EGO gliders data processing chain

Coriolis decoder user's manual

Version 2.9

March 24th, 2022

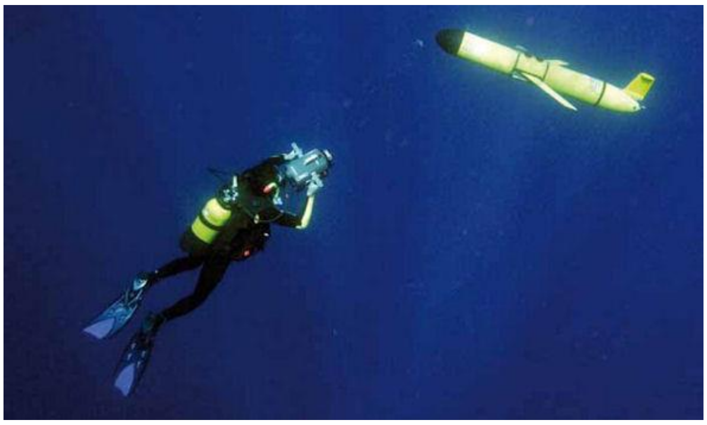


Table of contents

History

Version	Date	Comment
1.0	15/04/2016	JPR: initial version of the document.
1.1	26/08/2016 21/07/2017	JPR: Annex G added after implementation of DOXY derived parameter processing. JPR: link to download ETOPO2 file added after Riccardo GERIN review of the document.
2.0	25/02/2019	JPR: updated for decoder V005a.
2.1	02/01/2020	JPR: updated for decoder V006a (introduction of a dedicated configuration file).
2.2	27/01/2020	JPR: updated for decoder V007a (processing of SeaGlider NetCDF files).
2.3	27/08/2020	JPR: updated for decoder V008a: - Use of GEBCO bathymetric atlas for RTQC test #4, - Add RTQC test #25 (MEDD test), - "references" and "citation" can be set in the JSON deployment file.
2.4	30/03/2021	JPR: updated for decoder V008b: added processing of BBP470.
2.5	28/04/2021	JPR: updated for decoder V009a: added RTQC test #57.
2.6	01/06/2021	JPR: updated for decoder V010a: added case 102_207_206 for DOXY processing.
2.7	23/06/2021	JPR: added CSV as SeaExplorer possible input data format.
2.8	24/02/2022	JPR: added "linear function" as a new derivation method. JPR: added "gl_trace_parameter" visualization tool. JPR: updated the "coordinate_variables" management so that locations provided by the Glider can be stored and used (together with GPS fixes) to linearly interpolate unlocated measurements.
2.9	24/03/2022	JPR: two new input data file types for Slocum glider and one for SeaExplorer.

Reference documents

Reference N°	Title	Link
RD1	EGO gliders NetCDF format reference manual.	http://dx.doi.org/10.13155/34980
RD2	Argo Quality Control Manual for CTD and Trajectory Data.	http://dx.doi.org/10.13155/33951
RD3	Argo Quality Control Manual for Biogeochemical Data.	http://dx.doi.org/10.13155/40879
RD4	Processing Argo OXYGEN data at the DAC level.	http://doi.org/10.13155/39795
RD5	Processing BGC-Argo particle backscattering at the DAC level.	http://dx.doi.org/10.13155/39459
RD6	Processing Bio-Argo chlorophyll-a concentration at the DAC level.	http://dx.doi.org/10.13155/39468
RD7	Processing BGC-Argo CDOM concentration at the DAC level.	http://doi.org/10.13155/54541
RD8	Processing Bio-Argo nitrate concentration at the DAC Level.	http://dx.doi.org/10.13155/46121

1 Introduction

This user's manual describes how to install, configure and use the Coriolis Matlab decoder used to process EGO glider data.

The main function of the decoder is to format the glider data (glider meta-data, deployment meta-data and glider measurements) into a unique EGO NetCDF file specified in [RD1]. The glider measurements can be low resolution data transmitted by the instrument or high resolution data downloaded from its memory after recovery.

The decoder can also perform the following optional actions:

- Apply Real Time Quality Control (RTQC) tests on EGO file time series,
- Estimate Slocum subsurface currents and store them into the EGO file,
- Generate NetCDF profile files from EGO file data and apply specific RTQC tests to them.

The glider decoder is used through the ***gl_process_glider*** Matlab program.

The ***gl_process_glider_rt*** Matlab program additionally generates an XML report of the processing done and is designed to be used in a Data Assembly Center (DAC) real time flux.

This decoder user's manual (V2.9) describes the Coriolis EGO Glider decoder V011h (with RTQC V1.9) which generates EGO file compliant with format V1.4 (specified in [RD1]).

The current version of the decoder is stored in the global variable g_decGl_decoderVersion set in the *gl_init_default_values.m* file.

The current version of the RTQC is stored in the global variable g_decGl_rtqcVersion set in the *gl_add_rtqc_to_ego_file.m* file.

2 Glider data managed by the decoder

The following glider data formats are managed by the decoder:

- For a Slocum glider:
  - The *_sbd.dat files with their associated *_sbd.m files,
  - The *_sbd.dat and *_tbd.dat files with their associated *_sbd.m and *_tbd.m files,
  - The *_mbd.dat and *_nbd.dat files with their associated *_mbd.m and *_nbd.m files,
- For a SeaGlider glider:
  - The p*.bpo files,
  - The p*.pro files,
  - The p*.eng files and their associated p*.log files; the ppc*a.eng and ppc*b.eng files of the pumped CTD data if any,
  - The p*.nc files,
- For a SeaExplorer glider:
  - The *.gli.*.gz files and their associated *.pl*.gz files,
  - The *.gli.*.csv files and their associated *.pl*.csv files (the uncompressed version of the *.gz files),
  - The *.pl*.raw.csv files (for HR data).

3 Derived parameters

The decoder is able to compute the following derived parameters:

- PSAL,
- DOXY,
- BBP470,
- BBP700,
- CDOM,
- CHLA,
- NITRATE,
- TURBIDITY,
- Linear function.

4 Decoder installation and configuration

4.1 Decoder installation

4.1.1 Hardware and software requirements

The decoder can be installed on a PC type computer equipped with a Windows or Linux operating system.

The Matlab software must be installed on the host computer.

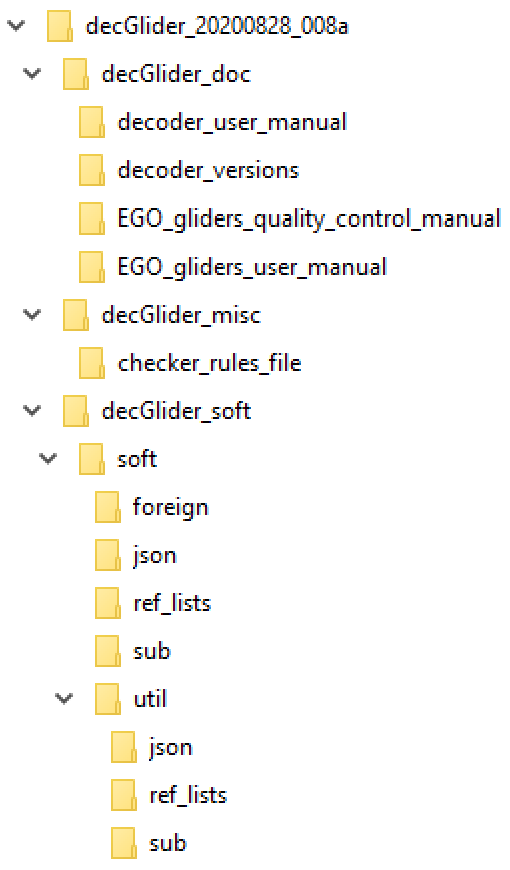
The Matlab version should be > R2006b since a native NetCDF library is expected.

Note also that:

- The GEBCO worldwide bathymetric atlas (https://www.bodc.ac.uk/data/open_download/gebco/gebco_2020/zip/ file) is needed by some RTQC tests.

4.1.2 Installation of the decoder

The tree diagram of the decoder is illustrated in the following figure.



```
decGlider_20200828_008a
    decGlider_doc
        decoder_user_manual
        decoder_versions
        EGO_gliders_quality_control_manual
        EGO_gliders_user_manual
    decGlider_misc
        checker_rules_file
    decGlider_soft
        soft
            foreign
            json
            ref_lists
            sub
            util
                json
                ref_lists
                sub
```

Three main directories are provided:

- The *decGlider_soft* directory contains the decoder software,
- The *decGlider_doc* directory contains the documentation associated to the decoder,
- The *decGlider_misc* directory contains miscellaneous files associated to the decoder.

To install the decoder, duplicate the *decGlider_soft/soft* directory to the appropriate directory of the host computer and add it, with its sub-directories, in your Matlab path.

4.2 Decoder configuration

To configure the decoder, you have to update, according to your own (Linux or PC) platform, the decoder configuration file.

This file, named *_glider_decoder_conf.txt*, is provided in the *decGlider_soft/soft* directory (note that **it can be moved anywhere in your Matlab path but cannot be renamed**).

Configuration variable descriptions are given in the following table.

Configuration variable name	Configuration variable description
DATA_DIRECTORY	Top directory of deployment directories.
LOG_DIRECTORY	Directory to store log files.
XML_DIRECTORY	Directory to store XML reports (***gl_process_glider_rt*** program only).
EGO_FORMAT_JSON_FILE	JSON reference file of the EGO 1.4 format (*decGlider_soft/soft/json/EGO_format_1.4.json* file).
COMPUTE_SLOCUM_SUBSURFACE_CURRENT	Flag to compute Slocum estimates of the subsurface current (1 to compute, 0 otherwise).
GENERATE_PROFILE_FILES	Flag to generate profile files from EGO file time series (1 to generate, 0 otherwise).
APPLY_RTQC_TESTS	Flag to apply RTQC test to EGO file time series, and if generated, to profile file data (1 to apply, 0 otherwise).
TESTXXX_*	Flag to apply this particular RTQC test (1 if you want to apply this test, 0 otherwise). See [RD2] or [RD3] for the concerned test description. Note that RTQC tests #2, 3, 4, 6, 7, 9, 11, 15, 19, 20, 25, 57 are performed on EGO file data whereas RTQC tests #8, 12, 13, 14 are performed on profile file data (generated when GENERATE_PROFILE_FILES is set to 1).
TEST004_GEBCO_FILE	If RTQC test #4 should be applied, set to the GEBCO file path name.
TEST015_GREY_LIST_FILE	If RTQC test #15 should be applied, set to the greylist file path name.

5 Using the decoder

The input arguments of the ***gl_process_glider*** or ***gl_process_glider_rt*** programs should be provided in pairs, i.e. ('argument_name', 'argument_value').

Allowed ‘argument_name’ values are:

- ‘glidertype’: to specify the type of the glider to process (the corresponding 'argument_value' value is expected to be: ‘seaglider’, ‘slocum’ or ‘seaexplorer’),
- ‘data’: to identify a given deployment to process (the corresponding 'argument_value' value is expected to be the name of one sub directory of the DATA_DIRECTORY directory).

both arguments are mandatory.

Examples:

gl_process_glider('glidertype', 'seaglider', 'data', ‘nearchos_mooseperseust02_09’)

- ⇨ process the glider deployment stored in the directory DATA_DIRECTORY/ *nearchos_mooseperseust02_09*/,
- ⇨ process this deployment as a SeaGlider glider one.

6 Decoder input data

The expected input data are:

- The EGO definition file format,
- The glider meta-data,
- The deployment meta-data,
- The glider measurements (low or high resolution data sampled by the glider during the deployment).

The EGO definition file format is part of the decoder and **should not be modified by users** (its location is set by the EGO_FORMAT_JSON_FILE configuration variable).

All other information should be provided in the directory DATA_DIRECTORY/<*glider_name*>_<*deployment_name*>, where <*glider_name*> and <*deployment_name*> should be replaced by the name of the glider and the name of the deployment respectively.

The DATA_DIRECTORY/<*glider_name*>_<*deployment_name*> should contain:

- A "*json*" directory for the glider and deployment meta-data,
- A "<*data*>" directory for the glider measurements:

  The <*data*> could be:

  - "*dat*" for a Slocum glider (it then contains the *_sbd.dat files with their associated *_sbd.m files),
  - "*bpo*" for a SeaGlider glider which provides p*.bpo files,
  - "*pro*" for a SeaGlider glider which provides p*.pro files,
  - "*eng*" for a SeaGlider glider which provides p*.eng files and their associated p*.log files; and ppc*a.eng and ppc*b.eng files of the pumped CTD data if any,
  - "*nc_sg*" for a SeaGlider glider which provides p*.nc files,
  - "*gz*" for a SeaExplorer glider (it then contains the *.gli.*.gz files and their associated *.pl*.gz files),
  - "*csv*" for a SeaExplorer glider (it then contains the *.gli.*.csv files and their associated *.pl*.csv files).

6.1 Deployment JSON input files

The DATA_DIRECTORY/<*glider_name*>_<*deployment_name*>*/json* directory should contain glider and deployment meta-data.

This information is provided in the following JSON files:

- One configuration file for the deployment, called <*glider_name*>_<*deployment_name*>*.json*,
- One configuration file for each sensor mounted on the glider.

6.1.1 The JSON deployment configuration file

The JSON configuration file of the deployment contains the following sections:

a. '*EGO_format_version*',

b. '*global_attributes*',

c. '*glider_characteristics*',

d. '*glider_deployment*',

e. '*coordinate_variables*',

f. '*glider_sensor*',

'*EGO_format_version*' item defines the EGO version associated to the JSON file.

See EGO gliders user's manual ([RD1]) to get the definition of expected information for items 'b', 'c' and 'd'.

'*coordinate_variables*' item explains how to create EGO coordinate variables from glider provided variables.

'*glider_sensor*' item refers to external files associated to each sensor mounted on the glider.

See Annex H to get a detailed description of the JSON deployment configuration file.

6.1.2 The JSON sensor configuration files

The name of this file is free (we however recommend to use *<sensor_name>_<sensor_serial_number>_<calibration_date>.json*); it should however comply with the information of the '*glider_sensor*' item of the deployment configuration file.

The file should contain:

- The EGO version involved (set in the '*EGO_format_version*' field),
- The sensor meta-data (through the '*SENSOR*'* fields),
- The meta-data of the parameters sampled by the sensor (through the '*PARAMETER*'* fields),
- Additional information to process derived parameters (through the *'CALIBRATION_COEFFICIENT'* field, see Annex G),
- Information on the parameters sampled by the sensor (in the *'parametersList'* field)

See Annex H to get a detailed description of the JSON sensor configuration files.

7 Decoder output data

The EGO file of the deployment is created in the DATA_DIRECTORY/*<glider_name>_<deployment_name>/* directory.

Its name is *<glider_name>_<deployment_name>_R.nc*.

The log file of the processing is stored in the LOG_DIRECTORY directory.

The ***gl_process_glider_rt*** program additionally generates and XML file in the XML_DIRECTORY directory.

Additional NetCDF profile files (generated when GENERATE_PROFILE_FILES=1) are created in the DATA_DIRECTORY/*<glider_name>_<deployment_name>/profiles/* directory. These files comply with the Argo V3.0 format.

Other sub-directories of the DATA_DIRECTORY/*<glider_name>_<deployment_name>/* directory are also created during a deployment processing:

- The '*mat*' directory contains a .mat file for each deployment measurement file processed. Each .mat file contains the measured data stored in a structure common to all glider type and measurement formats,
- The '*nc*' directory contains all the EGO files created from the .mat files contents.

'*profiles*', '*mat*' and '*nc*' sub-directories are deleted at the beginning of each new run of the decoder.

8 ANNEX A: description of EGO file processing steps

The generation of EGO file can be described in the following steps:

1 **Generate the (full) JSON file of the deployment**

Prior to any data processing, the decoder creates the (full) JSON file of the deployment.

This file, stored in the DATA_DIRECTORY/*<glider_name>_<deployment_name>* directory and called *deployment_<glider_name>_<deployment_name>/json*, is created from the EGO definition file format (EGO_FORMAT_JSON_FILE) and the JSON files specific to the deployment (stored in the DATA_DIRECTORY/*<glider_name>_<deployment_name>/json/* directory).

2 **Generate .mat files**

Each glider measurements file is parsed and the corresponding data are first stored in a common structure and then saved in a .mat file.

3 **Generate EGO files**

Each .mat file contents is processed and saved in one NetCDF file compliant with the EGO format.

4 **Compute PHASE and PHASE_NUMBER**

The PHASE and PHASE_NUMBER information is computed from each EGO file contents and stored in the file.

5 **Generate the final EGO file**

The EGO files are concatenated in a unique one: the final EGO file of the deployment.

6 **Compute estimated subsurface currents**

For a Slocum glider which provides the needed information ('*m_water_vx*' and '*m_water_vy*' data), the subsurface currents are estimated and stored in the final EGO file.

7 **Apply RTQC test on the EGO file time series**

The RTQC tests are applied on the EGO file time series.

8 **Generate profile files from the EGO file time series**

NetCDF profiles files are generated from the EGO file time series (according to PHASE and PHASE_NUMBER information).

The QC values, set during the previous step (#7), are duplicated in the profile data and additional specific RTQC tests are applied on profile data only.

9 ANNEX B: NetCDF profile file generation

If needed (when GENERATE_PROFILE_FILES=1) NetCDF profile files are generated from the EGO file time series and stored in the Argo V3.0 format.

During this process:

- The data to be reported in the profile files are selected according to PHASE (PHASE=1 for descending profile data and PHASE=4 for ascending profile data) and PHASE_NUMBER (different PHASE_NUMBER values should be in distinct output profile files).
- The date of the profile (JULD) is the mean date of all the measurements gathered in the profile.
- All the measurements are located (the GPS fixes are linearly interpolated for each measurement time), the profile location (JULD_LOCATION, LATITUDE, LONGITUDE) is set to the position of the measurement temporarily closest to the profile date (JULD).
- For Slocum and SeaGlider gliders, all the measurements are reported to the levels of the CTD ones. This is performed without any interpolation; for each CTD level, the timely closest measurement is duplicated and assigned to the CTD level.

10 ANNEX C: RTQC tests

Some of the Real Time Quality Control (RTQC) tests defined for the Argo project are apply to glider data.

These tests are documented in [RD2] and [RD3].

When NetCDF profiles files are generated, the QC set to the EGO data are reported in the profile data. Moreover, specific RTQC tests are applied on the profile data only.

10.1 RTQC tests applied on EGO time series

The following Argo tests are implemented in the decoder to be applied to the EGO file time series.

Test number	Test name
2	Impossible date test
3	Impossible location test
4	Position on land test
6	Global range test
7	Regional range test
9	Spike test
11	Gradient test
15	Grey list test
19	Deepest pressure test (*)
20	Questionable Argos position test
25	MEDian with a Distance (MEDD) test
57	DOXY specific test

(*) Note that the test #19 has been implemented for a nominal deepest pressure of 1000 dbar and for the CTD parameters only.

Some of these implemented tests can be ignored according to configuration information set in the upper part of the *gl_add_rtqc_to_ego_file.m* file (in testToPerformList variable).

10.2 RTQC tests applied on profile data

The following Argo tests are implemented in the decoder to be applied to the profile file data.

Test number	Test name
8	Pressure increasing tests
12	Digit rollover test
13	Stuck value test
14	Density inversion test

Some of these implemented tests can be ignored according to configuration information set in the upper part of the *gl_add_rtqc_to_profile_file.m* file (in testToPerformList variable).

11 ANNEX D: subsurface current estimates

If needed (when COMPUTE_SLOCUM_SUBSURFACE_CURRENT=1) estimates of the subsurface currents can be computed.

These estimates are based on the '*m_water_vx*' and '*m_water_vy*' information reported by the Slocum gliders.

This Matlab code (stored in the *gl_compute_subsurface_current.m* file) is based on a Python implementation provided by Lucas Merckelbach (Lucas.Merckelbach@hzg.de). The algorithm parameters are in the upper part of the file (they have been tuned according to Lucas recommendations).

12 ANNEX E: PHASE and PHASE_NUMBER determination

The PHASE determination is crucial for profile generation (directly based on PHASE=1 for descending profile data and PHASE=4 for ascending profile data). The implemented algorithm is based on the glider vertical speeds.

The PHASE_NUMBER determination is obvious once the PHASE is computed (PHASE_NUMBER is incremented each time the PHASE changes).

The PHASE and PHASE_NUMBER determination algorithm is stored in the *gl_set_phase.m* file. Its parameters (available in the upper part of the file) have been tuned after checks of the whole Coriolis glider data archive.

The following paragraph is dedicated to people who may be interested to understand the main principles of this algorithm.

12.1 Algorithm description

We use the dated (TIME parameter of the EGO file) immersion information (PRES or DEPTH parameters of the EGO NetCDF file).

The immersion is provided in meters (DEPTH) or dbar (PRES), the time in seconds (EPOCH or mission_time).

The algorithm is based on vertical speeds; we separate positive and negative vertical speeds using a **threshold computed from the data**.

To compute this threshold:

- We compute the vertical speed (in cm/s) over the dataset of a deployment (one or many Yos) and we create the corresponding histogram (with 0.5 cm/s steps),
- We don't consider the mode #0 (i.e. the [-0.5; 0.5] class),
- We sort the remaining classes according to their population,
- We only consider the set of sorted classes that enclose at least 80% (PART_OF_POINTS_FOR_MODE_SELECTION_PERCENT=80) of the whole dataset (taking mode #0 class population into account),
- The chosen threshold is 25% (PART_OF_MIN_MODE_FOR_THRESHOLD=1/4) of the less populated class selected at the previous step,

This threshold is used to assign descent/ascent PHASE to the corresponding points.

Ascent/descent phase sequences with less than 5 points (NB_BIN_FOR_PROFILE=5) are set to phase default value.

If more than 70% (MONO_DIRECTION_PROFILE_THRESHOLD=70/100) of the points have an ascent phase (resp. descent phase) the glider profiles in only one direction (during ascent resp.

descent):

- If this is the case: we compute the min duration of the profiles (*minDur*),
- If this is not the case: the min duration (*minDur*) of the profiles is initialized with a huge value.

The dataset is then split in slices with time gaps of more than *minDur/2*.

Note that, for gliders which profile in both directions, the dataset remains in only one slice.

We then process each slice to assign a phase value to the default phase value points.

All the consecutive points with a default phase value are assigned to the same phase value, which can be:

- If these points are at the beginning of the slice:
    - If the immersion is less than 2 dbar (THRESHOLD_PRES_FOR_SURF=2), their phase is set to *surface_drift*,
    - Otherwise their phase is set to *inconsistent* (for a descending profile) and *inflexion* (for an ascending profile).
- If these points are at the end of the slice:
    - If the immersion is less than 2 dbar (THRESHOLD_PRES_FOR_SURF=2), their phase is set to *surface_drift*,
    - Otherwise their phase is set to *inconsistent* (for an ascending profile) and *inflexion* (for a descending profile).
- Otherwise, the phase is set to:
    - *Inflexion* if its duration is less than 10 minutes (MAX_DURATION_OF_INFLEXION=10),
    - *Surface_drift* if the immersion is less than 2 dbar,
    - *Subsurface_drift* otherwise.

13 ANNEX F: additional tools

13.1 Visualization tool

13.1.1 Gl_trace_imm_vs_time

The ***gl_trace_imm_vs_time*** visualization tool can be used to visualize the glider pressure as a function of time (upper plot) and the glider vertical speed as a function of time (lower plot).

The color of the points depends on their corresponding PHASE as follows:

- Green for *surface_drift* PHASE,
- Light blue for *descent* PHASE,
- Cyan for *subsurface_drift* PHASE,
- Magenta for *inflexion* PHASE,
- Light red for *ascent* PHASE,
- Red for *inconsistent* PHASE,
- Black for default value PHASE (fill value).

13.1.1.1 Usage

The expected input parameter of this visualization tool is the full path name of the top directory of the deployments. It then looks for an EGO file in each sub-directory of the provided one.

The right and left arrows are used to change directory.

The up and down arrows are used to change EGO file (when multiple files are present in the current directory).

Examples:

gl_trace_imm_vs_time('C:/foo/bar/Gliders/archive/deployments')

If *C:/foo/bar/Gliders/archive/deployments* is the top directory of archived deployments, the final EGO file of the first deployment directory will be visualized. Then, the right arrow can be used to switch to the second one, etc…

gl_trace_imm_vs_time('C:/foo/bar/Gliders/archive/deployments/nearchos_mooseperseust02_09')

If one deployment path name is given, use the right arrow to switch to existing sub-directories until the '*nc*' one (which contains one EGO file for each input measurement file). Then use the down arrow to change the visualized EGO file.

13.1.2 Gl_trace_parameter

The ***gl_trace_parameter*** visualization tool can be used to visualize the glider timeseries of measurements. 2 parameters can be plot simultaneously.

The color of the points depends on their corresponding QC.

13.1.2.1 Usage

The expected input parameter of this visualization tool is the full path name of the top directory of the deployments. It then looks for an EGO file in each sub-directory of the provided one.

The right and left arrows are used to change directory.

The up and down arrows are used to change time sequence.

Type '*h*' to get the exhausted list of available commands.

13.2 Conversion tools

The ***nc_ego_2_csv*** tool is used to export (part of) one EGO file contents in a .csv file.

The tool configuration and input parameters are similar to the ***gl_process_glider*** program ones.

The ***nc_ego_2_kml*** tool is used to export one EGO file locations to a .kmz file. If generated profile files are also present, the enclosed locations are also exported to a separate .kmz file.

The tool configuration and input parameters are similar to the ***gl_process_glider*** program ones.

14 ANNEX G: derived parameter specification

The decoder is able to compute the following derived parameters:

- PSAL,
- DOXY,
- BBP700,
- BBP470,
- CDOM,
- CHLA,
- NITRATE,
- TURBIDITY,
- Linear function.

The definition of a derived parameter is done in the JSON configuration file of its associated sensor (see 15.1.2.5).

14.1 PSAL processing

The decoder can compute PSAL data from glider PRES, TEMP and CNDC data.

PSAL derived parameter doesn't need any additional calibration coefficient.

The PSAL processing is implemented in the "gl_compute_salinity.m" Matlab file.

14.2 DOXY processing

The decoder can compute DOXY data from DO sensor output data.

The processing methods are listed in [RD4].

They depend on:

- The sensor model,
- The input parameters,
- The sensor calibration method.

and have been named CASE_*SensorModelId_InputParamId_ComputationMethodId*.

The CASEs implemented so far are:

- **CASE_201_201_301 and CASE_202_201_301,**
- **CASE_201_202_202,**
- **CASE_202_205_302,**
- **CASE_202_205_303,**
- **CASE_202_205_304,**

- **CASE_202_204_303,**
- **CASE_102_207_206.**

DOXY derived parameter needs additional calibration coefficients which depend on each specific ‘CASE’ and should be provided in the *CALIBRATION_COEFFICIENT* field.

The DOXY processing is implemented in the following Matlab files:

- “gl_compute_DOXY_case_201_201_301.m” (also includes case_202_201_301),
- “gl_compute_DOXY_case_201_202_202.m”,
- “gl_compute_DOXY_case_202_205_302.m”,
- “gl_compute_DOXY_case_202_205_303.m”,
- “gl_compute_DOXY_case_202_205_304.m”,
- “gl_compute_DOXY_case_202_204_304.m”,
- “gl_compute_DOXY_case_102_207_206.m”.

14.2.1 Expected DOXY calibration coefficients

The following examples provide the calibration coefficients expected for each implemented ‘CASE’. The calibration coefficient values should be retrieved from the DO sensor calibration sheet.

See [RD4] to understand the meaning of each coefficient.

14.2.1.1 For CASE_201_201_301

```
"CALIBRATION_COEFFICIENT": [
          {
                "OPTODE_DOXY":
                      {
                            "Case": "201_201_301",
                            "DoxyCalibRefSalinity": 0
                      }
          }
   ],
```

14.2.1.2 For CASE_201_202_202

```
"CALIBRATION_COEFFICIENT": [
          {
                "OPTODE_DOXY":
                      {
                            "Case": "201_202_202",
                            "PhaseCoef0": -1.62468E-01,
                            "PhaseCoef1": 1.09760E+00,
                            "PhaseCoef2": 0.00000E+00,
                            "PhaseCoef3": 0.00000E+00,
                            "CCoef00": 5.02745E+03,
                            "CCoef01": -1.69644E+02,
                            "CCoef02": 3.47372E+00,
                            "CCoef03": -3.10884E-02,
                            "CCoef10": -2.72133E+02,
                            "CCoef11": 8.19642E+00,
                            "CCoef12": -1.68036E-01,
                            "CCoef13": 1.54063E-03,
                            "CCoef20": 5.94114E+00,
                                        "CCoef21": -1.57673E-01,
                                        "CCoef22": 3.27461E-03,
                                        "CCoef23": -3.08870E-05,
                                        "CCoef30": -6.03008E-02,
                                        "CCoef31": 1.39861E-03,
                                        "CCoef32": -2.98859E-05,
                                        "CCoef33": 2.90209E-07,
                                        "CCoef40": 2.33874E-04,
                                        "CCoef41": -4.68676E-06,
                                        "CCoef42": 1.05069E-07,
                                        "CCoef43": -1.04908E-09
                                }
                }
        ],
```

14.2.1.3 For CASE_202_205_302

```
    "CALIBRATION_COEFFICIENT": [
                {
                        "OPTODE_DOXY":
                                {
                                        "Case": "202_205_302",
                                        "PhaseCoef0": -1.44870E00,
                                        "PhaseCoef1": 1.01397E00,
                                        "PhaseCoef2": 0.000,
                                        "PhaseCoef3": 0.000,
                                        "FoilCoefA0": -2.8228020E-06,
                                        "FoilCoefA1": -6.7763060E-06,
                                        "FoilCoefA2": 1.8039070E-03,
                                        "FoilCoefA3": -1.9303320E-01,
                                        "FoilCoefA4": 6.2913340E-04,
                                        "FoilCoefA5": -2.9828240E-07,
                                        "FoilCoefA6": 1.0499040E+01,
                                        "FoilCoefA7": -5.4557400E-02,
                                        "FoilCoefA8": 9.2565000E-05,
                                        "FoilCoefA9": -4.3970450E-07,
                                        "FoilCoefA10": -2.9712800E+02,
                                        "FoilCoefA11": 2.2367310E+00,
                                        "FoilCoefA12": -7.9534540E-03,
                                        "FoilCoefA13": 0.000047795840,
                                        "FoilCoefB0": 7.5117270E-08,
                                        "FoilCoefB1": 3.6249380E+03,
                                        "FoilCoefB2": -3.7624690E+01,
                                        "FoilCoefB3": 2.4544850E-01,
                                        "FoilCoefB4": -3.3153260E-03,
                                        "FoilCoefB5": 4.7536400E-05,
                                        "FoilCoefB6": -4.8839130E-07,
                                        "FoilCoefB7": 0.0000000E+00,
                                        "FoilCoefB8": 0.0000000E+00,
                                        "FoilCoefB9": 0.0000000E+00,
                                        "FoilCoefB10": 0.0000000E+00,
                                        "FoilCoefB11": 0.0000000E+00,
                                        "FoilCoefB12": 0.0000000E+00,
                                        "FoilCoefB13": 0.0000000E+00,
                                        "FoilPolyDegT0": 1,
                                        "FoilPolyDegT1": 0,
                                        "FoilPolyDegT2": 0,
                                        "FoilPolyDegT3": 0,
                                        "FoilPolyDegT4": 1,
                                        "FoilPolyDegT5": 2,
                                        "FoilPolyDegT6": 0,
                                        "FoilPolyDegT7": 1,
                                        "FoilPolyDegT8": 2,
                                        "FoilPolyDegT9": 3,
                                        "FoilPolyDegT10": 0,
                                        "FoilPolyDegT11": 1,
                                        "FoilPolyDegT12": 2,
                                        "FoilPolyDegT13": 3,
                                        "FoilPolyDegT14": 4,
                                        "FoilPolyDegT15": 0,
                                        "FoilPolyDegT16": 1,
                                        "FoilPolyDegT17": 2,
                                        "FoilPolyDegT18": 3,
                                        "FoilPolyDegT19": 4,
                                        "FoilPolyDegT20": 5,
                                        "FoilPolyDegT21": 0,
                                        "FoilPolyDegT22": 0,
                                        "FoilPolyDegT23": 0,
                                        "FoilPolyDegT24": 0,
                                        "FoilPolyDegT25": 0,
                                        "FoilPolyDegT26": 0,
                                        "FoilPolyDegT27": 0,
                                        "FoilPolyDegO0": 4,
                                        "FoilPolyDegO1": 5,
                                        "FoilPolyDegO2": 4,
                                        "FoilPolyDegO3": 3,
                                        "FoilPolyDegO4": 3,
                                        "FoilPolyDegO5": 3,
                                        "FoilPolyDegO6": 2,
                                        "FoilPolyDegO7": 2,
                                        "FoilPolyDegO8": 2,
                                        "FoilPolyDegO9": 2,
                                        "FoilPolyDegO10": 1,
                                        "FoilPolyDegO11": 1,
                                        "FoilPolyDegO12": 1,
                                        "FoilPolyDegO13": 1,
                                        "FoilPolyDegO14": 1,
                                        "FoilPolyDegO15": 0,
                                        "FoilPolyDegO16": 0,
                                        "FoilPolyDegO17": 0,
                                        "FoilPolyDegO18": 0,
                                        "FoilPolyDegO19": 0,
                                        "FoilPolyDegO20": 0,
                                        "FoilPolyDegO21": 0,
                                        "FoilPolyDegO22": 0,
                                        "FoilPolyDegO23": 0,
                                        "FoilPolyDegO24": 0,
                                        "FoilPolyDegO25": 0,
                                        "FoilPolyDegO26": 0,
                                        "FoilPolyDegO27": 0,
                                        "TempCoef0": 22.985500000000000,
                                        "TempCoef1": -0.031128000000000,
                                        "TempCoef2": 0.000002952920000,
                                        "TempCoef3": -0.000000004222650,
                                        "TempCoef4": 0,
                                        "TempCoef5": 0
                        }
                }
        ],
```

14.2.1.4 For CASE_202_205_303

```
    "CALIBRATION_COEFFICIENT": [
                {
                        "OPTODE" :
                                {
                                        "Case" : "202_205_303",
                                        "PhaseCoef0" : 0,
                                        "PhaseCoef1" : 1,
                                        "PhaseCoef2" : 0,
                                        "PhaseCoef3" : 0,
                                        "FoilCoefA0" : -0.000002988314,
                                        "FoilCoefA1" : -6.137785E-06,
                                        "FoilCoefA2" : 1.684659E-03,
                                        "FoilCoefA3" : -1.857173E-01,
                                        "FoilCoefA4" : 6.784399E-04,
                                        "FoilCoefA5" : -5.597908E-07,
                                        "FoilCoefA6" : 1.040158E+01,
                                        "FoilCoefA7" : -5.986907E-02,
                                        "FoilCoefA8" : 1.360425E-04,
                                        "FoilCoefA9" : -4.776977E-07,
                                        "FoilCoefA10" : -3.032937E+02,
                                        "FoilCoefA11" : 2.530496E+00,
                                        "FoilCoefA12" : -1.267045E-02,
                                        "FoilCoefA13" : 1.040454E-04,
                                        "FoilCoefB0" : -0.000000356039,
                                        "FoilCoefB1" : 3.816713E+03,
                                        "FoilCoefB2" : -4.475507E+01,
                                        "FoilCoefB3" : 4.386164E-01,
                                        "FoilCoefB4" : -7.146342E-03,
                                        "FoilCoefB5" : 8.906236E-05,
                                        "FoilCoefB6" : -6.343012E-07,
                                        "FoilCoefB7" : 0.000000E+00,
                                        "FoilCoefB8" : 0.000000E+00,
                                        "FoilCoefB9" : 0.000000E+00,
                                        "FoilCoefB10" : 0.000000E+00,
                                        "FoilCoefB11" : 0.000000E+00,
                                        "FoilCoefB12" : 0.000000E+00,
                                        "FoilCoefB13" : 0.000000E+00,
                                        "FoilPolyDegT0" : 1,
                                        "FoilPolyDegT1" : 0,
                                        "FoilPolyDegT2" : 0,
                                        "FoilPolyDegT3" : 0,
                                        "FoilPolyDegT4" : 1,
                                        "FoilPolyDegT5" : 2,
                                        "FoilPolyDegT6" : 0,
                                        "FoilPolyDegT7" : 1,
                                        "FoilPolyDegT8" : 2,
                                        "FoilPolyDegT9" : 3,
                                        "FoilPolyDegT10" : 0,
                                        "FoilPolyDegT11" : 1,
                                        "FoilPolyDegT12" : 2,
                                        "FoilPolyDegT13" : 3,
                                        "FoilPolyDegT14" : 4,
                                        "FoilPolyDegT15" : 0,
                                        "FoilPolyDegT16" : 1,
                                        "FoilPolyDegT17" : 2,
                                        "FoilPolyDegT18" : 3,
                                        "FoilPolyDegT19" : 4,
                                        "FoilPolyDegT20" : 5,
                                        "FoilPolyDegT21" : 0,
                                        "FoilPolyDegT22" : 0,
                                        "FoilPolyDegT23" : 0,
                                        "FoilPolyDegT24" : 0,
                                        "FoilPolyDegT25" : 0,
                                        "FoilPolyDegT26" : 0,
                                        "FoilPolyDegT27" : 0,
                                        "FoilPolyDegO0" : 4,
                                        "FoilPolyDegO1" : 5,
                                        "FoilPolyDegO2" : 4,
                                        "FoilPolyDegO3" : 3,
                                        "FoilPolyDegO4" : 3,
                                        "FoilPolyDegO5" : 3,
                                        "FoilPolyDegO6" : 2,
                                        "FoilPolyDegO7" : 2,
                                        "FoilPolyDegO8" : 2,
                                        "FoilPolyDegO9" : 2,
                                        "FoilPolyDegO10" : 1,
                                        "FoilPolyDegO11" : 1,
                                        "FoilPolyDegO12" : 1,
                                        "FoilPolyDegO13" : 1,
                                        "FoilPolyDegO14" : 1,
                                        "FoilPolyDegO15" : 0,
                                        "FoilPolyDegO16" : 0,
                                        "FoilPolyDegO17" : 0,
                                        "FoilPolyDegO18" : 0,
                                        "FoilPolyDegO19" : 0,
                                        "FoilPolyDegO20" : 0,
                                        "FoilPolyDegO21" : 0,
                                        "FoilPolyDegO22" : 0,
                                        "FoilPolyDegO23" : 0,
                                        "FoilPolyDegO24" : 0,
                                        "FoilPolyDegO25" : 0,
                                        "FoilPolyDegO26" : 0,
                                        "FoilPolyDegO27" : 0,
                                        "TempCoef0" : 2.230390E+01,
                                        "TempCoef1" : -0.0308968,
                                        "TempCoef2" : 2.91186e-006,
                                        "TempCoef3" : -4.20346e-009,
                                        "TempCoef4" : 0.00000E00,
                                        "TempCoef5" : 0.00000E00,
                                        "ConcCoef0" : 1.93885,
                                        "ConcCoef1" : 1.07626
                         }
                  }
          ],
```

14.2.1.5 For CASE_202_205_304

```
    "CALIBRATION_COEFFICIENT": [
        {
            "OPTODE_DOXY": [
                {
                    "Case": "202_205_304",
                    "SVUFoilCoef0" : 2.78413E-03,
                    "SVUFoilCoef1" : 1.17229E-04,
                    "SVUFoilCoef2" : 2.44348E-06,
                    "SVUFoilCoef3" : 2.29569E02,
                    "SVUFoilCoef4" : -3.21309E-01,
                    "SVUFoilCoef5" : -5.64948E01,
                    "SVUFoilCoef6" : 4.56886E00,
                    "PhaseCoef0" : 0,
                    "PhaseCoef1" : 1,
                    "PhaseCoef2" : 0,
                    "PhaseCoef3" : 0
                }
            ]
        }
    ],
```

14.2.1.6 For CASE_202_204_304

```
    "CALIBRATION_COEFFICIENT": [
        {
            "OPTODE_DOXY": [
                {
                    "Case": "202_204_304",
                    "SVUFoilCoef0" : 2.78413E-03,
                    "SVUFoilCoef1" : 1.17229E-04,
                    "SVUFoilCoef2" : 2.44348E-06,
                    "SVUFoilCoef3" : 2.29569E02,
                    "SVUFoilCoef4" : -3.21309E-01,
                    "SVUFoilCoef5" : -5.64948E01,
                    "SVUFoilCoef6" : 4.56886E00,
                    "PhaseCoef0" : 0,
                    "PhaseCoef1" : 1,
                    "PhaseCoef2" : 0,
                    "PhaseCoef3" : 0
                }
            ]
        }
    ],
```

14.2.1.7 For CASE_102_207_206

```
    "CALIBRATION_COEFFICIENT": [
        {
            "OPTODE_DOXY": [
                {
                    "Case": "102_207_206",
                    "Soc" : 2.9909e-004,
                    "FOffset" : -812.55,
                    "CoefA" : 3.3742e-003,
                    "CoefB" : 1.5381e-004,
                    "CoefC" : -2.4764e-006,
                    "CoefE" : 0.036
                }
            ]
        }
    ],
```

14.3 BBP470 processing

The decoder can compute BBP470 data from BETA_BACKSCATTERING470 data.

The processing method is detailed in [RD5] and needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
                "BACKSCATTERINGMETER_BBP470":
                        {
                                "ScaleFactBBP470": 0.00001065,
                                "DarkCountBBP470": 50,
                                "KhiCoefBBP470": 1.076,
                                "MeasAngleBBP470": 124
                        }
        }
]
```

The BBP470 processing is implemented in the “gl_compute_BBP.m” Matlab file.

14.4 BBP700 processing

The decoder can compute BBP700 data from BETA_BACKSCATTERING700 data.

The processing method is detailed in [RD5] and needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
                "BACKSCATTERINGMETER_BBP700":
                        {
                                "ScaleFactBBP700": 0.000001868,
                                "DarkCountBBP700": 50,
                                "KhiCoefBBP700": 1.076,
                                "MeasAngleBBP700": 124
                        }
        }
]
```

The BBP700 processing is implemented in the “gl_compute_BBP.m” Matlab file.

14.5 CDOM processing

The decoder can compute CDOM data from FLUORESCENCE_CDOM data.

The processing method is detailed in [RD7] and needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
                "FLUOROMETER_CDOM":
                        {
                                "ScaleFactCDOM": 0.0905,
                                "DarkCountCDOM": 40
                        }
        }
]
```

The CDOM processing is implemented in the “gl_compute_CDOM.m” Matlab file.

14.6 CHLA processing

The decoder can compute CHLA data from FLUORESCENCE_CHLA data.

The processing method is detailed in [RD6] and needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
                "FLUOROMETER_CHLA":
                        {
                                "ScaleFactCHLA": 0.0073,
                                "DarkCountCHLA": 44
                        }
        }
]
```

The CHLA processing is implemented in the “gl_compute_CHLA.m” Matlab file.

14.7 NITRATE processing

The decoder can compute NITRATE data from MOLAR_NITRATE data.

The processing method is detailed in [RD8] and doesn’t need any additional coefficients.

The NITRATE processing is implemented in the “gl_compute_NITRATE.m” Matlab file.

14.8 TURBIDITY processing

The decoder can compute TURBIDITY data from SIDE_SCATTERING_TURBIDITY data.

The processing method is detailed in [TBD] and needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
                "BACKSCATTERINGMETER_TURBIDITY":
                        {
                                "ScaleFactTURBIDITY": 0.0054,
                                "DarkCountTURBIDITY": 46
                        }
        }
]
```

The TURBIDITY processing is implemented in the “gl_compute_TURBIDITY.m” Matlab file.

14.9 Linear function processing

The decoder can apply a linear function (f(X) = X * SLOPE + OFFSET) to any input data.

In this case, both “ego_variable_name” and “glider_variable_name” should be set in the configuration file of the sensor. The linear function is then applied to “glider_variable_name” data before being stored in “ego_variable_name” variable.

The processing method needs the following coefficients:

```
"CALIBRATION_COEFFICIENT": [
        {
            " concerned_sensor_name": [
                {
                    "Case": "slope_offset",
                    "SlopeValue" : 0.1,
                    "OffsetValue" : 0
                }
            ]
        }
    ],
```

Note that this specific derived parameter processing can be used for sensor output that doesn't fit the associated EGO variable unit.

Example:

If RBR conductivity is provided in mS/cm, it cannot be stored in "CNDC" EGO variable which unit is mhos/m. The f(X) = X * 0.1 + 0 function should first be applied to RBR conductivity. This should be done in the RBR sensor configuration file with the following lines.

```
"CALIBRATION_COEFFICIENT": [

    {

        "CTD_CNDC":

            {

                "Case": "slope_offset",

                "SlopeValue": 0.1,

                "OffsetValue": 0

            }

    }

],

"parametersList": [

    {

        "ego_variable_name"     : "CNDC",

        "glider_variable_name"  : "LEGATO_CONDUCTIVITY",

        "comment"               : "",

        "cell_methods"          : "",

        "reference_scale"       : "",

        "derivation_equation"   : "r",

        "derivation_coefficient": "",

        "derivation_comment"    : "",

        "derivation_date"       : "20210331000000",

        "processing_id"         : "slope_offset"

    },

…
```

15 ANNEX H: detailed description of deployment and sensor JSON files

To process a deployment, the decoder needs:

- The JSON file that describes the EGO format. This file is part of the decoder distribution (its location is set by the EGO_FORMAT_JSON_FILE configuration variable) and should not be modified by the user,
- The JSON files that provide meta-data on the deployment and rules to access, compute and store glider measurements,
- The glider measurements.

This annex describes the contents of the JSON files used to store the deployment meta-data and the rules to process glider measurements.

Two types of JSON files should be provided to the decoder for each deployment:

- One configuration file **for the deployment** itself,
- One configuration file **for each sensor** mounted on the glider during the deployment.

Note that all sensor information can be stored in the same file, however, we recommend to use one file for each sensor.

These two file types are linked through the *'glider_sensor'* section of the deployment configuration file (see below).

When processing NetCDF files from SeaGlider glider, templates (referring to NetCDF files contents), can be used in JSON files. Two kinds of templates can be used:

- A template that refers to a global attribute is expressed as <nc.global_att.*name_of_the_global_attribute*>;
- A template that refers to a data value is expressed as <nc.data.*name_of_the_data_variable*>.

Examples:

- `"platform_code": "<nc.global_att.platform_id>"` means that the value of the "platform_id" global attribute of the first NetCDF input file will be used to fill the "platform_code" information of the (deployment) JSON file;
- `"SVUFoilCoef0": "<nc.data.sg_cal_optode_SVUCoef0>"` means that the value of the "sg_cal_optode_SVUCoef0" variable of the first NetCDF input file will be used to fill the "SVUFoilCoef0" information of the (sensor) JSON file;
- `"derivation_comment": "Data copied in SeaGlider NetCDF file (base_station_version=<nc.global_att.base_station_version>; base_station_micro_version=<nc.global_att.base_station_micro_version> ; quality_control_version=<nc.global_att.quality_control_version>."` means that the value of the "base_station_version", "base_station_micro_version" and "quality_control_version" global attributes of the first NetCDF input file will be used in the "derivation_comment" information of the (sensor) JSON file

15.1.1 The configuration file of the deployment

The JSON file of the deployment should have the same name as the deployment directory (we recommend to use *<glider_name>_<deployment_name>.json*).

This file should contain the following sections:

- '*EGO_format_version*',
- '*global_attributes*',
- '*glider_characteristics*',
- '*glider_deployment*',
- '*coordinate_variables*',
- '*glider_sensor*'.

Example of empty JSON deployment file:

```
{
    "EGO_format_version": "1.4",

    "global_attributes": {
        "platform_code": "",
        "wmo_platform_code": "",
        "references": "",
        "comment": "",
        "title": "",
        "summary": "",
        "abstract": "",
        "keywords": "",
        "area": "",
        "institution": "",
        "institution_references": "",
        "sdn_edmo_code": "",
        "authors": [
            {
                "first_name": "",
                "last_name": "",
                "email": "",
                "orcid": "",
                "affiliations": ["", ""]
            },
            {
                "first_name": "",
                "last_name": "",
                "email": "",
                "orcid": "",
                "affiliations": ""
            }
        ],
        "data_assembly_center": "",
        "principal_investigator": "",
        "principal_investigator_email": "",
        "project_name": "",
        "observatory": "",
        "deployment_code": "",
        "deployment_label": "",
        "doi": "",
        "citation": "",
        "update_interval": ""
    },

    "glider_characteristics": {
        "PLATFORM_FAMILY": "",
        "PLATFORM_TYPE": "",
        "PLATFORM_MAKER": "",
        "GLIDER_SERIAL_NO": "",
        "GLIDER_OWNER": "",
        "OPERATING_INSTITUTION": "",
        "WMO_INST_TYPE": "",
        "POSITIONING_SYSTEM": ["", ""],
        "TRANS_SYSTEM": "",
        "TRANS_SYSTEM_ID": "",
        "TRANS_FREQUENCY": "",
        "BATTERY_TYPE": "",
        "BATTERY_PACKS": "",
        "SPECIAL_FEATURES": "",
        "FIRMWARE_VERSION_NAVIGATION": "",
        "FIRMWARE_VERSION_SCIENCE": "",
        "GLIDER_MANUAL_VERSION": "",
        "ANOMALY": "",
        "CUSTOMIZATION": "",
        "DAC_FORMAT_ID": ""
    },

    "glider_deployment": {
        "DEPLOYMENT_START_DATE": "",
        "DEPLOYMENT_START_LATITUDE": null,
        "DEPLOYMENT_START_LONGITUDE": null,
        "DEPLOYMENT_START_QC": 0,
        "DEPLOYMENT_PLATFORM": "",
        "DEPLOYMENT_CRUISE_ID": "",
        "DEPLOYMENT_REFERENCE_STATION_ID": "",
        "DEPLOYMENT_END_DATE": "",
        "DEPLOYMENT_END_LATITUDE": null,
        "DEPLOYMENT_END_LONGITUDE": null,
        "DEPLOYMENT_END_QC": 0,
        "DEPLOYMENT_END_STATUS": "",
        "DEPLOYMENT_OPERATOR": ""
    },

    "coordinate_variables": [
        {
            "glider_variable_name": "",
            "ego_variable_name": "TIME"
        },
        {
            "glider_variable_name": "",
            "ego_variable_name": "LATITUDE"
        },
        {
            "glider_variable_name": "",
            "ego_variable_name": "LONGITUDE"
        }
    ],

    "glider_sensor": [
        {
            "sensor_file_name": ""
        },
        {
            "sensor_file_name": ""
        }
    ]
}
```

15.1.1.1 ‘EGO_format_version’ section

This section is used to provide the version of the EGO file that will be generated. It is currently set to “1.4”.

15.1.1.2 ‘global_attributes’ section

The definition of each item of this section can be found in the chapter 2.2 of the EGO format reference manual [RD1].

Concerning “authors”; the JSON file provides additional fields (to store “email”, “orcid” and “affiliations”) needed by the DOI landing page. The decoder will use only “first_name” and “last_name” of each “authors” entry to generate the comma separated list that will be stored in the “authors” global attribute of the EGO format.

15.1.1.3 'glider_characteristics' section

The definition of each item of this section can be found in the chapter 2.4.2 of the EGO format reference manual [RD1].

15.1.1.4 'glider_deployment' section

The definition of each item of this section can be found in the chapter 2.4.3 of the EGO format reference manual [RD1].

Note that numerical values are expected for items "DEPLOYMENT_START_LATITUDE", "DEPLOYMENT_START_LONGITUDE", "DEPLOYMENT_START_QC", "DEPLOYMENT_END_LATITUDE", "DEPLOYMENT_END_LONGITUDE" and "DEPLOYMENT_END_QC".

15.1.1.5 'coordinate_variables' section

This section is used to provide rules to access glider data (through "glider_variable_name" fields) to fill coordinate variables: TIME, LATITUDE and LONGITUDE.

Link to data is mandatory for TIME; its associated "glider_variable_name" field should be set to the glider variable name of the measurements timestamps.

Example:

- For a 'slocum': "m_present_time",
- For a 'seaglider': "elaps_t",
- For a 'seaexplorer': "PLD_REALTIMECLOCK".

Link to data is not mandatory for LATITUDE and LONGITUDE.

If the user wants to fill LATITUDE and LONGITUDE variables with linearly interpolated values of GPS fixes only, associated "glider_variable_name" fields should not be set ("glider_variable_name": "").

If the user wants to use latitude and longitude data provided by the glider, the associated "glider_variable_name" fields should be set accordingly.

Example:

- For a 'slocum': "m_lat" and "m_lon",
- For a 'seaglider': "dive_pos_lat_dd_v" and "dive_pos_lon_dd_v",
- For a 'seaexplorer': "NAV_LATITUDE": "NAV_LONGITUDE".

Note that, in that second case, the remaining unset LATITUDE and LONGITUDE data are filled with linearly interpolated values of GPS and glider fixes.

<u>**Note on GPS variables**</u>

In the EGO format, GPS variables are stored in dedicated variables (TIME_GPS, LATITUDE_GPS and LONGITUDE_GPS).

These variables are filled by the decoder from input data that depend on the glider type:

- For a 'slocum': the glider variables “m_present_time”, “m_gps_lat” and “m_gps_lon” are used,

- For a 'seaglider':
    - with *.bpo or *.pro files, the decoder uses the GPS information, available in the data file header;
    - with p*.eng files, the decoder uses the GPS information, available in the associated p*.log files;
    - with p*.nc files, the decoder uses the GPS information, available in the "log_gps_time", "log_gps_lon" and "log_gps_lat" variables of the input NetCDF files.
- For a 'seaexplorer': the decoder selects locations with 'NavState' and 'NAV_RESOURCE' set to 116.

15.1.1.6 'glider_sensor' section

This section provides the links to the JSON sensor configuration files. Each file mentioned here should be present in the DATA_DIRECTORY/*<glider_name>_<deployment_name>/json* directory.

15.1.2 The configuration file of the sensor

We recommend to create one file for each sensor mounted on the glider during the deployment. It's up to each user to set the names of these files (we however recommend to use *<sensor_name>_<sensor_serial_number>_<calibration_date>.json*).

This file should contain the following sections:

- '*EGO_format_version*',
- Sensor meta-data,
- Parameter meta-data
- *'CALIBRATION_COEFFICIENT'*,
- *'parametersList'*.

Example of empty JSON sensor file (for all gliders but a SeaGlider one with NetCDF input files):

```
{
    "EGO_format_version": "1.4",

    "SENSOR": ["", "", ""],
    "SENSOR_MAKER": ["", "", ""],
    "SENSOR_MODEL": ["", "", ""],
    "SENSOR_SERIAL_NO": ["", "", ""],
    "SENSOR_MOUNT": ["", "", ""],
    "SENSOR_ORIENTATION": ["", "", ""],

    "PARAMETER": ["", "", "", ""],
    "PARAMETER_SENSOR": ["", "", "", ""],
    "PARAMETER_DATA_MODE": ["", "", "", ""],
    "PARAMETER_UNITS": ["", "", "", ""],
    "PARAMETER_ACCURACY": ["", "", "", ""],
    "PARAMETER_RESOLUTION": ["", "", "", ""],

    "CALIBRATION_COEFFICIENT": [],

    "parametersList": [
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        }
    ]
}
```

Example of empty JSON sensor file (for a SeaGlider with NetCDF input files):

In such case, a “glider_adjusted_variable_name” item has been added to each parameter structure. It can be used to import adjusted parameter data from NetCDF input file.

Note also, that in such case the QC associated data are also imported from NetCDF input file.

```
{
    "EGO_format_version": "1.4",

    "SENSOR": ["", "", ""],
    "SENSOR_MAKER": ["", "", ""],
    "SENSOR_MODEL": ["", "", ""],
    "SENSOR_SERIAL_NO": ["", "", ""],
    "SENSOR_MOUNT": ["", "", ""],
    "SENSOR_ORIENTATION": ["", "", ""],

    "PARAMETER": ["", "", "", ""],
    "PARAMETER_SENSOR": ["", "", "", ""],
    "PARAMETER_DATA_MODE": ["", "", "", ""],
    "PARAMETER_UNITS": ["", "", "", ""],
    "PARAMETER_ACCURACY": ["", "", "", ""],
    "PARAMETER_RESOLUTION": ["", "", "", ""],

    "CALIBRATION_COEFFICIENT": [],

    "parametersList": [
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "glider_adjusted_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "glider_adjusted_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "glider_adjusted_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "",
            "glider_variable_name": "",
            "glider_adjusted_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        }
    ]
}
```

15.1.2.1 ‘EGO_format_version’ section

This section is used to provide the version of the EGO file that will be generated. It is currently set to “1.4”.

15.1.2.2 Sensor meta-data

Sensor meta-data can be set through “SENSOR”, “SENSOR_MAKER”, “SENSOR_MODEL”, “SENSOR_SERIAL_NO”, “SENSOR_MOUNT” and “SENSOR_ORIENTATION” items.

The definition of each item can be found in the chapter 2.4.4.1 of the EGO format reference manual [RD1].

From the format specification, one can see that they all share the same N_SENSOR dimension (N_SENSOR=3 in the provided example above).

15.1.2.3 Parameter meta-data

Parameter meta-data can be set through “PARAMETER”, “PARAMETER_SENSOR”, “PARAMETER_DATA_MODE”, “PARAMETER_UNITS”, “PARAMETER_ACCURACY” and “PARAMETER_RESOLUTION” items.

The definition of each item can be found in the chapter 2.4.4.2 of the EGO format reference manual [RD1].

From the format specification, one can see that they all share the same N_PARAM dimension (N_PARAM=4 in the provided example above).

Note also that the “PARAMETER_UNITS” is the units of the provided “PARAMETER_ACCURACY” and “PARAMETER_RESOLUTION” information **and not the units of the parameter data** (which is already defined by the “PARAMETER” information).

15.1.2.4 ‘CALIBRATION_COEFFICIENT’ section

This section is used to provide additional information to compute derived parameters.

The decoder is able to compute derived parameters from (see Annex G); the needed equations are implemented in the decoder and the associated calibration coefficients (that depend on each sensor) should be provided in this section.

The *‘CALIBRATION_COEFFICIENT’* section should have one sub-section for each sensor for which one needs to compute derived parameters. Each sub-section is named by the concerned sensor name.

Example of ‘CALIBRATION_COEFFICIENT’ section:

```
"CALIBRATION_COEFFICIENT": [
    {
        "BACKSCATTERINGMETER_BBP700":
            {
                "ScaleFactBBP700": 0.000001868,
                "DarkCountBBP700": 50,
                "KhiCoefBBP700": 1.076,
                "MeasAngleBBP700": 124
            }
    },
    {
        "FLUOROMETER_CDOM":
            {
                "ScaleFactCDOM": 0.0905,
                "DarkCountCDOM": 40
            }
    },
    {
        "FLUOROMETER_CHLA":
            {
                "ScaleFactCHLA": 0.0073,
                "DarkCountCHLA": 44
            }
        }
    ],
```

In the case of DOXY derived parameter, more than one set of calibration coefficients could be provided.

Example of ‘CALIBRATION_COEFFICIENT’ section for DOXY derived parameter:

```
    "CALIBRATION_COEFFICIENT": [
        {
            "OPTODE_DOXY": [
                {
                    "Case": "201_201_301",
                    "DoxyCalibRefSalinity": 0
                },
                {
                    "Case": "201_202_202",
                    "PhaseCoef0": -3.22792E-01,
                    "PhaseCoef1": 1.10079E00,
                    "PhaseCoef2": 0.00000E+00,
                    "PhaseCoef3": 0.00000E+00,
                    "CCoef00": 5.02745E+03,
                    "CCoef01": -1.69644E+02,
                    "CCoef02": 3.47372E+00,
                    "CCoef03": -3.10884E-02,
                    "CCoef10": -2.72133E+02,
                    "CCoef11": 8.19642E+00,
                    "CCoef12": -1.68036E-01,
                    "CCoef13": 1.54063E-03,
                    "CCoef20": 5.94114E+00,
                    "CCoef21": -1.57673E-01,
                    "CCoef22": 3.27461E-03,
                    "CCoef23": -3.08870E-05,
                    "CCoef30": -6.03008E-02,
                    "CCoef31": 1.39861E-03,
                    "CCoef32": -2.98859E-05,
                    "CCoef33": 2.90209E-07,
                    "CCoef40": 2.33874E-04,
                    "CCoef41": -4.68676E-06,
                    "CCoef42": 1.05069E-07,
                    "CCoef43": -1.04908E-09
                }
            ]
        }
    ],
```

in such case, the method to be used will be set by the “case” item (see below).

15.1.2.5 ‘parametersList’ section

This section provides information used by the decoder to fill parameter data.

We have one sub-section for each parameter, it contains the following items:

- “ego_variable_name”: used to set the parameter name,
- “glider_variable_name”: used to set the glider variable name associated to the parameter. **Note that, if any, ‘.’ characters should be replaced by ‘_’ in these glider variable names (Ex: replace “gpctd.Pressure” by ‘gpctd_Pressure”),**
- **In the case of NetCDF input files from a SeaGlider glider**: “glider_adjusted_variable_name”: used to set the glider variable name associated to the adjusted parameter,
- “comment”: used to set the comment that will be stored in the <PARAM>:comment attribute,
- “cell_methods”: used to set the cell method that will be stored in the <PARAM>:cell_methods attribute,

- "reference_scale": used to set the reference scale that will be stored in the <PARAM>:reference_scale attribute,
- "derivation_equation": used to set the derivation equation that will be stored in the DERIVATION_EQUATION variable,
- "derivation_coefficient": used to set the derivation coefficients that will be stored in the DERIVATION_COEFFICIENT variable,
- "derivation_comment": used to set the derivation comment that will be stored in the DERIVATION_COMMENT variable,
- "derivation_date": used to set the derivation comment that will be stored in the DERIVATION_DATE variable,
- "processing_id": when multiple methods can be used to derive a parameter, this item is used to set the chosen one.

Example of 'parametersList' section:

```
    "parametersList": [
        {
            "ego_variable_name": "PRES",
            "glider_variable_name": "sci_water_pressure",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "pressure=PA0+PA1*n+PA2*n2;
n=x*PTCB0/(PTCB0+PTCB1*t+PTCB2*t^2; x=pressure output-PTAC0-PTCA1*t-PTCA2*t^2; y=thermistor
output; t=PTEMPA0+PTEMPA1*y+PTE",
            "derivation_coefficient": "PA0=6.349902e-002,PA1=5.029955e-003,PA2=-2.552827e-
011,PTHA0=-7.065576e+001,PTHA1=4.993609e-002,PTHA2=-2.435890e-007,PTCA0=5.247073e+005,PTCA1=-
4.163563e-001,PTCA2=5.399231e-002,PTCB0=2.519913e+001,PTCB1=2.250000e-004,PTCB2=0.000000e+00",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        },
        {
            "ego_variable_name": "TEMP",
            "glider_variable_name": "sci_water_temp",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "T=1/{a0+a1[ln(n)]+a2[ln^2(n)]+a3[ln^3(n)]}-273.15",
            "derivation_coefficient": "g=-7.884375e-005,h=2.991331e-004,i=-3.750703e-
006,j=1.825182e-007",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        },
        {
            "ego_variable_name": "CNDC",
            "glider_variable_name": "sci_water_cond",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Conductivity=(g+hf^2+if^3+jf^4)/(1+deltat+epsilonp)
Siemens/meter t=temperature(degreeC);p=pressure(decibars);delta=CTcor;epsilon=CPcor",
            "derivation_coefficient": "g=-1.027870e+000,h=1.731628e-001,i=-4295554e-
004,j=6.189649e-005,CPcor=-9.5700e-008,CTcor=3.2500e-006,WBOTC=9.2164e-007",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        }
    ]
```

How to define a derived parameter?

When a derived parameter is needed, one should **let the "glider_variable_name" item empty**.

In that case, the decoder:

1. Checks that all needed information are available:
   - input parameter data,
   - calibration coefficients (when needed),
2. Computes and stores the wanted derived parameter,
3. Fills the associated DERIVATION_EQUATION, DERIVATION_COEFFICIENT, DERIVATION_COMMENT and DERIVATION_DATE parameters.

Example of PSAL derived parameter definition:

```
    "CALIBRATION_COEFFICIENT": [],

    "parametersList": [
        {
            "ego_variable_name": "PRES",
            "glider_variable_name": "sci_water_pressure",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "pressure=PA0+PA1*n+PA2*n2;
n=x*PTCB0/(PTCB0+PTCB1*t+PTCB2*t^2; x=pressure output-PTAC0-PTCA1*t-PTCA2*t^2; y=thermistor
output; t=PTEMPA0+PTEMPA1*y+PTE",
            "derivation_coefficient": "PA0=6.349902e-002,PA1=5.029955e-003,PA2=-2.552827e-
011,PTHA0=-7.065576e+001,PTHA1=4.993609e-002,PTHA2=-2.435890e-007,PTCA0=5.247073e+005,PTCA1=-
4.163563e-001,PTCA2=5.399231e-002,PTCB0=2.519913e+001,PTCB1=2.250000e-004,PTCB2=0.000000e+00",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        },
        {
            "ego_variable_name": "TEMP",
            "glider_variable_name": "sci_water_temp",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "T=1/{a0+a1[ln(n)]+a2[ln^2(n)]+a3[ln^3(n)]}-273.15",
            "derivation_coefficient": "g=-7.884375e-005,h=2.991331e-004,i=-3.750703e-
006,j=1.825182e-007",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        },
        {
            "ego_variable_name": "CNDC",
            "glider_variable_name": "sci_water_cond",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Conductivity=(g+hf^2+if^3+jf^4)/(1+deltat+epsilonp)
Siemens/meter t=temperature(degreeC);p=pressure(decibars);delta=CTcor;epsilon=CPcor",
            "derivation_coefficient": "g=-1.027870e+000,h=1.731628e-001,i=-4295554e-
004,j=6.189649e-005,CPcor=-9.5700e-008,CTcor=3.2500e-006,WBOTC=9.2164e-007",
            "derivation_comment": "Capteur de pression calibr\u0106\u00a9 le 13/05/2011",
            "derivation_date": "20110518184114",

            "processing_id": ""
        },
        {
            "ego_variable_name": "PSAL",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Not measured by the glider. Calculated by Coriolis",
            "derivation_coefficient": "Not measured by the glider. Calculated by Coriolis",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        }
    ]
```

In this example, we ask the decoder to compute PSAL derived parameter.

For that it needs PRES, TEMP and CNDC parameters (which are previously defined) and no additional calibration coefficients.

Example of BBP700 derived parameter definition:

```
    "CALIBRATION_COEFFICIENT": [
        {
            "BACKSCATTERINGMETER_BBP700":
                {
                    "ScaleFactBBP700": 0.000001868,
                    "DarkCountBBP700": 50,
                    "KhiCoefBBP700": 1.076,
                    "MeasAngleBBP700": 124
                }
        }
    ],

    "parametersList": [
        {
            "ego_variable_name": "BETA_BACKSCATTERING700",
            "glider_variable_name": "sci_flbbcd_xx",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        },
        {
            "ego_variable_name": "BBP700",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Not measured by the glider. Calculated by Coriolis",
            "derivation_coefficient": "Not measured by the glider. Calculated by Coriolis",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": ""
        }
    ]
```

In this example, we ask the decoder to compute BBP700 derived parameter.

For that it needs BETA_BACKSCATTERING700 parameter (which is previously defined, if sci_flbbcd_xx is its associated glider variable name) and associated calibration coefficients provided in ‘CALIBRATION_COEFFICIENT’ section.

Example of DOXY and DOXY2 derived parameter definition:

```
    "CALIBRATION_COEFFICIENT": [
        {
            "OPTODE_DOXY": [
                {
                    "Case": "201_201_301",
                    "DoxyCalibRefSalinity": 0
                },
                {
                    "Case": "201_202_202",
                    "PhaseCoef0": -3.22792E-01,
                    "PhaseCoef1": 1.10079E00,
                    "PhaseCoef2": 0.00000E+00,
                    "PhaseCoef3": 0.00000E+00,
                    "CCoef00": 5.02745E+03,
                    "CCoef01": -1.69644E+02,
                    "CCoef02": 3.47372E+00,
                    "CCoef03": -3.10884E-02,
                    "CCoef10": -2.72133E+02,
                    "CCoef11": 8.19642E+00,
                    "CCoef12": -1.68036E-01,
                    "CCoef13": 1.54063E-03,
                    "CCoef20": 5.94114E+00,
                    "CCoef21": -1.57673E-01,
                    "CCoef22": 3.27461E-03,
                    "CCoef23": -3.08870E-05,
                    "CCoef30": -6.03008E-02,
                    "CCoef31": 1.39861E-03,
                    "CCoef32": -2.98859E-05,
                    "CCoef33": 2.90209E-07,
                    "CCoef40": 2.33874E-04,
                    "CCoef41": -4.68676E-06,
                    "CCoef42": 1.05069E-07,
                    "CCoef43": -1.04908E-09
                }
            ]
        }
    ],

    "parametersList": [
        {
            "ego_variable_name": "BPHASE_DOXY",
            "glider_variable_name": "sci_oxy3835_wphase_bphase",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "Batch No 2408\r\nCertificate no 3853_2408_40043",
            "derivation_date": "20100304143133",

            "processing_id": ""
        },
        {
            "ego_variable_name": "RPHASE_DOXY",
            "glider_variable_name": "sci_oxy3835_wphase_rphase",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "",
            "derivation_coefficient": "",
            "derivation_comment": "Batch No 2408\r\nCertificate no 3853_2408_40043",
            "derivation_date": "20100304143133",

            "processing_id": ""
        },
        {
            "ego_variable_name": "MOLAR_DOXY",
            "glider_variable_name": "sci_oxy3835_wphase_oxygen",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "molar_doxy=C0 + C1*dphase + C2*dphase^2 + C3*dphase^3 +
C4*dphase^4, with  Ci=Ci0 + Ci1*T + Ci2*T^2 + Ci3*T^3 (T: temperature)",
            "derivation_coefficient": "",
            "derivation_comment": "Batch No 2408\r\nCertificate no 3853_2408_40043",
            "derivation_date": "20100304143133",
            "processing_id": ""
        },
        {
            "ego_variable_name": "DOXY",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Not measured by the glider. Calculated by Coriolis",
            "derivation_coefficient": "Not measured by the glider. Calculated by Coriolis",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": "201_201_301"
        },
        {
            "ego_variable_name": "DOXY2",
            "glider_variable_name": "",
            "comment": "",
            "cell_methods": "",
            "reference_scale": "",

            "derivation_equation": "Not measured by the glider. Calculated by Coriolis",
            "derivation_coefficient": "Not measured by the glider. Calculated by Coriolis",
            "derivation_comment": "",
            "derivation_date": "",

            "processing_id": "201_202_202"
        }
    ]
```

In this example, we ask the decoder to compute DOXY and DOXY2 derived parameters.

For DOXY it should use case_201_201_301 method that needs MOLAR_DOXY parameter (which is previously defined) and associated calibration coefficients.

For DOXY2 it should use case_201_202_202 method that needs BPHASE_DOXY parameter (which is previously defined) and associated calibration coefficients.

Such configuration can then be used to compare dissolved oxygen computed by the glider (DOXY from MOLAR_DOXY) with dissolved oxygen computed by the decoder from (DOXY2 from BPHASE_DOXY).