiment.

3 Experiment configuration file

This file is described by the **e_model.cdl** model and contains all the geometric and geographic characteristics of the experiment. In particular, this file contains :

- the geometry of the experiment,
- the local bathymetry around the mooring sites,
- the geometry of the long baseline positionning systems,
- the connection with the ocean data,
- the data relative to the acoustic survey for the transponder positions,
- the mooring motion data as estimated from the processed navigator data,
- the calibrated pressure and temperature data,
- the clock drift corrections.

A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_MOOR_POS** for example, while all experiment related dimensions are prefixed with **e** like **eN_TOMO**.
The naming convention for the experiment file is **e_ccccc.nc** where

- **e_** is the reserved prefix for this file type.
- **ccccc** One or more alphanumeric characters to name the experiment and/or to discriminate between different versions of the file (example : **i_thetis2_a** for Thetis 2 version a).

3.1 Dimensions

- **N_MOOR_NAME** A mooring must exactly have this number of characters in its name.
- **N_PAIR_NAME** A pair must exactly have this number of characters in its name. This number is exactly (N_MOOR_NAME * 2).

- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_STRING** Length of a short string.
- **N_LSTR** Length of a long string.
- **N_WAY** Fixed to 2 to describe each way of a pair.
- **N_PROC** Number of possible processes to compute corrections.
- **N_CORR** Number of possible corrections : mooring motions or clock drift.
- **iN_BEACON** Maximum number of acoustic transponders interrogated by a navigator.
- **eN_TOMO** Number of deployed tomographic instruments. It must appear one mooring for each tomographic instrument even if 2 instruments are deployed on the same mooring line.
- **eN_PAIR** Maximum number of pairs, exactly $(N\_TOMO * N\_TOMO - 1)/2$;
- **eN_ZLEV** Maximum number of points to describe the harmonic sound speed profile at transponder locations.
- **eN_BLAT** Maximum number of latitude points used to describe the local bathymetry around each mooring site.
- **eN_BLON** Maximum number of longitude points used to describe the local bathymetry around each mooring site.
- **eN_INT** Maximum number of acoustic interrogations from ship to transponder to realize the transponder positionning survey.
- **eN_MOOR_POS** Maximum number of points used to describe the mooring motions.
- **eN_MOOR_PRES** Maximum number of calibrated and corrected temperature and pressure measurements.
- **eN_TIME** Maximum number of points to build the time series of moorings motions

- **eN_CLOCK_DRIFT** Maximum number of points to build clocks drifts corrections.
- **eN_SCIENTIST** Maximum number of chief scientists involved in the experiment.

3.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of the last update of the file.
- **comments(N_LSTR)** *char* Comments on the file updates, file contents ...
- **Experiment_name(N_STRING)** *char* The name of the experiment.
- **Project_name(N_STRING)** *char* The name of the project.
- **Experiment_description(N_LSTR)** *char* A summary of the project and experiment.
- **Experiment_histor(N_LSTR)** *char* A short history of the experiment.
- **Chief_scientist(eN_SCIENTIST, N_STRING)** *char* Names, institutions of the chief scientists involved in the project/experiment.
- **Data_manager(eN_SCIENTIST, N_STRING)** *char* Names, institutions of the persons in charge of the data set.
- **Reference_date_time(N_DATE_TIME)** *char* Date and hour chosen as a reference for the experiment. In the instrument, time is usually counted in elapsed seconds or days from a reference point. This reference time is unique for an experiment. The format is DD/MM/YYYY HH:MM:SS where hours are given in a 24-hour format.
- **Start_date(N_DATE_TIME)** *char* Date and time of the beginning of the experiment. The format is the output of the **datestr(d, 0)** command of matlab : DD-MMM-YYYY HH:MN:SS where hours are in a 24-hour format.

- **End_date(N_DATE_TIME)** *char* Date and time of the end of the experiment. The format is the output of the **datestr(d, 0)** command of matlab : DD-MMM-YYYY HH:MN:SS where hours are in a 24-hour format.
- **Geographical_area(N_STRING)** *char* A short description of the experiment area.
- **South_latitude** *double* South limit of the experiment. This is expressed in decimal degree with the convention of positive to the north. *Unit = degree, Range = [-90, 90].*
- **North_latitude** *double* North limit of the experiment. *Unit = degree, Range = [-90, 90].*
- **West_longitude** *double* West limit of the experiment. The convention is positive to the east. *Unit = degree, Range = [-180, 180].*
- **East_longitude** *double* East limit of the experiment. *Unit = degree, Range = [-180, 180].*
- **Ellipsoid(N_STRING)** *char* Name of the reference ellipsoid. Default is WGS84.
- **Ellipsoid_Mj_axis** *double* Reference ellipsoid major axis. *Unit = m.*
- **Ellipsoid_mn_axis** *double* Reference ellipsoid minor axis. *Unit = m.*
- **Ocean_data_file(N_STRING)** *char* Name of the default ocean file used for the experiment even if other files can be used. The last used ocean data file is named in the `tomolab.ini` file.
- **Ocean_data_descr(N_LSTR)** *char* Description of the default ocean data base.
- **Sound_absorption_algorithm(N_STRING)** *char* Name/reference of the algorithm used to compute the sound absorption for the experiment. Choice is either Thorp, Lovett or No absorption.
- **Sound_speed_algorithm(N_STRING)** *char* Name/reference of the algorithm used to compute the sound speed for the experiment. Choice is either Del Grosso or Chen-Millero algorithm.
- **Mean_sound_speed** *float* Mean sound speed used to design the experiment. *Unit = m/s, Range = [1400, 1600].*

- **A_ph** *float* Ocean pH dependant coefficient which is requested to compute the sound absorption.

- **Notes(N_LSTR)** *char* Any relevant comments on the mooring network.

Mooring information

- **M_names(eN_TOMO, N_MOOR_NAME)** *char* Mooring name. A different name must be associated with 2 instruments deployed on the same mooring line.

- **M_mooring_date(eN_TOMO, N_DATE_TIME)** *char* Date of mooring operations.

- **M_recovering_date(eN_TOMO, N_DATE_TIME)** *char* Date of recovering operations.

- **M_target_lat(eN_TOMO)** *double* This is the nominal latitude point of the mooring. *Unit = degree, Range = [-90, 90].*

- **M_target_lon(eN_TOMO)** *double* This is the nominal longitude point of the mooring. *Unit = degree, Range = [-180, 180].*

- **M_target_depth(eN_TOMO)** *float* This is the nominal instrument depth. *Unit = m, Range = [0, 2000].*

- **M_actual_lat(eN_TOMO)** *double* The best estimated/computed real latitude of the mooring. *Unit = degree, Range = [-90, 90].*

- **M_actual_lon(eN_TOMO)** *double* The best estimated/computed real longitude of the mooring. *Unit = degree, Range = [-180, 180].*

- **M_actual_depth(eN_TOMO)** *float* The best estimated/computed real depth of the mooring (suggested value the shallowest depth). *Unit = m, Range = [0, 2000].*

- **M_actual_notes(N_LSTR)** *char* Dates of and comments on the computations made to obtain the so-called _actual_values above

- **M_bottom_depth(eN_TOMO)** *float* Corrected bottom depth at the mooring point (computed with true harmonic sound speed). *Unit = m, Range = [0, 6000].*

- **M_harm_sound_speed(eN_TOMO)** *float* Harmonic sound speed at the mooring point from surface to bottom. Used to compute the corrected depth at the mooring point. *Unit = m/s, Range = [1400, 1600].*

- **M_rho_g(eN_TOMO)** *float* Mean density above the instrument times gravity for pressure to depth conversions at each mooring position between pressure sensor and surface. *Unit* = $kg/m^2/s^2$, *Range = [9800, 11000].*

- **M_ambient_noise(eN_TOMO)** *short* Ambient noise level at each mooring position. Used to predict rough estimate of signal-to-noise ratio during the experiment design. *Unit* = $dB/1\mu Pa/\sqrt{Hz}$, *Range = [50, 100].*

- **M_ship_positionning(eN_TOMO, N_STRING)** *char* Comments on ship positionning system during mooring operations.

- **M_bathy_grid_lat(eN_TOMO, eN_BLAT)** *double* Latitudes for the bathymetric grid around each mooring site. This is the place to possibly store the data acquired during the bathymetric survey before each deployment. *Unit = degree, Range = [-90, 90].*

- **M_bathy_grid_lon(eN_TOMO, eN_BLON)** *double* Longitudes for the bathymetric grid around each mooring site. *Unit = degree, Range = [-180, 180].*

- **M_bathy_grid_depth(eN_TOMO, eN_BLAT, eN_BLON)** ıfloat Corrected depth values acquired during the bathymetric survey. *Unit = m, Range = [0, 10000].*

transponders information

- **B_bottom(eN_TOMO,iN_BEACON)** *float* Corrected depth of the bottom on the points where the transponders are deployed. *Unit = m, Range = [0, 6000].*

- **B_harm_sound_speed(eN_TOMO, iN_BEACON, eN_ZLEV)** *float* Harmonic sound speed profile at the transponder deploiement point. *Unit = m/s, Range = [1400, 1600].*

- **B_hssp_z(eN_ZLEV)** *float* Levels for harmonic sound speed at transponder position. *Unit = m, Range = [0, 6000].*

- **B_target_lat(eN_TOMO, iN_BEACON)** *double* The nominal latitude of each transponder. *Unit = degree, Range = [-90, 90].*
- **B_target_lon(eN_TOMO, iN_BEACON)** *double* The nominal longitude of each transponder. *Unit = degree, Range = [-180, 180].*
- **B_target_depth(eN_TOMO, iN_BEACON)** *float* The nominal depth of each transponder. *Unit = m, Range = [0, 6000].*
- **B_actual_lat(eN_TOMO, iN_BEACON)** *double* The best estimated real latitude of each transponder. These numbers are computed from the transponder survey data. *Unit = degree, Range = [-90, 90].*
- **B_actual_lat_err(eN_TOMO, iN_BEACON)** *float* The associated error. *Unit = m, Range = [0, 1000].*
- **B_actual_lon(eN_TOMO, iN_BEACON)** *double* The best estimated real longitude of each transponder. These numbers are computed from the transponder survey data. *Unit = degree, Range = [-180, 180].*
- **B_actual_lon_err(eN_TOMO, iN_BEACON)** *float* The associated error. *Unit = m, Range = [0, 1000].*
- **B_actual_depth(eN_TOMO, iN_BEACON)** *float* The best estimated real depth of each transponder. These numbers are computed from the transponder survey data. *Unit = m, Range = [0, 6000].*
- **B_actual_depth_err(eN_TOMO, iN_BEACON)** *float* The associated error. *Unit = m, Range = [0, 100].*

transponder survey information

- **BS_dates(eN_TOMO, eN_INT)** *int* Dates of transponders survey with ship. The dates are expressed in elapsed seconds since the reference date. *Unit = s.*
- **BS_round_trip_time(eN_TOMO, iN_BEACON, eN_INT)** *float* Round trip travel times from ship to transponders. *Unit = s, Range = [0, 15].*
- **BS_ship_transducer_offset_X(eN_TOMO)** *float* Ship mounted interrogation transducer X offset relative to the reference of the acquired GPS data. *Unit = m.*

- **BS_ship_transducer_offset_Y(eN_TOMO)** *float* Ship mounted interrogation transducer Y offset relative to the reference of the acquired GPS data. *Unit = m.*

- **BS_ship_transducer_offset_Z(eN_TOMO)** *float* Ship mounted interrogation transducer Z offset relative to the reference of the acquired GPS data. *Unit = m.*

- **BS_ship_transducer_lat(eN_TOMO, eN_INT)** *double* Ship mounted interrogation transducer latitude at interrogation dates. *Unit = degree, Range = [-90, 90].*

- **BS_ship_trans_lat_err(eN_TOMO, eN_INT)** *float* Error on ship transducer latitude. *Unit = m, Range = [0, 1000].*

- **BS_ship_transducer_lon(eN_TOMO, eN_INT)** *double* Ship mounted interrogation transducer longitude at interrogation dates. *Unit = degree, Range = [-180, 180].*

- **BS_ship_trans_lon_err(eN_TOMO, eN_INT)** *float* Error on ship transducer longitude. *Unit = m, Range = [0, 1000].*

- **BS_ship_transducer_depth(eN_TOMO, eN_INT)** *float* Ship mounted interrogation transducer depth at interrogation dates. *Unit = m, Range = [0, 6000].*

- **BS_ship_trans_depth_err(eN_TOMO, eN_INT)** *float* Error on ship transducer depth. *Unit = m, Range = [0, 100].*

- **BS_ship_nav_error(eN_TOMO, eN_INT)** *float* Error associated with the navigation quality. *Unit = m.*

Mooring navigation information

- **MI_clean_descr(eN_TOMO, N_STRING, N_PROC)** *char* Description and date of the method used to clean the navigation data or to ccompute the clock drift.

- **MI_dates(eN_TOMO, eN_MOOR_POS)** *int* Dates of the instrument positions used to describe the mooring motions. The dates are expressed in elapsed seconds since the reference date. *Unit = s.*

- **MI_ntime_beacon(eN_TOMO, iN_BEACON, eN_MOOR_POS)** *short* Number of navigator data used to compute the travel times.

Each navigation send a successive number of pings to obtain a set of redondant data. The final travel time memorized in the variable **MI_ttime_beacon** is the mean of these succesive replies once filtered out the outliers. This number is in some way an indicator of the navigation travel times quality.

- **MI_ttime_beacon(eN_TOMO, iN_BEACON, eN_MOOR_POS)** *float* Navigator corrected travel times used to compute the mooring motions. *Unit = s, Range = [0, 6.5535].*
- **MI_latitudes(eN_TOMO, eN_MOOR_POS)** *double* Computed latitudes of the instrument motions as estimated from the navigation program. *Unit = degree, Range = [-90, 90].*
- **MI_lat_err(eN_TOMO, eN_MOOR_POS)** *float* Error on the estimated latitudes as a result of the navigation program. *Unit = m, Range = [0, 1000].*
- **MI_longitudes(eN_TOMO, eN_MOOR_POS)** *double* Computed longitudes of the instrument motions as estimated from the navigation program. *Unit = degree, Range = [-180, 180].*
- **MI_lon_err(eN_TOMO, eN_MOOR_POS)** *float* Error on the estimated longitudes as a result of the navigation program. *Unit = m, Range = [0, 1000].*
- **MI_depth(eN_TOMO, eN_MOOR_POS)** *float* Computed depths of the instrument motions as estimated from the navigation program. *Unit = m, Range = [0, 2000].*
- **MI_depth_err(eN_TOMO, eN_MOOR_POS)** *float* Error on the estimated depths as a result of the navigation program. *Unit = m, Range = [0, 20].*
- **MI_clock_dates(eN_TOMO, eN_CLOCK_DRIFT)** *int* Dates of clock drift corrections expressed in seconds since the reference date. *Unit = s*
- **MI_clock_drift(eN_TOMO, eN_CLOCK_DRIFT)** *float* Clock drift corrections applied at the same time as the mooring motions corrections. *Unit = s.*
- **MI_clock_drift_descr(eN_TOMO, eN_STRING)** *char* Description of the clock drift corrections applied.

Pressure and temperature information

- **PT_dates(eN_TOMO, eN_MOOR_PRES)** *int* Dates of pressure and temperature measurements. The dates are expressed in elapsed seconds since the reference date. *Unit = s.*

- **PT_Pc(eN_TOMO, eN_MOOR_PRES)** *float* Calibrated and corrected pressure. *Unit = db =* 10^4*Pa, Range = [0, 2500].*

- **PT_Pc_err(eN_TOMO)** *float* Error on pressure in decibars. *Unit = db =* 10^4*Pa, Range = [0, 10].*

- **PT_Tc(eN_TOMO, eN_MOOR_PRES)** *float* Calibrated and corrected temperature. *Unit =* ^oC*, Range = [-5, 30].*

- **PT_Tc_err(eN_TOMO)** *float* Error on temperature. *Unit =* ^oC*, Range = [0, 1].*

Pair information

- **Pair_names(eN_PAIR, N_PAIR_NAME)** *char* Names of the pairs. The name is built from the mooring names. For example, moorings M1 and M2 result in the pair named M1M2.

- **Pair_reciprocity(eN_PAIR)** *short* Pair reciprocity flag with the following convention :

  0 No data for the pair

  1 data for the way M1 to M2

  -1 data for the way M2 to M1

  2 data for both ways.

- **Pair_range(eN_PAIR)** *float* Range between the instruments pair. All the range should be computed with the matlab routine `sodano.m` using the _actual_positions. *Unit = m, Range = [0, 5000000].*

- **Pair_mean_ss(eN_PAIR)** *float* A mean sound speed values for the pair and used to compute the propagation travel times and to convert change in pair range due to mooring motion into travel time delay. *Unit = m/s, Range = [1400, 1600].*

- **Pair_safety(eN_PAIR, N_WAY)** *float* Safety delay added to the pair propagation times to obtain the **Pair_delay**. *Unit = s, Range = [0, 20].*

- **Pair_delay(eN_PAIR, N_WAY)** *short* Instrument programmed delay for the listening window at the receiver. *Unit = s, Range = [0, 3600].*
- **Pair_ttcor_dist(eN_PAIR, N_WAY)** *float* Travel times corrections accounting for absolute distance uncertainties and possible instrument delays. *Unit = s.*
- **Pair_cor_descr(eN_PAIR, N_STRING, N_WAY, N_CORR)** *char* Description and date of the applied corrections (1=mooring motions, 2=clock drift).
- **Pair_dates_cor(eN_PAIR, eN_TIME, N_WAY)** *float* Time vectors containing the dates at which the moorings motions and clocks drifts corrections must be applied. The dates are always referenced to the reception times. Transmission time is computed as reception time minus propagation time. *Unit = s.*
- **Pair_mm(eN_PAIR, eN_TIME, N_WAY)** *float* Travel times corrections due to the moorings motions. *Unit = s, Range = [-5, 5].*
- **Pair_cd(eN_PAIR, eN_TIME, N_WAY)** *float* Travel times corrections due to clocks drifts. *Unit = s, Range = [-60, 60].*

4 Ocean data file

This file is described by the **o_model.cdl** model and contains data regarding the *a priori* knowledge on the ocean volume under investigation during the experiment. This *a priori* knowledge is described in two complementary parts : sections or modes. Both of them can be established for different time periods. Sections are described in terms of range dependent sound speed. The range averaged sound speed is a requisite parameter. Additionnal range averaged parameters can be provided as temperature, salinity, potential temperature, density anomaly, ...
Mode sets include the following required variables

- a mean sound speed,
- a temperature profile,
- a conversion function from sound speed anomalies to temperature anomalies and corresponding error

and can be extended with :

- additional mean parameter profiles,
- a set of vertical sound speed anomaly modes associated with the mean sound speed profile,
- information relative to each mode (mode eigenvalue or rms, percentage of explained variance for EOFs, Rossby radius for dynamic modes,...)

Almost all the dimensions of the file are experiment dependant.
The naming convention for the ocean data file is **o_cccccn.nc** where

- **o_** is the reserved prefix for this file type.
- **ccccc** One or more alphanumeric characters to name the experiment and/or to discriminate between different versions of the file (example : **i_thetis2_a** for Thetis 2 version a).

4.1 Dimensions

- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.

- **N_STRING** Number of characters in a short string.
- **N_LSTR** Length of a long string.
- **N_PAIR_NAME** A pair must exactly have this number of characters in its name. This number is exactly (N_MOOR_NAME * 2).
- **TWO** To fix a 2-dimension to arrays.
- **eN_SECT** Number of sections described. If this dimension need to be changed, the wile will be rewritten.
- **eN_NZ** Maximum number of levels to discretize the sections and modes in depth.
- **eN_DX** Maximum number of points to discretize the sections in range
- **eN_BATH** Maximum number of points to discretize the bathymetry along section
- **eN_PER** Maximum number of periods for which the sections are described. This could be a time period like season or month or whatever.
- **eN_PAR_SECT** Maximum number of mean parameters in addition to sound speed provided along the sections. Minimum can be 0 but recommended minimum is 3 : $T, \delta T \rightarrow \delta C+$ error.
- **eN_MODE** Maximum number of modes used to describe the area.
- **eN_SET** Maximum number of mode sets used to describe the area. For example, the sets could be the same modes but computed at different time periods and/or different modes. If this dimension need to be changed, the file will be rewritten.
- **eN_PAR_MODE** Maximum number of mean parameters in addition to sound speed provided with each mode set. Minimum can be 0 but recommended minimum is 3 : $T, \delta T \rightarrow \delta C+$ error.
- **eN_INFO** Maximum number of extra information provided with a mode.
- **eN_CONVERSION** Maximum number of elements of the conversion relation.

4.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of the last update of the file.
- **comments(N_LSTR)** *char* Comments on the file content...
- **Reference_date_time(N_DATE_TIME)** *char* Reference date for the section or mode periods.
- **Ocean_db_filename(N_STRING)** *char* File name of the ocean database considered.
- **Ocean_db_descr(N_LSTR)** *char* Description of the Ocean database. It corresponds to the global attirbute of the file.
- **Z_level(eN_NZ)** *float* Levels defined for the vertical discretisation. The *positive* attribute is defined to indicate the sign convention. *Unit = m, Positive = down, Range = [0, 15000].*

Section description

- **N_bath_act(eN_SECT)** *short* Actual number of points for each section bathymetry. Must be $\leq$ eN_BATH.
- **N_per_act(eN_SECT)** *short* Actual number of periods per section. Must be $\leq$ eN_PER.
- **N_dx_act(eN_SECT)** *short* Actual number of points per section for range discretization. Must be $\leq$ eN_DX.
- **Sect_descr(eN_SECT, N_LSTR)** *char* Description of section, data base used, sound speed algorithm, ...
- **Sect_flag(eN_SECT)** *short* -1 = synthetic; 0 = real, range dependent; 1 = real, range independent at Mooring 1; 2 = real, range independent at Mooring 2; 3 = real, range independent, Mean. *Default = 0.*
- **Sect_pair_name(eN_SECT, N_PAIR_NAME)** *char* Name of the pair associated with the section.

- **Sect_center(eN_SECT)** *short* Define the central meridian of longitude along section. If this value is 0, the longitudes are given in the range [-180, +180]. If the value is 180, the longitudes are given in the range [0, +360]. *Unit = degree, Default = 0.*
- **Sect_lat(eN_SECT, eN_DX)** *double* Latitudes of the points along the section. Sec_lat(:, 1) and Sect_lat(:, N_dx_act(:)) are the mooring point latitudes. *Unit = degree, Range = [-90, 90].*
- **Sect_lon(eN_SECT, eN_DX)** *double* Longitudes of the points along the section. Sec_lon(:, 1) and Sect_lon(:, N_dx_act(:)) are the mooring point longitudes. *Unit = degree, Range = [-180, 180].*
- **Sect_range(eN_SECT, eN_DX)** *float* Range along the sections according to the discretisation. *Unit = m.* The first value is 0 and the last one is the exact distance of the pair.
- **Sect_periods(eN_SECT, eN_PER, N_STRING)** *char* Description of the different periods considered for the sections.
- **Sect_periods_start(eN_SECT, eN_PER)** *float* Starting dates of periods referenced to **Reference_date_time**. *Unit = decimal days.*
- **Sect_periods_end(eN_SECT, eN_PER)** *float* Ending dates of periods referenced to **Reference_date_time**. *Unit = decimal days.*
- **Sect_bathy(eN_SECT, eN_BATH)** *float* Bathymetry along the sections according to the discretisation. *Unit = m, Range = [0, 10000].*
- **Sect_ssp(eN_SECT, eN_PER, eN_DX, eN_NZ)** *float* Sound speed profiles along the sections. *Unit = m/s, Range = [1450, 1550].*
- **Sect_mean_ssp(eN_SECT, eN_PER, eN_NZ)** *float* Mean sound speed profile along section. *Unit = m/s, Range = [1450, 1550].*
- **Sect_mean_par_descr(eN_PAR_SECT, N_STRING)** *char* Description and units of the mean parameters provided for the sections.
- **Sect_mean_par(eN_SECT, eN_PAR_SECT, eN_NZ)** *float* Mean parameters provided for the sections.
- **Index_mode(eN_SECT, eN_SET)** *short* Index to put in relation a set of vertical modes with a section.

Mode description

- **Mode_set_descr(eN_SET, N_LSTR)** *char* Description of the mode sets : period, data base, sound speed algorithm, ...
- **Mode_south_lat(eN_SET)** *double* South limit of the modes sets. *Unit = degree, Range = [-90, 90].*
- **Mode_north_lat(eN_SET)** *double* North limit of the modes sets. *Unit = degree, Range = [-90, 90].*
- **Mode_west_lon(eN_SET)** *double* West limit of the modes sets. *Unit = degree, Range = [-180, 180].*
- **Mode_east_lon(eN_SET)** *double* East limit of the modes sets. *Unit = degree, Range = [-180, 180].*
- **N_mode_act(eN_SET)** *short* Number of actual modes per set.
- **Mode_mean_ssp(eN_SET, eN_NZ)** *float* Mean sound speed for the mode set. *Unit = m/s, Range = [1450, 1550].*
- **Mode_mean_par_descr(eN_PAR_MOD, N_LSTR)** *char* Description of mean parameters (temperature, salinity, $\delta T \rightarrow \delta C$ conversion and error)
- **Mode_mean_par(eN_SET, eN_PAR_MOD, eN_NZ)** *float* Mean parameters associated with modes.
- **Mode_info_descr(eN_SET, eN_MODE, N_LSTR)** *char* Description and units of information relative to each mode.
- **Mode_info(eN_SET, eN_MODE, eN_INFO)** *float* Information relative to each mode (eigenvalue, Rossby radius, ...)
- **Mode_conversion_relation(eN_SET, eN_NZ, eN_CONVERSION)** *float* Elements of the conversion relation:
  - 1) Reference temperature,
  - 2) Reference sound speed
  - 3) Conversion factor c→t,
  - 4) temperature rms error;
- **Mode_rms_amplitude(eN_SET, eN_MODE)** *float* RMS amplitude (square rooted eigenvalue) of each mode.

- **Mode_C(eN_SET, eN_MODE, eN_NZ)** *float* sets of vertical sound speed modes.
- **N_per_mode(eN_SET)** *short* Actual number of periods per Mode-Set.
- **Mode_periods(eN_SET, eN_PER)** *float* Time of mode periods referenced to **Reference_date_time**. *Unit = days.*
- **Mode_amplitude_evolution(eN_SET, eN_PER, eN_MODE)** *float* Time Evolution of Mean RMS amplitude (square rooted eigenvalue) of each mode.

5 Acoustic data files

There are 2 file types associated with the acoustic data ; one is related to the direct problem, the other to the inverse problem.

5.1 Direct acoustic calculations

This file is described by the **a_model.cdl** model and contains data regarding the direct acoustic calculations for extracted sound-speed sections.
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_IR** for example, while all experiment related dimensions are prefixed with **e** like **eN_PER**. Dimensions for parametrization of acoustic calculations are prefixed with **a** like **aN_DXI**.
The naming convention for the data file is **a_xxxyyy[.ccc].nc** where

- **xxx** One or more alphanumeric characters for the 1rst mooring name,
- **yyy** Same number of alphanumeric characters for the 2nd mooring name,
- **[.ccc]** One or more optional alphanumeric characters for distinguishing different file versions. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

5.1.1 Dimensions

- **N_DATE_TIME** Number of characters for a date. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_OCEAN_NAME** Number of characters for ocean data filename
- **N_PAIR_NAME** Number of characters for pair names
- **N_STRING** Number of characters for a string
- **N_LSTR** Number of characters for long string, to describe special events for example.
- **iN_IR** Number of frequencies to describe source signal.
- **eN_PER** Number of periods covered (e.g. #season, #months,...).

- **eN_NZ** Number of original sound-speed depths in ocean data file.
- **eN_DX** Number of original sound-speed ranges in ocean data file
- **aN_DXI** umber of ranges used for range-dependent acoustic calculation.
- **aN_R_GEO** Cummulative number of points describing the eigenray geometry
- **aN_R_NUM** Number of launched rays.
- **aN_R_ENU** Number of eigenrays.
- **aN_R_NZTL** Number of depths used for ray-theoretic transmission loss calculations.
- **aN_M_NUM** Number of propagating normal modes used.
- **aN_M_TIM** Number of points for arrival pattern description
- **aN_M_AT** Number of peak arrivals.
- **aN_M_NRD** Number of receiver depths.
- **aN_M_NZ** Number of depths for mode shape description.
- **aN_M_NF** Number of frequencies for normal-mode calculations.

5.1.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of last update of this file.
- **comments(N_LSTR)** *char* Comments on file contents.

Identification header : general

- **Experiment_name(N_STRING)** *char* Experiment identifier (imported from ocean data file).
- **AC_oc_name(N_OCEAN_NAME)** *char* Name of input ocean data file.
- **AC_pair_name(N_PAIR_NAME)** *char* Mooring pair identifier.

- **AC_sect_index** *short* Index relating to the section dimension of the ocean data file, through which the section underlying the acoustic results can be identified in the ocean data file.

- **AC_sect_length** *float* Horizontal section length (range). *Unit = m.*

- **AC_sect_lat1** *double* The section is described by the geographical coordinates of its 2 endpoints, as in ocean data file. This variable is the latidude of 1st mooring in the pair name. *Unit = degree, Range = [-90, 90].*

- **AC_sect_lon1** *double* longitude of 1st mooring in the pair name. *Unit = degree, Range = [-180, 180].*

- **AC_sect_lat2** *double* latidude of 2nd mooring in the pair name. *Unit = degree, Range = [-90, 90].*

- **AC_sect_lon2** *double* longitude of 2nd mooring in the pair name. *Unit = degree, Range = [-180, 180].*

- **AC_sect_direction** *short* Flag denoting the direction of propagation between the two moorings in the pair name (1: from 2nd to 1st mooring, -1: from 1st to 2nd mooring)

- **AC_PER_ID** *short* ID of time period (e.g. season, month,...) used. . If ID has a value between 1 and eN_PER one period has been considered. If ID equals eN_PER+1 then all periods (each one separately) have been considered.

Identification header : sound speed section

- **Ocean_DB_name(N_STRING)** *char* Name of ocean data base the section was extracted from.

- **Sect_periods(eN_PER, N_STRING)** *char* Description/comments on time periods.

- **Reference_date_time(N_DATE_TIME)** *char* Reference date for Sect_periods, Mode_periods.

- **Sect_periods_start(eN_PER)** *float* Start of periods relative to the reference date. *Units = decimal days.*

- **Sect_periods_end(eN_PER)** *char* End of periods relative to the reference date. *Units = decimal days.*

- **Sound_speed_algorithm(N_STRING)** *char* Name of sound speed algorithm.
- **AC_water_depth** *float* Water depth used for acoustic calculations. *Unit = m, Range = [0, 6000].*
- **AC_ECC(eN_NZ)** *float* Earth-curvature correction applied for acoustic calculations. *Unit = m/s.*
- **AC_depth_oversample** *short* Number of additional points in depth (between any two successive original data points). 0: unchanged, 1: doubling, 2: tripling, etc...
- **AC_range_oversample** *short* Number of additional points in range (between any two successive original data points). 0: unchanged, 1: doubling, 2: tripling, etc...
- **AC_SSPinterp_method** *short* Type of depth interpolation (0:linear, 1:spline).
- **AC_Num_range_segments** *short* Number of range segments after interpolation/oversampling.
- **AC_range(aN_DXI)** *float* Ranges of the rightmost (closer to receiver) boundaries of the range segments (after sound-speed interpolation/oversampling). *Unit = m.*

Identification header : setup

- **AC_source_depth** *float* Source depth used for acoustic calculations. *Unit = m, Range = [0, 2000].*
- **AC_receiver_depth** *float* Receiver depth used for acoustic calculations. *Unit = m, Range = [0, 2000].*

Identification header : signal

- **AC_source_central_freq** *float* Source central frequency. *Unit = Hz.*
- **AC_source_freqs(iN_IR)** *float* Frequencies at which source level is described. *Unit = Hz.*
- **AC_source_amplitudes(iN_IR)** *float* Source signal level, delogarithmized (not dB).

- **AC_source_bandwidth** *float* Model source bandwidth BW used for acoustic calculations (Gaussian pulse: $e^{-4\pi(f-f_0)^2/BW^2}$, where $\pi = 3.14159$, f: frequency in Hz, f_0: central frequency in Hz).

Identification header : acoustic codes

- **AC_calc_type** *short* type of acoustic calculations (1:rays, 2: modes, 3: rays and modes).
- **AC_R_code_ID** *short* ID of acoustic ray code used (1: RAYTRACE).
- **AC_R_code_name(N_STRING)** *char* Name of acoustic ray code used.
- **AC_M_code_ID** *short* ID of acoustic normal-mode code used (1: Standard normal modes, 2: Fast normal modes).
- **AC_M_code_name(N_STRING)** *char* Name of acoustic normal-mode code used.
- **AC_R_angle_start** *float* Start launch angle (grazing) for ray calculations. *Unit = degree, Range = [-45 +45].*
- **AC_R_angle_end** *float* End launch angle (grazing) for ray calculations. *Unit = degree, Range = [-45 +45].*
- **AC_R_angle_step** *float* Launch-angle step for ray calculations. *Unit = degree.*
- **AC_M_number** *short* Number of propagating modes used.
- **AC_M_freq_min** *float* Minimum frequency for mode calculations. *Unit = Hz.*
- **AC_M_freq_max** *float* Maximum frequency for normal-mode calculations. *Unit = Hz.*
- **AC_M_freq_step** *float* Frequency step for normal-mode calculations. *Unit = Hz.*
- **AC_M_time_step** *float* Time step (resolution) for arrival-pattern calculations. *Unit = s.*

Acoustic calculation : ray results

- **AC_R_depth(eN_PER, aN_R_NUM)** *float* Depths of launched rays at the receiver range. *Unit = m.*
- **AC_R_time(eN_PER, aN_R_NUM)** *double* Arrival times of launched rays at the receiver range. *Unit = s.*
- **AC_R_ZTL(eN_PER,aN_R_NZTL)** *float* Depths at receiver range for transmission loss calculations. *Unit = m.*
- **AC_R_TL(eN_PER, aN_R_NZTL)** *float* Transmission loss for various depths at receiver range. *Unit = dB re 1m.*
- **AC_R_eigenray_filtered** *short* Flag for eigenray filtering, (1: only selected eigenrays are saved, 0: all calculated eigenrays are saved).
- **AC_R_eigenray_ID(eN_PER, aN_R_ENU)** *short* Ray identifier (+- number of turning points);
- **AC_R_eigenray_launch_angle(eN_PER, aN_R_ENU)** *float* Eigenray launch angle (grazing). *Unit = degree.*
- **AC_R_eigenray_arrival_angle(eN_PER, aN_R_ENU)** *float* Eigenray arrival angle (grazing). *Unit = degree.*
- **AC_R_eigenray_arrival_time(eN_PER, aN_R_ENU)** *double* Eigenray arrival time. *Unit = s.*
- **AC_R_eigenray_arrival_amplitude(eN_PER, aN_R_ENU)** *float* Eigenray arrival amplitude. *Unit = dB.*
- **AC_R_eigenray_dturn_depth(eN_PER, aN_R_ENU)** *float* Minimum down-turning depth. *Unit = m.*
- **AC_R_eigenray_uturn_depth(eN_PER, aN_ER_ENU)** *float* Maximum up-turning depth. *Unit = m.*
- **AC_R_eigenray_type(eN_PER, aN_ER_ENU)** *char* Type of eigenray (RR: refracted, SR: surface reflected and refracted, RB refracted and bottom reflected, SB: refracted as well as surface and bottom reflected).
- **AC_R_eigenray_x(eN_PER, aN_R_BILK_ERAYGEO)** *float* eigenray geometry range vector. *Unit = m.*

- **AC_R_eigenray_z(eN_PER, aN_R_BILK_ERAYGEO)** *float* eigenray geometry depth vector. *Unit = m.*

Acoustic calculation : mode results

- **AC_M_freqs(aN_M_NF)** *float* Vector of frequencies used for acoustic calculations. *Unit = Hz.*
- **AC_M_press_real(eN_PER, aN_M_NF)** *float* Real parts of the calculated complex pressure at the fixed receiver range and depth in the frequency domain.
- **AC_M_press_imag(eN_PER, aN_M_NF)** *float* Imaginary parts of the calculated complex pressure at the fixed receiver range and depth in the frequency domain.
- **AC_M_time(eN_PER, aN_M_TIM)** *double* Time vector for arrival pattern. *Unit = s.*
- **AC_M_press_amp(eN_PER, aN_M_TIM)** *float* Arrival pattern(s), i.e. pressure amplitude at the receiver in the time domain, for eN_PER period(s).
- **AC_M_arr_time(eN_PER, aN_M_AT)** *double* Peak arrival times. *Unit = s.*
- **AC_M_rec_depth(aN_M_NRD)** *float* Receiver depths (variable-depth results). *Unit = m.*
- **AC_M_press_real_VD(eN_PER, aN_M_NF, aN_M_NRD)** *float* Real parts of the calculated complex pressure in the frequency domain at receiver range for various receiver depths.
- **AC_M_press_imag_VD(eN_PER, aN_M_NF, aN_M_NRD)** *float* Imaginary parts of the calculated complex pressure in the frequency domain at receiver range for various receiver depths.
- **AC_M_press_amp_VD(eN_PER, aN_M_TIM, aN_M_NRD)** *float* Arrival pattern(s), i.e. pressure amplitude in the time domain, at receiver range for various receiver depths.
- **AC_M_arr_time_VD(eN_PER, aN_M_AT, aN_M_NRD)** *double* Peak arrival times for various receiver depths. *Unit = s.*

- **AC_M_velocity(eN_PER,aN_M_NUM,aN_M_NF,aN_DXI)** *float* Modal group velocity vs. frequency for all range segments. *Unit = m/s.*
- **AC_M_traveltime(eN_PER,aN_M_NUM,aN_M_NF)** *double* Modal group traveltimes (adiabatic) vs. frequency. *Unit = s.*
- **AC_M_depth(aN_M_NZ)** *float* Depth vector for the description of mode shapes. *Unit = m.*
- **AC_M_shape(eN_PER,aN_M_NUM,aN_M_NZ)** *float* Mode amplitude for variable depth.
- **AC_M_ZTL(eN_PER,aN_M_NZ)** *float* Depths at receiver range for transmission loss calculations. *Unit = m.*
- **AC_M_TL(eN_PER,aN_M_NZ)** *float* Transmission loss at receiver range for variable depth. *Unit = dB re 1m.*
- **AC_M_excitation(eN_PER,aN_M_NUM,aN_DXI)** *float* Mode excitation for all range segments.
- **AC_M_coupling(eN_PER,aN_M_NUM,aN_M_NUM,aN_DXI)** *float* Coupling coefficients at range-segment interfaces.

5.2 Inverse acoustic calculations

This file is described by the **z_model.cdl** model and contains data regarding the inversion-related acoustic calculations.
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_IR** for example, while all experiment related dimensions are prefixed with **e** like **eN_NZ**. Dimensions for parametrization of acoustic calculations are prefixed with **a** like **aN_BST**.
The naming convention for the data file is **z_xxxyyy[.ccc].nc** where

- **xxx** One or more alphanumeric characters for the 1rst mooring name,
- **yyy** Same number of alphanumeric characters for the 2nd mooring name,
- **[.ccc]** One or more optional alphanumeric characters for distinguishing different file versions. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

5.2.1 Dimensions

- **N_DATE_TIME** Number of characters for a date. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_OCEAN_NAME** Number of characters for ocean data filename
- **N_PAIR_NAME** Number of characters for pair names
- **N_STRING** Number of characters for a string
- **N_LSTR** Number of characters for long string, to describe special events for example.
- **iN_IR** Number of frequencies to describe source signal.
- **eN_NZ** Number of original sound-speed depths (from ocean file).
- **aN_R_NUM** Number of launched rays.
- **aN_R_ENU** Number of eigenrays.
- **aN_R_AT** Number of selected ray arrivals

- **aN_M_TIM** Number of points for arrival pattern description
- **aN_M_AA** Total number of peak arrivals
- **aN_M_AT** Number of selected peak arrivals.
- **aN_M_NF** Number of frequencies for acoustic calculations.
- **aN_BST** Number of background states.
- **aN_SSM** Number of used sound-speed modes.
- **aN_Manip** Number of ocean files used for modification of mean (reference) profile.

5.2.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of last update of this file.
- **comments(N_LSTR)** *char* Comments on file contents.

Identification header : general

- **Experiment_name(N_STRING)** *char* Experiment identifier (imported from ocean data file).
- **AC_oc_name(N_OCEAN_NAME)** *char* Name of input ocean data file.
- **AC_pair_name(N_PAIR_NAME)** *char* Mooring pair identifier.
- **AC_sect_length** *float* Horizontal section length (range). *Unit = m.*
- **AC_sect_lat1** *double* The section is described by the geographical coordinates of its 2 endpoints, as in ocean data file. This variable is the latidude of 1st mooring in the pair name. *Unit = degree, Range = [-90, 90].*
- **AC_sect_lon1** *double* longitude of 1st mooring in the pair name. *Unit = degree, Range = [-180, 180].*
- **AC_sect_lat2** *double* latidude of of 2nd mooring in the pair name. *Unit = degree, Range = [-90, 90].*

- **AC_sect_lon2** *double* longitude of of 2nd mooring in the pair name. *Unit = degree, Range = [-180, 180].*

- **AC_sect_direction** *short* Flag denoting the direction of propagation between the two moorings in the pair name (1: from 2nd to 1st mooring, -1: from 1st to 2nd mooring).

- **AC_SET_index** *short* Index (relating to the eN_SET dimension of the ocean data file) through which the sound speed mode set underlying the acoustic inversion-related results can be identified in the ocean data file.

- **AC_Manip_oc_name(N_OCEAN_NAME, aN_Manip)** *char* Names of ocean data files used for modification of mean (reference) profile.

- **AC_Manip_depth** *float* Manipulation depth for reference profile. *Unit = m.*

- **AC_Modif_flag(aN_Modif)** *short* Flag denoting the type of values(1: Section, 2: Mode set).

- **AC_Modif_versionID(aN_Modif)** *short* Index relating to the section dimension of the ocean data file.

- **AC_Modif_periodID(aN_Modif)** *short* ID of time period used.

- **AC_Modif_setID(aN_Modif)** *short* ID of Sound Speed Mode set.

- **AC_Modif_depth** *float* Adjustment depth for the modification of mean (reference) profile.

Identification header : sound speed section

- **Ocean_DB_name(N_STRING)** *char* Name of ocean data base the section was extracted from.

- **Sound_speed_algorithm(N_STRING)** *char* Name of sound-speed algorithm.

- **AC_water_depth** *float* Water depth used for acoustic calculations. *Unit = m, Range = [0, 6000].*

- **AC_ECC(eN_NZ)** *float* Earth-curvature correction applied for acoustic calculations. *Unit = m/s.*

- **AC_depth_oversample** *short* Number of additional points in depth (between successive original data points).

- **AC_theta1_back(aN_BST)** *float* Background states used for inversion-related calculations.

- **AC_dtheta(aN_SSM)** *float* Sound-speed mode amplitude perturbations used for finite difference calculation of influence coefficients.

Identification header : setup

- **AC_source_depth** *float* Source depth used for acoustic calculations. *Unit = m, Range = [0, 2000].*

- **AC_receiver_depth** *float* Receiver depth used for acoustic calculations. *Unit = m, Range = [0, 2000].*

Identification header : signal

- **AC_source_central_freq** *float* Source central frequency. *Unit = Hz.*

- **AC_source_freqs(iN_IR)** *float* Frequencies at which source level is described. *Unit = Hz.*

- **AC_source_amplitudes(iN_IR)** *float* Source signal level, delogarithmized (not dB).

- **AC_source_bandwidth** *float* Model source bandwidth BW used for acoustic calculations (Gaussian pulse: $e^{-4\pi(f-f_0)^2/BW^2}$, where $\pi = 3.14159$, f: frequency in Hz, f_0: central frequency in Hz).

Identification header : acoustic codes

- **AC_calc_type** *short* type of acoustic calculations (1:rays, 2: normal modes, 3: rays and normal modes).

- **AC_R_code_ID** *short* ID of acoustic ray code used (1: Raytrace / Finite differences, 2:Raytrace / Analytic).

- **AC_R_code_name(N_STRING)** *char* Name of acoustic ray code used.

- **AC_M_code_ID** *short* ID of acoustic normal-mode code used (1: Normal modes / Finite differences, 2: Fast normal modes / Finite differences, 3: Normal modes / Analytic).

- **AC_M_code_name(N_STRING)** *char* Name of acoustic normal-mode code used.
- **AC_R_angle_start** *float* Start launch angle (grazing) for ray calculations. *Unit = degree.*
- **AC_R_angle_end** *float* End launch angle (grazing) for ray calculation. *Unit = degree.*
- **AC_R_angle_step** *float* Launch-angle step for ray calculations. *Unit = degree.*
- **AC_M_number** *short* Number of propagating modes used.
- **AC_M_freq_min** *float* Minimum frequency for normal-mode calculations. *Unit = Hz.*
- **AC_M_freq_max** *float* Maximum frequency for normal-mode calculations. *Unit = Hz.*
- **AC_M_freq_step** *float* Frequency step for normal-mode calculations. *Unit = Hz.*
- **AC_M_time_step** *float* Time step (resolution) for arrival-pattern calculations. *Unit = s.*

Acoustic calculation : ray results

- **AC_R_eigenray_ID(aN_BST, aN_R_ENU)** *short* Ray identifier (+- number of turning points) for inversion-related ray runs;
- **AC_R_eigenray_back_arr_time(aN_BST, aN_R_ENU)** *double* Background ray arrival times. *Unit = s.*
- **AC_R_eigenray_back_arr_amp(aN_BST, aN_R_ENU)** *float* Background ray arrival amplitudes. *Unit = dB.*
- **AC_R_eigenray_pert_arr_time(aN_BST, aN_R_ENU, aN_SSM)** *double* Perturbed ray arrival time, in the direction of each sound-speed mode. *Unit = s.*
- **AC_R_eigenray_pert_arr_amp(aN_BST, aN_R_ENU, aN_SSM)** *float* Perturbed ray arrival amplitude, in the direction of each sound-speed mode. *Unit = dB.*

- **AC_R_eigenray_infl_coeff(aN_BST, aN_R_ENU, aN_SSM)** *float* Influence coefficients.
- **AC_R_back_arr_select(aN_BST, aN_R_AT)** *short* Index pointing to selected ray arrivals in AC_R_eigenray_ID.
- **AC_R_back_arr_time(aN_BST, aN_R_AT)** *double* Background arrival times corresponding to selected ray arrivals. *Unit = s.*
- **AC_R_infl_coeff(aN_BST, aN_R_AT, aN_SSM)** *float* Influence coefficients corresponding to selected ray arrivals.

Acoustic calculation : mode results

- **AC_M_freqs(aN_M_NF)** *float* Vector of frequencies used for inversion-related acoustic calculations. *Unit = Hz.*
- **AC_M_back_press_real(aN_BST, aN_M_NF)** *float* Real parts of the calculated complex pressure at each background state in the frequency domain.
- **AC_M_back_press_imag(aN_BST, aN_M_NF)** *float* Imaginary parts of the calculated complex pressure at each background state in the frequency domain.
- **AC_M_pert_press_real(aN_BST, aN_M_NF, aN_SSM)** *float* Real part of either the calculated perturbed complex pressure (finite-difference case), or the functional derivative of the complex pressure (analytic case) in the direction of each sound speed mode in the frequency domain.
- **AC_M_pert_press_imag(aN_BST, aN_M_NF, aN_SSM)** *float* Imaginary part of either the calculated perturbed complex pressure (finite-difference case), or the functional derivative of the complex pressure (analytic case) in the direction of each sound speed mode in the frequency domain.
- **AC_M_back_time(aN_M_TIM)** *double* Time vector for arrival pattern description. *Unit = s.*
- **AC_M_back_press_amp(aN_BST, aN_M_TIM)** *float* Background arrival pattern(s) - pressure amplitude(s) - in the time domain.

- **AC_M_pert_press_amp(aN_BST, aN_M_TIM, aN_SSM)** *float* Perturbed arrival pattern(s) in the direction of each sound-speed mode (finite-difference case) in the time domain.
- **AC_M_back_time_all(aN_BST, aN_M_AA)** *double* Background peak arrival times. *Unit = s.*
- **AC_M_infl_coeff_all(aN_BST, aN_M_AA, aN_SSM)** *float* Influence coefficients.
- **AC_M_back_arr_select(aN_BST, aN_M_AT)** *short* Index pointing to selected peak arrivals in AC_M_back_time_all.
- **AC_M_back_arr_time(aN_BST, aN_M_AT)** *double* Background arrival times corresponding to selected peak arrivals. *Unit = s.*
- **AC_M_infl_coeff(aN_BST, aN_M_AT, aN_SSM)** *float* Influence coefficients corresponding to selected peak arrivals.

6 Level-0 data file

This file is described by the **l0_model.cdl** model and contains the raw acoustic data as read from the instrument storage medium without any compression or processing. It is the reference level for the acoustic data. For that level there is no reciprocity and data are one way with the convention that the 1rst mooring in the pair name is the receiver.
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_CHAN** for example, while all experiment related dimensions are prefixed with **e** like **eN_REC**.
All the level-0 data files are grouped in a directory named **level0** and the naming convention for the data file is **xxxyyy.lv0** where

- **xxx** One or more alphanumeric characters for the receiver mooring name,
- **yyy** Same number of alphanumeric characters for the source mooring name,
- **.lv0** the reserved extension for the level

6.1 Dimensions

- **N_MOOR_NAME** A mooring must exactly have this number of characters in its name.
- **N_PAIR_NAME** A pair must exactly have this number of characters in its name. This number is exactly (N_MOOR_NAME * 2).
- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_LSTR** Length of a long string.
- **N_SZ** This dimension is computed during the decoding process and is set accordingly.
- **TWO** Fixed to 2.
- **iN_CHAN** Maximum number of channels for the receivers.
- **eN_REC** Number of receptions for the pair during the experiment.

6.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of the last update of the file.
- **comments(N_LSTR)** *char* Comments on the file content...

Pair information

- **P_name(N_PAIR_NAME)** *char* Name of the pair.
- **P_sampling** *short* The sampling frequency of the acoustic data. As the data are only one way and that a source is tuned to one fixed frequency, then the acoustic data of the pair are all sampled at the same frequency. *Unit = Hz.*
- **P_start** *int* Starting date for the pair expressed in seconds relatively to the experiment reference date. *Unit = s.*
- **P_end** *int* Ending date for the pair expressed in seconds relatively to the experiment reference date. *Unit = s.*

Status information

- **S_num(eN_REC)** *short* Reception serial number.
- **S_size(eN_REC)** *int* Size of the reception expressed in bytes.
- **S_addr_buf(eN_REC)** *int* Address of the reception in the receiver buffer.
- **S_start_rec(eN_REC)** *int* Starting date of the reception expressed in seconds relatively to the experiment reference date. *Unit = s.*
- **S_end_rec(eN_REC)** *int* Ending date of the reception expressed in seconds relatively to the experiment reference date. *Unit = s.*
- **S_source(eN_REC)** *byte* Source number. This is an arbitrary number given to the source during the task generation.
- **S_frequency(eN_REC)** *short* Source frequency. *Unit = Hz.*
- **S_signal(eN_REC)** *byte* Signal type. Up to now we used only one type of signal for all our instruments but ...

- **S_process(eN_REC)** *byte* Processing applied to the signal at reception time with the following convention :

  0 raw data (pass-band filtered)

  1 demodulated and low-pass filtered

  2 decimated

  3 correlated

  4 correlated and doppler corrected

  – by adding 10 to the above number it means that the data are in addition coherently summed.

- **S_seq(eN_REC)** *byte* Number of received sequences in the reception. A sequence is the basic unit of time of a transmission. With a M-sequence coding it corresponds to the length of a complete M-sequence.

- **S_seq_size(eN_REC)** *short* Number of samples in a sequence.

- **S_gain(eN_REC, iN_CHAN)** *byte* Automatic gain control setting for each channel of the reception. Expressed in instrument unit.

- **S_rms(eN_REC, iN_CHAN)** *short* Root mean square noise value for each channel computed when the gain is fixed. Expressed in instrument unit.

Data

- **D_index(eN_REC, TWO)** *int* The acoustic data variable **D_raw** has an unlimited dimension to afford the various possible length of each reception. A reception exact size is computed as sz = S_seq * S_seq_size * iN_CHAN. An indice for the beginning and ending of each reception in the **D_raw** array can be computed during the decoding extracting process to allow a direct access to each reception. The starting and ending indices of each reception are stored in this variable. The reception k is stored at addresses **D_raw(D_index(k,1):D_index(k,2))** or **D_raw(D_index(k,:))**.

- **D_raw(N_SZ)** *int* The raw acoustic data as decoded and extracted from the instrument storage medium.

7 Level-1 data file

This file is described by the **l1_model.cdl** model and contains the complex correlated data which are reduced to a fixed size of one sequence. This is the basic state for that level and it is the one which is archived. Starting from that state other processings are optional. Optional processings will be described with :

- a date of applied processing,
- name of processing,
- version of processing software,
- control parameters of the processing,
- processing output parameters which are stored in the file.

At that level the reciprocity is assumed and data are both ways with the convention that the 1rst mooring in the pair name is the receiver for the direct way.
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All instrument related dimensions are prefixed with **i** like **iN_CHAN** for example, while all experiment related dimensions are prefixed with **e** like **eN_REC**.
All the level-1 data files are grouped in a directory named **level1** and the naming convention for the data file is **xxxyyy[.ccc].lv1** where

- **xxx** One or more alphanumeric characters for the receiver mooring name,
- **yyy** Same number of alphanumeric characters for the source mooring name,
- **.lv1** the reserved extension for the level
- **[.ccc]** One or more optional alphanumeric characters for distinguishing different file versions. The dot ('.') preceding the optional character is mandatory to archive the file at the data banking center.

7.1 Dimensions

- **N_MOOR_NAME** A mooring must exactly have this number of characters in its name.
- **N_PAIR_NAME** A pair must exactly have this number of characters in its name. This number is exactly (N_MOOR_NAME * 2).
- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_STRING** Length of a short string.
- **N_LSTR** Length of a long string.
- **N_PROC** Maximum number of processing which is possible to apply to the data.
- **TWO** To fix a 2-dimension to arrays.
- **iN_CHAN** Maximum number of channels for the receivers.
- **eN_WAY** This dimension is set according to the reciprocity flag of the pair. Must be at least 1.
- **eN_REC** Number of receptions for the pair during the experiment.

7.2 Variables

General

- **last_update(N_DATE_TIME)** *char* Date of the last update of the file.
- **comments(N_LSTR)** *char* Comments on the file content.

Pair information

- **P_name(N_PAIR_NAME)** *char* Name of the pair.
- **P_sampling(eN_WAY)** *short* The sampling frequency of the acoustic data. As the data are both ways and that a source is tuned to one fixed frequency, then the acoustic data of the pair are eventually sampled at 2 frequencies. *Unit = Hz.*

- **P_start(eN_WAY)** *int* Starting date for the pair expressed in seconds relatively to the experiment reference date. *Unit = s.*

- **P_end(eN_WAY)** *int* Ending date for the pair expressed in seconds relatively to the experiment reference date. *Unit = s.*

- **P_win_orig(eN_WAY)** *float* Absolute time of the first sample in the reception window taking into account instrumental delays, applied windowing.... *Units = s, range = [0, 3600]*

- **P_win_orig_descr(eN_WAY, N_STRING)** *char* Description of how was obtained the absolute time in P_win_orig.

Status information

- **S_start_rec(eN_REC, eN_WAY)** *int* Starting date of the reception expressed in seconds relatively to the experiment reference date. *Unit = s.*

- **S_seq_size(eN_REC, eN_WAY)** *short* Number of samples in a sequence.

- **S_number(eN_WAY)** *short* Number of receptions for each way.

Extra parameters

- **EP_doppler(eN_REC, iN_CHAN, eN_WAY)** *float* Estimated doppler. This quantity is the doppler value associated with the best slice of the ambiguity function. *Unit = Hz.*

- **EP_win_rec(eN_REC, TWO, eN_WAY)** *short* Starting and ending points of the useable part of each reception. *Unit = sample number (starting from 1).*

- **EP_noise(eN_REC, iN_CHAN, eN_WAY)** *int* Estimated noise level computed outside the above defined window. *Unit = dB.*

- **EP_signal(eN_REC, iN_CHAN, eN_WAY)** *int* Estimated signal level computed inside the above defined window. *Unit = dB.*

Extra processings

- **PS_date(N_PROC, N_DATE_TIME)** *char* Date of the applied processing.

- **PS_process(N_PROC, N_STRING)** *char* Processing name.

- **PS_proc_ver(N_PROC, N_STRING)** *char* Processing software version.

- **PS_control(N_PROC, N_LSTR)** *char* Processing control parameters given as a list of keywords with a value (KW1 = x, KW2 = y, KW3 = z, ...) where KWi are the parameters of the processing routine. These keywords must be defined for each possible processings.

- **PS_output(N_PROC, N_STRING)** *char* Ouput parameters are given as a list of variables which are stored in the file. Once the name of a variable is known, all its dimensions and attributes can easily be retrieved.

Data

- **D_cor(eN_REC, iN_CHAN, eN_WAY)** *int* Correlated data. Each reception has a fixed length of one sequence.

8 Level-2 data file

This file is described by the **l2_model.cdl** model and contains estimated arrival peaks eventually tracked and/or identified.
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All experiment related dimensions are prefixed with **e** like **eN_AVREC**.
All the level-2 data files are grouped in a directory named **level2** and the naming convention for the data file is **xxxyyy[.ccc].lv2** where

- **xxx** One or more alphanumeric characters for the receiver mooring name,
- **yyy** Same number of alphanumeric characters for the source mooring name,
- **.lv2** the reserved extension for the level
- **[.ccc]** One or more optional alphanumeric characters for distinguishing different file versions. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

8.1 Dimensions

- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_IN_NAME** Number of characters for file names.
- **N_PAIR_NAME** Number of characters for pair names.
- **N_STRING** Normal string length.
- **N_LSTR** Long string length.
- **N_BST** Number of background states
- **eN_AVREC** Maximum number of receptions (after averaging).
- **eN_PEAKS** Maximum number of peaks (after averaging).
- **eN_WAY** Number of transmission directions.

- **eN_MP1** Number of primary model peaks.
- **eN_MP2** Number of secondary model peaks (in case of hybrid approach).
- **eN_MPS** Number of simulated model peaks.

8.2 Variables

Header information

- **last_update(N_DATE_TIME)** *char* Date and comments on updates of this file.
- **comments(N_STRING)** *char* Comments on file contents/modifications.
- **Reference_date_time(N_DATE_TIME)** *char* Date and hour chosen as a reference for the experiment. In the instrument, time is usually counted in elapsed seconds or days from a reference point. This reference time is unique for an experiment. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **Experiment_name(N_STRING)** *char* Name of the experiment.

Pair description

- **P_name(N_PAIR_NAME)** *char* Name of the pair.
- **l2_contents_id** *short* Flag denoting type of travel times (0: estimated only, 1: estimated + identified).

Arrival time estimation

- **EST_l1_name(N_IN_NAME)** *char* Name of level-1 file used for travel-time estimation.
- **EST_method_id(eN_WAY)** *short* Estimation method ID (1:simulation, 2: Local maxima, 3: Cleaning method).
- **EST_method_description(N_STRING, eN_WAY)** *char* Description of estimation method.
- **EST_parameters(N_LSTR, eN_WAY)** *char* String containing estimation parameter values, depending on the method.

- **EST_yearday(eN_AVREC, eN_WAY)** *float* Reception yearday counted from reference date (day 0). *Unit = julian days.*
- **EST_travel_time(eN_AVREC, eN_PEAKS, eN_WAY)** *double* Estimated travel times sorted by increasing order in each reception. *Unit = s.*
- **EST_travel_ertm(eN_AVREC, eN_PEAKS, eN_WAY)** *float* Estimated travel time errors. *Unit = s.*
- **EST_travel_ampl(eN_AVREC, eN_PEAKS, eN_WAY)** *float* Estimated arrival amplitudes. *Unit = dB.*
- **EST_angle_arr(eN_AVREC, eN_PEAKS, eN_WAY)** *float* Estimated (grazing) angles of arrival. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **EST_angle_arr_err(eN_AVREC, eN_PEAKS, eN_WAY)** *float* Error on estimated angles of arrival. *Unit = decimal degrees, Range = [0.0, 10.0].*
- **EST_cutoff_time(eN_AVREC, eN_WAY)** *double* Estimated cutoff times. *Unit = s.*
- **EST_cutoff_err(eN_AVREC, eN_WAY)** *float* Estimated cutoff time errors. *Unit = s.*

Peak identification
primary model peaks (usually rays ....)

- **ID_z1_name(N_IN_NAME)** *char* Name of primary (inversion-related) acoustic file used for identification .
- **ID_z1_description(N_STRING)** *char* Description of model peaks in the primary acoustic file (e.g. ray arrivals, peak arrivals etc...).
- **ID_z1_id(eN_MP1)** *int* Index pointing to peaks selected from the primary acoustic file.
- **ID_z1_BSTid(N_BST)** *int* Index pointing to background states selected from the primary acoustic file.
- **ID_z1_tt(eN_MP1, N_BST)** *double* Background arrival times of the model peaks selected from the primary acoustic file. *Unit = s.*

- **ID_z1_aa(eN_MP1, N_BST)** *float* Amplitude of the primary model peaks. *Unit = dB, Min = 0.*
- **ID_z1_as(eN_MP1, N_BST)** *float* Grazing angles at source of the primary model peaks. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **ID_z1_ar(eN_MP1, N_BST)** *float* Grazing angles at receiver of the primary model peaks. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **ID_z1_nt(eN_MP1, N_BST)** *int* Number of turning points of the primary model peaks. *Unit = count, Min = 0.*

secondary model peaks (usually rays ....)

- **ID_hybrid** *short* Flag denoting hybrid mixing of model peaks (0: no mixing 1: mixing).
- **ID_z2_name(N_IN_NAME)** *char* Name of secondary (inversion-related) acoustic file used for identification.
- **ID_z2_id(eN_MP2)** *int* ndex pointing to peaks selected from the secondary acoustic file.
- **ID_z2_BSTid(N_BST)** *int* ndex pointing to background states selected from the secondary acoustic file.
- **ID_z2_description(N_STRING)** *char* Description of model peaks isecondary acoustic file (e.g. ray arrivals, peak arrivals etc...).
- **ID_z2_tt(eN_MP2, N_BST)** *double* Background arrival times of the model peaks selected from the secondary acoustic file. *Unit = s.*
- **ID_z2_aa(eN_MP2, N_BST)** *float* Amplitude of the secondary model peaks. *Unit = dB, Min = 0.*
- **ID_z2_as(eN_MP2, N_BST)** *float* Grazing angles at source of the secondary model peaks. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **ID_z2_ar(eN_MP2, N_BST)** *float* Grazing angles at receiver of the secondary model peaks. *Unit = decimal degrees, Range = [-90.0, +90.0].*

- **ID_z2_nt(eN_MP2, N_BST)** *int* Number of turning points of the secondary model peaks.*Unit = count, Min = 0.*

Simulation

- **ID_s_rayID(eN_MPS)** *short* Ray identifier (+- number of turning points)
- **ID_s_index(eN_MPS,eN_AVREC)** *short* Index mapping the ray identifiers to the estimated arrival times contained in EST_travel_time.

Identification method

- **ID_method_id(eN_WAY)** *short* Identification method ID (1: Simulation, 2: Manual tracking + identification, 3: Statistical identification).
- **ID_method_description(N_STRING, eN_WAY)** *char* Description of identification method.
- **ID_parameters(N_LSTR,eN_WAY)** *char* String containing identification parameter values, depending on the method.

Identification results

- **ID_z1_index(eN_MP1,eN_AVREC,eN_WAY)** *int* Index mapping the primary model peaks to the estimated travel times contained in EST_travel_times.
- **ID_z1_tpred(eN_MP1, eN_AVREC, eN_WAY)** *double* Predicted travel times of the primary model peaks. *Unit = seconds.*
- **ID_z2_index(eN_MP2, eN_AVREC, eN_WAY)** *int* Index mapping the secondary model peaks to the estimated travel times contained in EST_travel_times.
- **ID_z2_tpred(eN_MP2, eN_AVREC, eN_WAY)** *double* Predicted travel times of the secondary model peaks. *Unit = seconds.*

9 Level-3 data file

This file is described by the **l3_model.cdl** model and contains results from slice inversion i.e : mode amplitudes, sound-speed and/or temperature profiles, layer averaged temperatures,...
A naming convention is used to distinguish between the dimensions which only depend on the instrument characteristics and the ones which are dependant of the experiment configuration. All experiment related dimensions are prefixed with **e** like **eN_MP1**.
All the level-3 data files are grouped in a directory named **level3** and the naming convention for the data file is **xxxyyy[.ccc].lv3** where

- **xxx** One or more alphanumeric characters for the receiver mooring name,
- **yyy** Same number of alphanumeric characters for the source mooring name,
- **.lv3** the reserved extension for the level
- **[.ccc]** Optional unlimited number of characters to build different versions of the file i.e different processings applied to the data. The dot ('.') preceding the optional characters is mandatory to archive the file at the data banking center.

9.1 Dimensions

- **N_DATE_TIME** Number of characters for a date
- **N_OCEAN_NAME** Number of characters for ocean data file
- **N_PAIR_NAME** Number of characters for pair names
- **N_STRING** Number of characters for a string
- **N_LSTR** Number of characters to describe special events
- **eN_MP1** Number of primary model peaks
- **eN_MP2** Number of secondary model peaks
- **INV_DSSM** Number of depths of sound-speed modes.
- **INV_SSM** Number of sound-speed modes

- **INV_YD** Number of yeardays
- **INV_D** Number of depths
- **INV_DL** Number of depth layers
- **INV_BST** Number of background states

9.1.1 Variables

Header information

- **last_update(N_DATE_TIME)** *char* Date and comments on updates of this file.
- **comments(N_STRING)** *char* Comments on file contents/modifications.
- **Experiment_name(N_STRING)** *char* Name of the experiment.
- **Reference_date_time(N_DATE_TIME)** *char* Date and hour chosen as a reference for the experiment. In the instrument, time is usually counted in elapsed seconds or days from a reference point. This reference time is unique for an experiment. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **Geographical_area(N_STRING)** *char* A short description of the experiment area.

Input NetCDF files

- **Exp_fname(N_STRING)** *char* Name of the experiment file.
- **l2_fname(N_STRING)** *char* Name of level-2 data file.
- **oc_fname(N_STRING)** *char* Name of ocean data file.
- **z1_fname(N_STRING)** *char* Name of primary acoustic file (inversion-related).
- **z2_fname(N_STRING)** *char* Name of secondary acoustic file (inversion-related).

Pair information

- **INV_Pair_name(N_PAIR_NAME)** *char* Mooring pair identifier (pair name).

- **INV_sect_direction(N_STRING)** *char* Description of propagation direction or averaging between reciprocal transmissions.
- **INV_sect_lat1** *double* Latidude of mooring 1. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **INV_sect_lon1** *double* Longitude of mooring 1. *Unit = decimal degrees, Range = [-180, 180].*
- **INV_sect_lat2** *double* Latidude of mooring 2. *Unit = decimal degrees, Range = [-90.0, +90.0].*
- **INV_sect_lon2** *double* Longitude of mooring 2. *Unit = decimal degrees, Range = [-180, 180].*

Slice inversion code / settings

- **INV_code_name(N_STRING)** *char* Name of inversion code used.
- **INV_code_description(N_STRING)** *char* Description of inversion code.
- **INV_type(N_STRING)** *char* Inversion type (absolute / relative-time).
- **INV_RDcor(N_STRING)** *char* String denoting whether RD correction has been applied to the acoustic data (Yes / No).
- **INV_init(N_STRING)** *char* Description of initialization data (Historical / User-defined / None).
- **INV_cutoff(N_STRING)** *char* String denoting whether the inversion has been constrained by the cutoff peak (Yes / No).
- **INV_intersect(N_STRING)** *char* String denoting whether track intersection has been allowed for (Yes / No).
- **INV_theta1_background(INV_BST)** *float* Background states used for the inversion (with respect to mode-1 amplitude).
- **ID_z1_id(eN_MP1)** *short* Model peaks from the primary acoustic file used for the inversion.
- **ID_z2_id(eN_MP2)** *short* Model peaks from the secondary acoustic file used for the inversion.

Sound-speed modes

- **INV_SSM(INV_DSSM,INV_SSM)** *float* Sound-speed modes used for the inversion.
- **INV_SSP_refer(INV_DSSM)** *float* Basic reference sound-speed profile used for the inversion. *Unit = m/s.*
- **INV_SSM_depth(INV_DSSM)** *float* Depths for sound-speed modes and reference profile. *Unit = m, Range = [0, 6000].*

Dates of slice inversion results

- **INV_yearday(INV_YD)** *float* yeardays of processed receptions, counted from Reference_date_time. *Units = julian days, Conventions = reference date = day 0.*
- **INV_depth(INV_D)** *float* Depths for inversion estimated sound-speed and temperature profiles. *Unit = m.*

Slice inversion results : Mode amplitudes

- **INV_SSM_amplitude(INV_YD,INV_SSM)** *float* Inversion estimated sound-speed mode amplitudes.
- **INV_SSM_rms_error(INV_YD,INV_SSM)** *float* Rms errors of sound-speed mode amplitudes.

Slice inversion results : Sound-speed profiles

- **INV_SSP(INV_YD,INV_D)** *float* Inversion estimated sound-speed profiles. *Unit = m/s.*
- **INV_SSP_rms_error(INV_YD,INV_D)** *float* Rms errors od sound-speed profiles. *Unit = m/s.*

Slice inversion results : Temperature profiles

- **INV_TEMP(INV_YD,INV_D)** *float* Inversion estimated profiles of potential temperature. *Unit = Celsius degree.*
- **INV_TEMP_rms_error(INV_YD,INV_D)** *float* Potential temperature rms errors. *Unit = Celsius degree.*

Slice inversion results : Depth averaged temperatures (layer averages)

- **INV_L_depth_min(INV_DL)** *float* Layer minimum depths. *Unit = m.*
- **INV_L_depth_max(INV_DL)** *float* Layer maximum depths. *Unit = m.*
- **INV_L_TEMP(INV_YD,INV_DL)** *float* Inversion estimated depth-averaged potential temperatures over the defined layers. *Unit = Celsius degree.*
- **INV_L_TEMP_rms_error(INV_YD,INV_DL)** *float* Rms errors of depth-averaged potential temperatures. *Unit = Celsius degree.*

10 Formats for reduced database

10.1 Reduced ocean database

This file is used to create a small ocean database which contains the data relative to the area of the experiment. The file model is given in **o_oceandb.cdl**.

10.1.1 Dimensions

- **eN_DX** Maximum number of points used to discretize the area in longitude.
- **eN_DY** Maximum number of points used to discretize the area in latitude.
- **eN_DZ** Maximum number of points used to discretize the area in depth.
- **eN_TIME** Maximum number of points used to discretize the parameters in time.

10.1.2 Variables

- **LATITUDE(eN_DY)** *double* Latitude of gridpoints. *Unit = degrees_north, Range = [-90.0, +90.0]*
- **LONGITUDE(eN_DX)** *double* Longitude of gridpoints. *Unit = degrees_east, Range = [-180.0, +180.0]*
- **Z_level(eN_DZ)** *float* Depth levels defined for the vertical discretisation. *Unit = meter, Range = [0, 15000].*
- **TIME(eN_TIME)** *float* Time defined for the parameters. *Unit = days.*
- **TEMP(eN_TIME, eN_DZ, eN_DY, eN_DX)** *float* Temperature (in situ). *Unit = degree Celsius, Range = [-3.0, +40.0].*
- **PSAL(eN_TIME, eN_DZ, eN_DY, eN_DX)** *float* Practical Salinity (sal78). *Unit = PSU, Range = [0, 60].*

10.2 Reduced bathymetry

This file is used to create a small bathymetry which contains the data relative to the area of the experiment. The file model is given in **o_bathydb.cdl**.

10.2.1 Dimensions

- **eN_DX** Maximum number of points used to discretize the area in longitude.
- **eN_DY** Maximum number of points used to discretize the area in latitude.

10.2.2 Variables

- **LATITUDE(eN_DY)** *double* Latitude of gridpoints. *Unit = degrees_north, Range = [-90.0, +90.0]*
- **LONGITUDE(eN_DX)** *double* Longitude of gridpoints. *Unit = degrees_east, Range = [-180.0, +180.0]*
- **Z(eN_DY, eN_DX)** *float* Elevation. *Unit = meter, Range = [-12000, 10000].*

10.3 CTD data

This file is used to export/import CTD data in Tomolab. The file model is described in **o_ctd.cdl**.

10.3.1 Dimensions

- **N_DATE_TIME** Number of characters used to write a date.
- **mN_PROF** Maximum number of CTD profiles in this file.
- **mN_ZLEV** Maximum number of vertical levels.

10.3.2 Variables

- **DATE(mN_PROF, N_DATE_TIME)** *char* Date time of each profile. *Conventions = "DD/MM/YYYY HH24:MI:SS.*
- **LATITUDE(mN_PROF)** *double* Latitude of each profile. *Unit = degrees_north, Range = [-90, +90].*
- **LONGITUDE(mN_PROF)** *double* Longitude of each profile. *Unit = degrees_east, Range = [-180, 180].*
- **BOTTOM_DEPTH(mN_PROF)** *float* Bottom depth of each profile. *Unit = m, Range = [0, 15000].*

- **PRES(mN_PROF, mN_ZLEV)** *float* Pressure. *Unit = decibar = 1000 Pa, Range = [0, 15000].*
- **TEMP(mN_PROF, mN_ZLEV)** *float* Temperature (in situ). *Unit = degree Celsius, Range =[-3, 40].*
- **PSAL(mN_PROF, mN_ZLEV)** *float* Practical Salinity (sal78). *Unit = PSU, Range = [0, 60].*

11 Data base of instruments

This file is described by the **dbi.cdl** model. It is the place where the technical characteristics of the european (others are welcome too) tomographic instruments are recorded. The description is done in terms of modules which are the basic components of the instruments. Traceability of deployments, maintenance and updates performed on the instruments should be possible through a constant updating of that file. Clock and temperature/pressure sensors history is memorized here too by keeping trace of calibrations and characteristics measured/deduced from experiments. This file should be maintained by a reduced number of persons.

11.1 Dimensions

- **N_NAME** This is the maximum number of characters in an instrument or module name.
- **N_DATE_TIME** The dimension of date and time strings. The format is DD-MMM-YYYY HH:MM:SS where hours are given in a 24-hour format.
- **N_STRING** The maximum number of characters in a short string.
- **N_LSTR** The maximum number of characters in a long string.
- **N_INST** The number of instrument described in the database.
- **N_FREQ** The maximum number of frequencies supported by receivers
- **N_CHAN** The maximum number of channels supported by receivers
- **N_BEACON** The maximum number of received frequencies supported by navigators.
- **N_NRT** The number of recoverable transponders used for navigation.
- **N_COEF** The maximum nunber of coefficients used to convert sensors engineering units to physical units. A second order polynomial with time varying coefficients is currently in use. The method used to compute the coefficients is detailed in the variables section.
- **N_POW_SUP** The maximum number of power supplies in the instruments. Some instruments have separate power supplies for the various functions of the instrument : emission, clock, navigation, ...

- **N_IR** The maximum number of points used to measure the frequency response of the source.
- **N_CAL** The maximum number of sensors calibration.
- **N_VALP** The maximum number of points measured for the pressure sensor calibrations.
- **N_VALT** The maximum number of points measured for the temperature sensor calibrations.
- **N_AGING** Max number of aging corrections
- **N_EXP** The maximum number of involvement in experiment. This is to keep trace of the various deployments of the instruments.
- **N_PTS** The maximum number of points used to describe a clock drift curve as sometimes estimated from the acoustic data.

11.2 Variables

General

- **last_update(N_DATE_TIME)** *char* date and time of the last modification. This variable must be set up by the persons/programs who are able/authorised to modify it.
- **comments(N_LSTR)** *char* Comments on the modifications done. This variable must be set up by the persons/programs who are able/authorised to modify it.

Instrument

- **I_id(N_INST, N_NAME)** *char* Contains the identifier of each instrument. This identifier is normally unique for each instrument.
- **I_owner(N_INST, N_STRING)** *char* Instrument owner and eventually the date of acquisition.
- **I_type(N_INST, N_STRING)** *char* For example ERATO or SARA or JHAS or ....
- **I_function(N_INST, N_STRING)** *char* A word to say if the instrument is a transceiver, a receiver or a source.

- **I_max_depth(N_INST)** *short* Maximum allowed depth for the instrument. *Unit = m, Range = [0, 2000].*

- **I_bat_cap(N_INST, N_POW_SUP)** *short* A number which indicates the capacity of each battery block dedicated to each power supply. The convention for the power supplies is : 1- emission function; 2- Navigation/storage; 3- General electronics; 4- Clock. *Unit = Ah.*

- **I_nb_bat(N_INST)** *short* A number which gives the maximum number of batteries blocks dedicated to the emission function for each instrument. These 2 numbers are used to design an experiment.

- **I_history(N_INST, N_EXP, N_STRING)** *char* A string variable to keep trace of experiment deployment for each instrument.

- **I_comment(N_INST, N_LSTR)** *char* A string variable to describe special events during experiment : failure, broken parts,...

Controller

- **CX_id(N_INST, N_NAME)** *char* Identifier of the controller module.

- **CX_ver_soft(N_INST, N_STRING)** *char* The current software version of the controller.

- **CX_gen_task(N_INST, N_STRING)** *char* The program used to generate a text file describing the task to perform (usually .tsk files)

- **CX_comp_task(N_INST, N_STRING)** *char* The program used to compile and generate the executable task which is downloaded into the instrument controller.

- **CX_cons_sleep(N_INST)** *float* The consumption of the controller module in sleep mode. *Unit = Ah.*

- **CX_cons_awake(N_INST)** *float* The consumption of the controller module in run mode. *Unit = Ah.*

- **CX_ip_n(N_INST)** *short* Number of measured internal parameters at each acquisition. Internal parameters are mainly the different voltages of the power supplies used by the electronics and the internal pressure and temperature. This number varies with the instrument type.

- **CX_ip_run(N_INST)** *short* Execution time to measure all the internal parameters. Quantity used to generate the task and to check the controller timing coherence. *Unit = s.*
- **CX_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the controller module deployments and/or updates.
- **CX_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each controller.

Clock

- **CK_id(N_INST, N_NAME)** *char* Clock identifier.
- **CK_type(N_INST, N_STRING)** *char* Type of clock (crystal, rubidium, ...)
- **CK_set_fr(N_INST)** *short* Time needed to have a stabilized 1 MHz. When ask to output the 1 MHz frequency, the clock starts a phase lock loop (PLL) in order to deliver a stable 1 MHz. The stabilisation time of the PLL to obtain a precise 1 MHz (i.e with a stability less than 10^{-6}) is rather long ($\approx$ 7 min) and this is important to emit or sample a signal without bias on the frequency. *Unit = s.*
- **CK_cons_sleep(N_INST)** *float* Consumption of the clock module in sleep mode. *Unit = Ah.*
- **CK_cons_awake(N_INST)** *float* Consumption of the clock module in run mode. *Unit = Ah.*
- **CK_aging_date(N_INST, N_AGING, N_DATE_TIME)** *char* Aging correction date.
- **CK_aging_temp(N_INST, N_AGING)** *float* Aging correction temperature. *Unit =* ^oC.
- **CK_aging_coef(N_INST, N_AGING)** *float* This coefficient used to compensate the crystal aging is adjusted to reduce the drift before an experiment. This variable keep trace of the values of that coefficients. Unit is instrument dependant.
- **CK_experiment(N_INST, N_EXP, N_STRING)** *char* Experiment name.

- **CK_date_ante(N_INST, N_EXP, N_DATE_TIME)** *char* Date of UTC check at deploiement.
- **CK_utc_ante(N_INST, N_EXP)** *float* Difference with UTC at deploiement. *Unit = s.*
- **CK_date_post(N_INST, N_EXP, N_DATE_TIME)** *char* Date of UTC check at recovery.
- **CK_utc_post(N_INST, N_EXP)** *float* Difference with UTC at recovery. *Unit = s.*
- **CK_drift(N_INST, N_EXP)** *float* This variable contains the drift measured after each experiment.
- **CK_drift_curve(N_INST, N_EXP, N_PTS)** *float* This variable will contain a drift curve of the clock which is estimated from the acoustic data.
- **CK_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the clock.
- **CK_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each clock.

Transducer

- **TR_id(N_INST, N_NAME)** *char* Identifier of the transducer. With the ERATO it is the same as the instrument but it is not necessarly the case for the HLF or JHAS transducers.
- **TR_type(N_INST, N_STRING)** *char* A few words to describe the transducer type.
- **TR_frequency(N_INST)** *short* Frequency for which the transducer is used. *Unit = Hz.*
- **TR_bandwidth(N_INST)** *short* Transducer bandwidth. *Unit = Hz.*
- **TR_level(N_INST)** *short* Resulting acoustic level. *Unit = dB re 1μPa @ 1m.*
- **TR_ir_fr(N_INST, N_IR)** *float* In case we have the frequency response of the transducer, this variable will contain the frequencies of measurement. *Unit = Hz.*

- **TR_ir_lv(N_INST, N_IR)** *float* In case we have the frequency response of the transducer, this variable will contain the level of the transducer for each frequency point above. *Unit = dB.*
- **TR_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the transducer.
- **TR_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each transducer.

Power amplifier

- **PA_id(N_INST, N_NAME)** *char* Power amplifier identifier.
- **PA_type(N_INST, N_NAME)** *char* Power amplifier type.
- **PA_power(N_INST)** *short* Electrical power. *Unit = VA.*
- **PA_cons_awake(N_INST)** *float* Power amplifier run mode consumption. *Unit = Ah.*
- **PA_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the power amplifier.
- **PA_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each power amplifier.

Transmitter

- **TX_id(N_INST, N_NAME)** *char* Transmitter identifier. This module generates the signal and delivers it to the power amplifier which drives the transducer. Normally TR, PA and TX are not separable modules and are all parts of the same source. It should not be allowed by the sofware to mix these elements. In case there is a need to do it great care must be taken to insure there is a total compatibility between the modules.
- **TX_ver_soft(N_INST, N_STRING)** *char* Current software version of the module.
- **TX_frequency(N_INST)** *short* Frequency delivered by the signal generator. *Unit = Hz.*
- **TX_signal_code(N_INST)** *short* Signal code used in the transmitter. For m-sequence PSK signal, the code is the shift register loop combination.

- **TX_delay(N_INST)** *float* This is the time between the SYNC pulse i.e the command to deliver the sound and the time when the sound is really in the water. This value must be periodically checked in lab. *Unit = s.*
- **TX_cons_awake(N_INST)** *float* Consumption of the source in run mode. *Unit = Ah.*
- **TX_arm(N_INST)** *short* Time needed by the controller to prepare a transmission : power on and initialization for the transmission. *Unit = s.*
- **TX_run(N_INST)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the transmission. *Unit = s.*
- **TX_end(N_INST)** *short* Time needed by the controller to cleanly terminate the transmission and retrieve the transmitter status. *Unit = s.*
- **TX_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the source.
- **TX_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each source.

Receiver

- **RX_id(N_INST, N_NAME)** *char* Receiver module identifier.
- **RX_ver_soft(N_INST, N_STRING)** *char* Current software version.
- **RX_frequencies(N_INST, N_FREQ)** *short* Frequencies the receiver is able to acquire. *Unit = Hz.*
- **RX_channel(N_INST)** *short* Number of channels the receiver is able to acquire.
- **RX_gain(N_INST, N_FREQ, N_CHAN)** *short* Gain of the receiver for each channel and each frequency. *Unit = dB.*
- **RX_agc_min(N_INST)** *short* Minimum automatic gain control value. *Unit = dB.*

- **RX_agc_max(N_INST)** *short* Maximum automatic gain control value. *Unit = dB.*
- **RX_agc_step(N_INST)** *float* Automatic gain control step. *Unit = dB.*
- **RX_delay(N_INST, N_FREQ)** *float* Delay between the SYNC pulse and the first acquired sample at each frequency. *Unit = s.*
- **RX_offset_acq(N_INST, N_FREQ, N_CHAN)** *float* Delays between the first samples of each channel. This is relevant when the channels are not simultaneously sampled. *Unit = s.*
- **RX_cons_awake(N_INST)** *float* Consumption of the receiver module in run mode. *Unit = Ah.*
- **RX_arm(N_INST)** *short* Time needed by the controller to initiate a reception. *Unit = s.*
- **RX_run(N_INST)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the reception. *Unit = s.*
- **RX_end(N_INST)** *short* Time needed by the controller to terminate a reception and get the receiver status back. *Unit = s.*
- **RX_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the receiver.
- **RX_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each receiver.

Receiving Array

- **AR_id(N_INST, N_NAME)** *char* Array identifier. An array can eventually be composed of a single hydrophone. Array is a word for acoustic receiving sensor.
- **AR_type(N_INST, N_STRING)** *char* Make and type of acoustic receiving sensor.
- **AR_channel(N_INST)** *int* Number of channels in the array.
- **AR_length(N_INST)** *int* Length of the array. Zero if the array is a single hydrophone. *Unit = m.*

- **AR_offset_depth(N_INST)** *float* Offset of depth for the array center relatively to the pressure sensor position. The pressure sensor is the reference point for the instrument immersion. *Unit = m.*
- **AR_geom(N_INST, N_CHAN)** *float* Gives the distance of each acoustic receiving center of each channel to the array center. *Unit = m.*
- **AR_type_hydro(N_INST, N_STRING)** *char* Make and type of the hydrophones used in the array.
- **AR_sh(N_INST, N_CHAN)** *short* Sensitivity of each channel. This includes the individual hydrophone sensitivity plus a possible array gain. *Unit = dB re 1V/μPa.*
- **AR_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the array.
- **AR_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each array.

Navigator

- **NV_id(N_INST, N_NAME)** *char* Navigator identifier.
- **NV_ver_soft(N_INST, N_STRING)** *char* Current software version.
- **NV_freq_ping(N_INST)** *short* Frequency of the emitted ping. *Unit = Hz.*
- **NV_time_ping(N_INST)** *float* Duration of the emitted ping. *Unit = s.*
- **NV_counting(N_INST)** *byte* Boolean to indicates when the navigator starts to measure the travel times. 1 if it starts the measure at the pulse end. Some navigators start the measure of the travel times at the end of the ping duration and in that case the ping duration must be taken into account for the correct travel times.
- **NV_delay_ping(N_INST)** *float* Delay between the SYNC pulse and the emitted ping. *Unit = s.*
- **NV_blank_ping(N_INST)** *float* Blanking of the reception after the ping. *Unit = s.*

- **NV_freq_rcv(N_INST, N_BEACON)** *short* Frequencies of the navigator in receiving mode. *Unit = Hz.*
- **NV_delay_rcv(N_INST, N_BEACON)** *float* Delay between the frequencies detection and the stop of counting. *Unit = s.*
- **NV_res_rcv(N_INST)** *float* Resolution measurement i.e. sampling period of received signals. *Unit = s.*
- **NV_offset_depth(N_INST)** *short* Offset of depth for the navigation transducer relatively to the pressure sensor position. The pressure sensor is the reference point for the instrument immersion. *Unit = m.*
- **NV_cons_awake(N_INST)** *float* Consumption of the navigator module in run mode. *Unit = Ah.*
- **NV_arm(N_INST)** *short* Time needed by the controller to initiate a navigation. *Unit = s.*
- **NV_run(N_INST)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the navigation. *Unit = s.*
- **NV_end(N_INST)** *short* Time needed by the controller to terminate a navigation and get back the navigator status. *Unit = s.*
- **NV_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the navigator.
- **NV_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each navigator.

Recoverable navigation transponders

- **NB_id(N_NRT, N_NAME)** *char* Acoustic transponder identifier. Set to *exp.* when expandable ones are used.
- **NB_type(N_NRT, N_STRING)** *char* Acoustic transponder type and other comments on the recoverable transponders.
- **NB_sernum(N_NRT, N_STRING)** *char* Acoustic transponder serial number.
- **NB_inter_freq(N_NRT)** *short* Interrogation frequency of the transponders. *Unit = Hz.*

- **NB_resp_freq(N_NRT)** *short* Frequency of each transponder. *Unit = Hz.*
- **NB_resp_delay(N_NRT)** *float* Internal delay of each transponder. *Unit = s.*
- **NB_history(N_RNT, N_EXP, N_STRING)** *char* To keep trace of the deployments of the transponders.
- **NB_comment(N_RNT, N_LSTR)** *char* To add some extra comments on the life of each transponder

Pressure and temperature module

- **PT_id(N_INST, N_NAME)** *char* Pressure and temperature measurements module identifier.
- **PT_Ptype(N_INST, N_STRING)** *char* Type of pressure sensor.
- **PT_Psernum(N_INST, N_STRING)** *char* Serial number of pressure sensor.
- **PT_Prange(N_INST, TWO)** *int* Pressure sensor range. *Unit = db* $= 10^4 Pa$.
- **PT_Ttype(N_INST, N_STRING)** *char* Type of temperature sensor.
- **PT_sernum(N_INST, N_STRING)** *char* Serial number of temperature sensor.
- **PT_Trange(N_INST, TWO)** *int* Temperature sensor range. *Unit* $= {}^oC$.
- **PT_cons_awake(N_INST)** *float* Consumption of the module in run mode. *Unit = Ah.*
- **PT_arm(N_INST)** *short* Time needed by the controller to initiate a measure. *Unit = s.*
- **PT_run(N_INST)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the measure. *Unit = s.*

- **PT_end(N_INST)** *short* Time needed by the controller to terminate a measure and get the pressure and temperature module status back. *Unit = s.*

- **PT_dates_cal_P(N_INST, N_CAL, N_DATE_TIME)** *char* To keep trace of the dates of the calibration. Usually a calibration is performed both before and after an experiment in order to better qualify the sensors.

- **PT_cal_Pref(N_INST, N_CAL, N_VALP)** *float* For each calibration, the reference pressure values will be kept in that variable. *Unit* $= db = 10^4 Pa$.

- **PT_cal_Praw(N_INST, N_CAL, N_VALP)** *float* For each calibration, the raw values as delivered by the sensors will be kept in that variable.

- **PT_conv_Pcoefs(N_INST, N_CAL, N_COEF)** *float* The conversion coefficients $[a\ b\ c]^T$ used to translate from engineering units to physical units are stored in that variable. Each coefficient is supposed to have linear drift in time. Both before and after cruise calibrations data are used to estimate the conversion coefficients and their temporal drift coefficients. The set of coefficients is computed as a quadratic fit to the data. Knowing the sensor outputs X to the reference values R taken at different time, the coefficients, under the hypothesis of a linear drift in time, must satisfy :

$$R = a(t)X^2 + b(t)X + c(t) = a \times (1 + \frac{\alpha}{a}t)X^2 + b \times (1 + \frac{\beta}{b}t)X + c \times (1 + \frac{\gamma}{c}t) \tag{1}$$

This equation can be written in matrix notation as

$$[R_1 R_2]^T = G \times A \tag{2}$$

where R_1 and R_2 are the reference values for each calibration, A the coefficients vector $[a\ b\ c\ \alpha\ \beta\ \gamma]^T$. G is the of the form

$$G = \begin{bmatrix} x_{11}^2 & x_{11} & 1 & 0 & 0 & 0 \\ & & \vdots & & & \\ x_{1i}^2 & x_{1i} & 1 & 0 & 0 & 0 \\ & & \vdots & & & \\ x_{1n}^2 & x_{1n} & 1 & 0 & 0 & 0 \\ x_{21}^2 & x_{21} & 1 & t \times x_{21}^2 & t \times x_{21} & t \\ & & \vdots & & & \\ x_{2i}^2 & x_{2i} & 1 & t \times x_{2i}^2 & t \times x_{2i} & t \\ & & \vdots & & & \\ x_{2m}^2 & x_{2m} & 1 & t \times x_{2m}^2 & t \times x_{2m} & t \end{bmatrix} \tag{3}$$

As far as the number of calibration values is greater than the number of coefficients, the solution is given as

$$A = (G^T G)^{-1} G^T R \tag{4}$$

- **PT_drift_Pcoefs(N_INST, N_CAL, N_COEF)** *float* Temporal drift coefficients $[\alpha\ \beta\ \gamma]^T$ associated to the conversion coefficients. Computation of the coefficients is explained above.

- **PT_Perror(N_INST, N_CAL)** *float* Measurement error as deduced from the calibrations. It is defined as the maximum of the calibration residuals. *Unit = dB.*

- **PT_dates_cal_T(N_INST, N_CAL, N_DATE_TIME)** *char* To keep trace of the dates of the calibration. Usually a calibration is performed both before and after an experiment in order to better qualify the sensors.

- **PT_cal_Tref(N_INST, N_CAL, N_VALT)** *float* For each calibration, the reference temperature values will be kept in that variable. *Unit = ^oC.*

- **PT_cal_Traw(N_INST, N_CAL, N_VALT)** *float* For each calibration, the raw values as delivered by the sensors will be kept in that variable.

- **PT_conv_Tcoefs(N_INST, N_CAL, N_COEF)** *float* The conversion coefficients $[a\ b\ c]^T$ used to translate from engineering units to physical units are stored in that variable. See pressure coefficients for the computation.

- **PT_drift_Tcoefs(N_INST, N_EXP, N_COEF)** *float* Temporal drift coefficients $[\alpha\ \beta\ \gamma]^T$ associated to the conversion coefficients. Computation of the coefficients is explained above.

- **PT_Terror(N_INST, N_CAL)** *float* Measurement error as deduced from the calibrations. It is defined as the maximum of the calibration residuals. $Unit = {}^oC$.

- **PT_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the P&T module.

- **PT_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each P&T module.

Storage unit

- **ST_id(N_INST, N_NAME)** *char* Storage unit identifier.

- **ST_type(N_INST, N_STRING)** *char* Storage unit type.

- **ST_sernum(N_INST, N_STRING)** *char* Storage unit serial number.

- **ST_capacity(N_INST)** *short* Storage unit capacity expressed in Mbytes.

- **ST_cons_awake(N_INST)** *float* Consumption of the storage module in run mode. $Unit = Ah$.

- **ST_arm(N_INST)** *short* Time needed by the controller to initiate a storage. $Unit = s$.

- **ST_run(N_INST)** *short* Time needed by the controller to execute some instructions during the run in addition to the specific time of the storage. $Unit = s$.

- **ST_end(N_INST)** *short* Time needed by the controller to terminate a storage and get back the storage status. $Unit = s$.

- **ST_history(N_INST, N_EXP, N_STRING)** *char* To keep trace of the deployments of the storage unit.

- **ST_comment(N_INST, N_LSTR)** *char* To add some extra comments on the life of each storage unit.