argo data management

Euro-Argo data management
http://dx.doi.org/10.17882/45589

Coriolis Argo floats data processing chain

Version 1.7

May 22nd, 2023

ARGO
part of the integrated global observation strategy

Coriolis Argo floats data processing chain
http://dx.doi.org/10.17882/45589

Authors: Jean-Philippe Rannou

Table of contents

History

Version	Date	Comment
1.0	16/09/2016	JPR: initial version of the document.
1.1	18/10/2016	JPR: updated to be compliant with the ‘008a’ version of the software.
1.2	21/02/2017	JPR: updated to be compliant with the ‘009a’ version of the software.
1.3	11/02/2020	JPR: updated to be compliant with the ‘033a’ version of the software.
1.4	19/06/2020 18/01/2021	JPR: updated to be compliant with the ‘035c’ version of the software. JPR: Matlab version needed (at least R2016b – “split” function used).
1.5	07/09/2021	JPR: updated to be compliant with the ‘044a’ version of the software (possible generation of the 3.2 format version of the trajectory file and possible integration of Argos locations error ellipses in the trajectory).
1.6	20/10/2021	JPR: updated to be compliant with the ‘044p’ version of the software (only one directory for Argos locations error ellipses collected by web service).
1.7	22/05/2023	JPR: updated to be compliant with the ‘055l’ version of the software (distinct flags and directories to generate TRAJ 3.1 and TRAJ 3.2 files).

Reference documents

Reference N°	Title	Link
RD1	Argo user’s manual.	https://doi.org/10.13155/29825
RD2	Argo Quality Control Manual for CTD and Trajectory Data	https://doi.org/10.13155/33951
RD3	Argo Quality Control Manual for Biogeochemical Data	https://doi.org/10.13155/40879
RD4	Argo auxiliary files format for the Coriolis DAC	https://doi.org/10.13155/51995

1 Introduction

This user's manual describes how to install, configure and use the Coriolis Matlab decoder of Argo floats data.

The main functions of this decoder are:

- To decode the Argo float data,
- To format decoded data into the four Argo NetCDF CF files.

The decoder can also apply Real Time Quality Control (RTQC) tests on formatted Argo data.

Two main Matlab programs are developed around a common core decoder:

- A decoder for a float Principal Investigator (PI). It is used to decode, in delayed mode, a given amount of already received Argo float data.

  There are two PI decoders for each float type:

  - ***decode_provor_2_csv*** and ***decode_provor_2_nc*** for NKE floats (ARVOR and PROVOR) decoding,
  - ***decode_apex_2_csv*** and ***decode_apex_2_nc*** for TWR floats (APEX) decoding,
  - ***decode_nova_2_csv*** and ***decode_nova_2_nc*** for MetOcean floats (NOVA and DOVA) decoding,
  - ***decode_nemo_2_csv*** and ***decode_nemo_2_nc*** for OPTIMARE floats (NEMO) decoding.
- A decoder for a Data Assembly Centre (DAC). It is used to decode, in real time, the incoming Argo float data flux. There is one DAC decoder; it is called ***decode_argo_2_nc_rt***.

This decoder user's manual (V1.7) describes the **Coriolis Argo decoder version 055l** (with **profile RTQC version 6.0** and **trajectory RTQC version 3.5**) which generates Argo files compliant with format version 3.1 or 3.2 for trajectory file (specified in [RD1]).

The current version of the decoder is stored in the global variable `g_decArgo_decoderVersion` set in the *init_default_values.m* file.

The current version of the profile RTQC is stored in the global variable `g_decArgo_addRtqcToProfileVersion` set in the *add_rtqc_to_profile_file.m* file.

The current version of the trajectory RTQC is stored in the global variable `g_decArgo_addRtqcToTrajVersion` set in the *add_rtqc_to_trajectory_file.m* file.

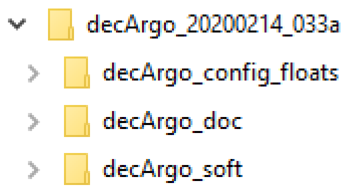
2 Floats managed by the decoder

The float type and versions managed by the current version of the decoder are listed in the *_CoriolisArgoFloatVersions_YYYYMMDD.xlsx* file (where *YYYYMMDD* is the update date of the Excel file) (see 8.2).

3 Description of the decoder package

The decoder is provided as a zipped archive named *decArgo_YYYYMMDD_xxxy.7z*, where *YYYYMMDD* is the date of the package creation and *xxxy* the decoder version.

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



Three main directories are provided:

- The *decArgo_soft* directory contains the decoder software,
- The *decArgo_doc* directory contains the documentation associated to the decoder,
- The *decArgo_config_floats* directory contains float configuration file samples.

(see Annex A for a detailed description of their contents).

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 (see Annex E) operating system.

The Matlab software must be installed on the host computer.

The Matlab version should be ≥ R2016b since the "split" function is used.

Note also that:

- The Matlab Statistics Toolbox is needed (for the Matlab ***fitlm*** function), if you want to compute NITRATE values from the transmitted spectrum (see Annex F for details),
- The GEBCO worldwide bathymetric atlas (https://www.bodc.ac.uk/data/open_download/gebco/gebco_2020/zip/) is needed by some RTQC tests,
- The (free) following components are also needed if you want to use the ***nc_trace_disp*** tool to plot the float displacements:
    - The M_Map Matlab package (https://www.eoas.ubc.ca/~rich/map.html),
    - The ETOPO2 worldwide bathymetric atlas (https://www.ngdc.noaa.gov/mgg/global/relief/ETOPO2/ETOPO2v2-2006/ETOPO2v2g/raw_binary/ETOPO2v2g_i2_MSB.zip file),
    - The SRTM30+ worldwide bathymetric atlas (http://topex.ucsd.edu/WWW_html/srtm30_plus.html) if you want to switch to a more detailed bathymetry.

4.1.2 Installation of the decoder

To install the decoder, unzip the decoder package archive and add the *decArgo_YYYYMMDD_xxxy/decArgo_soft/soft* directory, with its sub-directories, to 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.

4.2.1 PI decoder configuration

The configuration file of the PI decoder is *decArgo_YYYYMMDD_xxxy/decArgo_soft/soft/_argo_decoder_conf.txt*.

The configuration variables are listed and described in the following table.

Configuration variable name	Configuration variable description
`FLOAT_LIST_FILE_NAME`	Default list of floats to process.
`EXPECTED_CYCLE_LIST`	For Argos floats only. Used to choose the cycles to decode. However, as TRAJ data may use information from previous cycles, it is recommend to always set (`EXPECTED_CYCLE_LIST = 9999` or `EXPECTED_CYCLE_LIST = [~])` i.e. to always process existing cycles from the beginning
`FLOAT_TRANSMISSION_TYPE`	Set to 1 for Argos floats, 2 for floats with Iridium RUDICS transmission (ex: Remocean French floats), 3 for Iridium SBD common floats and 4 for BIO floats with Iridium SBD transmission (ex: INCOIS FLBB floats).
`FLOAT_INFORMATION_FILE_NAME`	Set to the file which stores the basic configuration information of the managed floats (*decArgo_YYYYMMDD_xxxy/decArgo_config_floats/argoFloatInfo/argo_floats_information_co.txt*).
`HEX_ARGOS_FILE_FORMAT`	For Argos floats only. Set to 1.
`DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`	For Argos floats only. Set to the top directory of the Argos HEX files.
`DIR_INPUT_HEX_ARGOS_FILE_FORMAT_2`	For Argos floats only. Unused since `HEX_ARGOS_FILE_FORMAT = 1`.
`HEX_ARGOS_DATA_DIRECTORY_STRUCTURE`	For Argos floats only. Set to 3.
`DIR_INPUT_RSYNC_DATA`	For Iridium floats only. Set to the top directory of the Iridium data repository.
`DIR_INPUT_RSYNC_LOG`	For Iridium floats only. Unused.
`DIR_INPUT_JSON_TECH_LABEL_FILE`	Set to the directory which stores TECH label files (*decArgo_YYYYMMDD_xxxy/decArgo_soft/config/techParamNames/)*.
`DIR_INPUT_JSON_CONF_LABEL_FILE`	Set to the directory which stores CONF label files (*decArgo_YYYYMMDD_xxxy/decArgo_soft/config/configParamNames/)*.
`DIR_INPUT_JSON_FLOAT_META_DATA_FILE`	Set to the directory which stores the detailed configuration information of the managed floats (*decArgo_YYYYMMDD_xxxy/decArgo_config_floats/json_float_meta_argos/* or *decArgo_YYYYMMDD_xxxy/decArgo_config_floats/json_float_meta_ir_sbd/* or … depending of `FLOAT_TRANSMISSION_TYPE` information).
`DIR_INPUT_DM_BUFFER_LIST`	Unused.
`IRIDIUM_DATA_DIRECTORY`	For Iridium floats only. Set to the top directory of the Iridium data storage.
`DIR_OUTPUT_LOG_FILE`	Set to the directory used to store output log files.
`DIR_OUTPUT_CSV_FILE`	Set to the directory used to store output CSV files.
`DIR_OUTPUT_XML_FILE`	Unused.
`DIR_OUTPUT_NETCDF_FILE`	Set to the top directory used to store output NetCDF files.
`DIR_OUTPUT_NETCDF_TRAJ_3_1_FILE`	Set to the top directory used to store output V3.1 NetCDF trajectory file. Note that if it is identical to DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE, only the 3.2 version of the trajectory file is generated.
`DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE`	Set to the top directory used to store output V3.2 NetCDF trajectory file.
`GENERATE_NC_MULTI_PROF`	Flag to choose to generate NetCDF Argo multi-profile file (1 if you want to generate it, 0 otherwise).
`GENERATE_NC_MONO_PROF`	Flag to choose to generate NetCDF Argo mono-profile files (1 if you want to generate it, 0 otherwise).
`GENERATE_NC_TECH`	Flag to choose to generate NetCDF Argo technical file (1 if you want to generate it, 0 otherwise).
`GENERATE_NC_META`	Flag to choose to generate NetCDF Argo meta-data file (1 if you want to generate it, 0 otherwise).
`GENERATE_NC_TRAJ_3_1`	Flag to choose to generate the 3.1 version of NetCDF Argo trajectory file (1 if you want to generate it, 0 otherwise).
`GENERATE_NC_TRAJ_3_2`	Flag to choose to generate the 3.2 version of NetCDF Argo trajectory file (1 if you want to generate it, 0 otherwise).
`ADD_ARGOS_ERROR_ELLIPSES`	Flag to add error ellipses of Argos locations in the trajectory files (1 if you want to add them, 0 otherwise).
`DIR_INPUT_ARGOS_ERROR_ELLIPSES_MAIL`	Top directory of input Argos locations error ellipses collected by mail (should not be set if not available).

DIR_INPUT_ARGOS_ERROR_ELLIPSES_WS	Top directory of input Argos locations error ellipses collected by web service (should not be set if not available).
APPLY_RTQC	Flag to apply RTQC tests to decoded NetCDF files (1 if you want to apply RTQC tests, 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.
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.
ADD_THREE_MINUTES	For NKE Argos floats only. The Argos times provided by old versions of NKE Argos floats have a 6 minutes resolution. If you set this flag to 1, 3 minutes will be added to some of these times (at Coriolis we usually set this item to 0, i.e. we don't apply this correction).

4.2.2 DAC decoder configuration

The configuration file of the DAC decoder is *decArgo_YYYYMMDD_xxxy/decArgo_soft/soft/_argo_decoder_conf.json*.

The configuration variables are listed and described in the following table.

Configuration variable name	Configuration variable description
FLOAT_LIST_FILE_NAME	Default list of floats to process.
EXPECTED_CYCLE_LIST	For Argos floats only. Used to choose the cycles to decode. However, as TRAJ data may use information from previous cycles, it is recommend to always set (EXPECTED_CYCLE_LIST = 9999 or EXPECTED_CYCLE_LIST = [~]) i.e. to always process existing cycles from the beginning
FLOAT_TRANSMISSION_TYPE	Set to 1 for Argos, 2 for BIO floats with Iridium RUDICS transmission (ex: Remocean French floats), 3 for Iridium SBD common floats and 4 for BIO floats with Iridium SBD transmission (ex: INCOIS FLBB floats).
DIR_INPUT_JSON_FLOAT_DECODING_PARAMETERS_FILE	Set to the directory which stores float basic configuration information files.
HEX_ARGOS_FILE_FORMAT	For Argos floats only. Set to 1.
DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1	For Argos floats only. Set to the top directory of the Argos HEX files.
DIR_INPUT_HEX_ARGOS_FILE_FORMAT_2	For Argos floats only. Unused since HEX_ARGOS_FILE_FORMAT = 1.
HEX_ARGOS_DATA_DIRECTORY_STRUCTURE	For Argos floats only. Set to 3.
DIR_INPUT_RSYNC_DATA	For Iridium floats only. Set to the top directory of the Iridium data repository.
DIR_INPUT_RSYNC_LOG	For Iridium floats only. Set to the directory which stores RSYNC log files.
DIR_INPUT_JSON_TECH_LABEL_FILE	Set to the directory which stores TECH label files (*decArgo_YYYYMMDD_xxxy/decArgo_soft/config/techParamNames/*).
DIR_INPUT_JSON_CONF_LABEL_FILE	Set to the directory which stores CONF label files (*decArgo_YYYYMMDD_xxxy/decArgo_soft/config/configParamNames/*).
DIR_INPUT_JSON_FLOAT_META_DATA_FILE	Set to the directory which stores the detailed configuration information of the managed floats (*decArgo_YYYYMMDD_xxxy/decArgo_config_floats/json_float_meta_argos/* or *decArgo_YYYYMMDD_xxxy/decArgo_config_floats/json_float_meta_ir_sbd/* or ... depending of FLOAT_TRANSMISSION_TYPE information).
DIR_INPUT_DM_BUFFER_LIST	For FLOAT_TRANSMISSION_TYPE #2 and #4 floats only. Set to the directory used to store the 'buffers definition' files (see Annex C).
IRIDIUM_DATA_DIRECTORY	For Iridium floats only. Set to the top directory of the Iridium data storage.
DIR_OUTPUT_LOG_FILE	Set to the directory used to store output log files.
DIR_OUTPUT_CSV_FILE	Set to the directory used to store output CSV files.
DIR_OUTPUT_XML_FILE	Set to the directory used to store output XML files.
DIR_OUTPUT_NETCDF_FILE	Set to the top directory used to store output NetCDF files.
DIR_OUTPUT_NETCDF_TRAJ_3_1_FILE	Set to the top directory used to store output V3.1 NetCDF trajectory file. Note that if it is identical to DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE, only the 3.2 version of the trajectory file is generated.
DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE	Set to the top directory used to store output V3.2 NetCDF trajectory file.

PROCESS_REMAINING_BUFFERS	Flag to choose if the not complete buffers should be processed (see Annex B).
GENERATE_NC_MULTI_PROF	Flag to choose to generate NetCDF Argo multi-profile file (1 if you want to generate it, 0 otherwise).
GENERATE_NC_MONO_PROF	Flag to choose to generate NetCDF Argo mono-profile files (1 if you want to generate it, 0 otherwise).
GENERATE_NC_TECH	Flag to choose to generate NetCDF Argo technical file (1 if you want to generate it, 0 otherwise).
GENERATE_NC_META	Flag to choose to generate NetCDF Argo meta-data file (1 if you want to generate it, 0 otherwise).
GENERATE_NC_TRAJ_3_1	Flag to choose to generate the 3.1 version of NetCDF Argo trajectory file (1 if you want to generate it, 0 otherwise).
GENERATE_NC_TRAJ_3_2	Flag to choose to generate the 3.2 version of NetCDF Argo trajectory file (1 if you want to generate it, 0 otherwise).
ADD_ARGOS_ERROR_ELLIPSES	Flag to add error ellipses of Argos locations in the trajectory files (1 if you want to add them, 0 otherwise).
DIR_INPUT_ARGOS_ERROR_ELLIPSES_MAIL	Top directory of input Argos locations error ellipses collected by mail (should not be set if not available).
DIR_INPUT_ARGOS_ERROR_ELLIPSES_WS	Top directory of input Argos locations error ellipses collected by web service (should not be set if not available).
APPLY_RTQC	Flag to apply RTQC tests to decoded NetCDF files (1 if you want to apply RTQC tests, 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.
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.
ADD_THREE_MINUTES	For NKE Argos floats only. The Argos times provided by old versions of NKE Argos floats have a 6 minutes resolution, if you set this information to 1, 3 minutes will be added to some of these times (at Coriolis we usually set this item to 0, i.e. we don't apply this correction).

5 Float configuration

Information on each managed float is provided in two files:

- One file contains the basic decoder configuration information,
- One file contains the float meta-data information.

5.1 Float configuration files for PI decoder

5.1.1 Float decoder configuration information

For each managed float, 12 configuration information are needed by the decoder.

These data are stored in a unique file (`FLOAT_INFORMATION_FILE_NAME`) which contains one line for each float.

Note that we usually manage these data in an Excel file (*argo_floats_information_co.xls*) but the decoder uses the ASCII version of this file (*argo_floats_information_co.txt* generated from a copy of the 12 first columns of the Excel file).

The needed information is the following:

- Column #1: float WMO number,
- Column #2: decoder Id associated to the float,
- Column #3: Argos ptt (Argos transmission) or IMEI number (Iridium SBD transmission) or login name (Iridium RUDICS transmission) of the float,
- Column #4: for Argos floats only. Frame length (in bytes) of the Argos float message (it is 31 for the current Argos floats (with 28-bits Argos ids) and 32 for older Argos floats (with 20-bits Argos ids)),
- Column #5: for Argos floats only. Float cycle length (in hours),
- Column #6: for Argos floats only. Float subsurface drift sampling period (in hours),
- Column #7: for NKE BIO floats only. Firmware version of the PROVOR_II type float,
- Column #8: float launch date,
- Column #9: float launch longitude,
- Column #10: float launch latitude,
- Column #11: date of the first descent of the float (day #0 of the NKE float internal calendar),
- Column #12: for Iridium floats only. End decoding date (used to manage received data when the same IMEI number has been used by more than one float).

5.1.2 Float meta-data file

The float meta-data are stored in a file called *wmo_meta.json*; the files of the managed floats are gathered in the `DIR_INPUT_JSON_FLOAT_META_DATA_FILE` directory.

Most of the information has a name compliant with the Argo Metadata V3.1 format (see chapter 2.4 of [RD1] for their detailed description). These data will be copied by the decoded in the generated META NetCDF file.

Some specific items are processed by the decoder:

- CONFIG_PARAMETER_NAME and CONFIG_PARAMETER_VALUE are used to fill the LAUNCH_CONFIG_PARAMETER_NAME and LAUNCH_CONFIG_PARAMETER_VALUE arrays of the META NetCDF file,
- CONFIG_REPETITION_RATE is used to fill CONFIG_PARAMETER_NAME and CONFIG_PARAMETER_VALUE arrays of the META NetCDF file for multi mission floats,
- CALIBRATION_COEFFICIENT is used to compute derived parameter,
- RT_OFFSET is used to apply RT adjustments on decoded data,
- CALIB_RT_PARAMETER, CALIB_RT_EQUATION, CALIB_RT_COEFFICIENT, CALIB_RT_COMMENT and CALIB_RT_DATE are used to fill PARAMETER and SCIENTIFIC_CALIB_* arrays of the PROFILE NetCDF files,
- SENSOR_MOUNTED_ON_FLOAT is used to store the sensor list of BIO floats.

5.1.2.1 Float meta-data file generation

It's up to each user to generate its own meta-data file.

At Coriolis we use specific tools (***generate_json_float_meta_****) to generate the float meta-data files from data exported from the Coriolis data base.

5.2 Float configuration files for DAC decoder

5.2.1 Float decoder configuration information

The float decoder configuration information is stored in a file called *wmo_emitterId_info.json* (*emitterId* could be Argos ptt (Argos transmission) or IMEI number (Iridium SBD transmission) or login name (Iridium RUDICS transmission)); the files of the managed floats are gathered in the `DIR_INPUT_JSON_FLOAT_DECODING_PARAMETERS_FILE` directory.

For each managed float, 15 configuration information are provided in this file:

- WMO: float WMO number,
- PTT: Argos ptt (Argos transmission) or IMEI number (Iridium SBD transmission) or login name (Iridium RUDICS transmission) of the float,
- FLOAT_TYPE: float type (presently PROVOR, APEX or NOVA),
- DECODER_VERSION: Coriolis version of the float,
- DECODER_ID: decoder Id associated to the float,
- FRAME_LENGTH: for Argos floats only. Frame length (in bytes) of the Argos float message (it is 31 for the current Argos floats (with 28-bits Argos ids) and 32 for older Argos floats (with 20-bits Argos ids)),
- CYCLE_LENGTH: for Argos floats only (unused otherwise). Float cycle length (in hours),
- DRIFT_SAMPLING_PERIOD: for Argos floats only (unused otherwise). Float subsurface drift sampling period (in hours),
- DELAI: for NKE BIO floats only. Firmware version of the PROVOR_II type float,
- LAUNCH_DATE: float launch date,
- LAUNCH_LON: float launch longitude,
- LAUNCH_LAT: float launch latitude,
- END_DECODING_DATE: for Iridium floats only. End decoding date (used to manage

received data when the same IMEI number has been used by more than one float),

- REFERENCE_DAY: date of the first descent of the float (day #0 of the NKE float internal calendar),
- DM_FLAG: float decoding Delayed Mode flag (see Annex C).

5.2.1.1 Float configuration file generation

The tool ***generate_json_float_info*** can be used to generate the float configuration files (from an ASCII file generated from a copy of the 14 first columns of the *argo_floats_information_co.xls* Excel file).

5.2.2 Float meta-data file

The already described (see 5.1.2) float meta-data files are common to PI and DAC decoders.

6 Using the PI decoder

6.1 Pre-processing of float transmitted data

To be decoded, the input raw data should first be pre-processed. This step depends on float transmission type.

6.1.1 For Argos floats

Two types of input data are collected for Argos floats:

- Argos Hex data transmitted by the float together with associated Argos locations estimated by CLS,
- Optionally, Argos locations error ellipses computed by CLS.

6.1.1.1 Argos Hex data

The Argos Hex data, coming from CLS by e-mails or CD-ROM, need to be prepared to be used by the decoder. This process is done in the following steps.

6.1.1.1.1 Step #0: copy all received Argos data in a unique directory

All received data (e-mail files, CD-ROM contents, …) should be first copied in a unique directory.

All these files should be at the same level of the directory (no sub-directories are allowed). Don't worry about duplicated data (step #2 will delete the duplicates).

6.1.1.1.2 Step #1: split the data

In this step we split the data by Argos Id number and by satellite pass.

The input directory should be the one created on step #0. The output directory will contain one directory for each Argos Id and within each of these sub-directories, one file per satellite pass for the concerned Argos Id.

The tool ***split_argos_cycle*** is used for step #1.

6.1.1.1.3 Step #2: delete duplicated data

In this step we check the satellite pass files generated from step #1 and delete duplicated data.

The tool ***delete_double_argos_split*** is used for step #2.

6.1.1.1.4 Step #3: create Argos cycle files

In this step we create Argos cycle file (containing all the data transmitted by the float after each cycle) from satellite pass files obtained in step #2.

Each Argos cycle file contains the data of the satellite pass files concatenated and chronologically sorted. A new Argos cycle file is created each time we find a 10 (`g_decArgo_minNonTransDurForNewCycle` global variable) hours delay without any data transmission.
The tool ***create_argos_cycle_files*** is used for step #3.

6.1.1.1.5 Step #4: correct Argos cycle files

In this step we check that the number of lines of each satellite pass set by CLS in the satellite pass header is consistent with the real number of lines of the satellite pass; erroneous counts are corrected in the output file.

The tool ***co_cls_correct_argos_raw_file*** is used for step #4.

6.1.1.1.6 Step #5: name Argos cycle files

In this step we compute the cycle number associated to each Argos cycle file and create the final name of the file.

The name of the Argos cycle file should be:

ArgosId_YYYY-MM-DD-hh-mm-ss_WMO_CyNum.txt, where

- *ArgosId*: is the float PTT number (on 6 digits),
- *YYYY-MM-DD-hh-mm-ss*: is the date of the earlier float message of the file,
- *WMO*: is the float WMO number,
- *CyNum*: is the cycle number.

Note also that:

- *WMO* can be equal to 'WWWWWWW' if the ArgosId to WMO link is unknown at the time of reception of the data,
- *CyNum* can be equal to:
    - 'EEE': empty file (not at least one float message),
    - 'WWW': ArgosId to WMO link is unknown at the time of reception of the data,
    - 'MMM': meta-data unavailable to compute cycle number,
    - 'TTT': test data (dated before float launch date),
    - 'GGG': ghost messages,
    - 'UUU': cycle number value (manually) disabled by the user.

Only 'identified' cycle files (i.e. with valid *WMO* and *CyNum* numbers) are processed by the decoder.

The tool ***move_and_rename_prv_argos_files*** or ***move_and_rename_apx_argos_files*** is used for step #5. It uses information from the json meta-data files.

6.1.1.1.7 Step #6: clean ghost data at the end of Argos cycle files

In this step we try to detect data from a ghost transmission at the end of the cycle file and to move it to another appropriate cycle file.

This step is applied to Apex data only (since a reliable Last Message Time (LMT) is crucial for the determination of the Apex cycle timings).

The tool ***clean_ghost_in_apx_argos_cycle_files*** is used for step #5.

6.1.1.1.8 Final step: check the processed output files

Use the tool ***check_argos_cycle_files*** to check the work done and the file contents (particularly in columns 'D' to 'G' of the CSV file generated by this tool).

The output files can be used for decoding (`DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1` directory).

Note also that the tool *process_argos_data_prv* (for NKE and Metocean floats) or *process_argos_data_apx* (for Apex floats) can be used to process these steps.

6.1.1.2 Argos locations error ellipses

The Argos locations error ellipses computed by CLS were originally received by e-mails, they are now collected through a web service.

The e-mail data were received as CSV data with the following header:

```
Program;PTT;Satellite;Location date;Location class;Latitude;Longitude;Latitude solution
2;Longitude solution 2;Number of messages;Nbr mes > - 120 dB;Best level;Pass
duration;NOPC;Location index;Frequency;Altitude;Error radius;Semi-major axis;Semi-minor
axis;Ellipse orientation;GDOP;Message date;Compression
```

these data have been processed with the ***split_argos_ellipses_mail_files*** tool to generate CSV files with the following header:

```
Program;PTT;Satellite;Location date;Location class;Latitude;Longitude;Frequency;Altitude;Error
radius;Semi-major axis;Semi-minor axis;Ellipse orientation
```

and the resulting output directory is set to the `DIR_INPUT_ARGOS_ERROR_ELLIPSES_MAIL` configuration parameter.

The CSV files collected through a web service have the following header:

```
programNumber;platformId;platformType;platformModel;platformName;satellite;bestMsgDate;duratio
n;nbMessage;message120;bestLevel;frequency;locationDate;latitude;longitude;altitude;locationCl
ass;gpsSpeed;gpsHeading;index;nopc;errorRadius;semiMajor;semiMinor;orientation;hdop
```

and are set to the `DIR_INPUT_ARGOS_ERROR_ELLIPSES_WS` configuration parameter.

6.1.2 For Iridium SBD floats

6.1.2.1 Rename the mail files

The incoming Iridium mail files should be renamed to be used by the decoder. This can be done with the tool ***move_and_rename_iridium_sbd_mail_files***.

Each processed file will be renamed:

co_YYYYMMDDThhmmss_IMEI_MOMSN_MTMSN_PID.txt, where

- *YYYYMMDDThhmmss*: is the date of the Iridium session,
- *IMEI*: is the float IMEI number,
- *MOMSN*, *MTMSN*: are the MOMSN and MTMSN numbers of the transmission,
- *PID*: is the PID of the process that collected the mail (unused by the decoder).

The newly named files can then be moved to the repository (`DIR_INPUT_RSYNC_DATA` directory).

6.1.2.2 Duplicate the mail files

To be decoded, the Iridium mail files should be duplicated from the repository (`DIR_INPUT_RSYNC_DATA` directory) to the decoder specific directory (`IRIDIUM_DATA_DIRECTORY` directory). This can be done with the tool ***copy_iridium_mail_files***.

6.1.3 For Iridium RUDICS floats

6.1.3.1 Duplicate the Iridium files

To be decoded, the Iridium files should be duplicated from the repository (`DIR_INPUT_RSYNC_DATA` directory) to the decoder specific directory (`IRIDIUM_DATA_DIRECTORY` directory). This can be done with the tools ***copy_remocean_sbd_files*** or ***copy_cts5_files*** or ***copy_apx_iridium_rudics_files*** or ***copy_apx_apf11_iridium_rudics_files*** or ***copy_nemo_files***.

6.2 Decoding of float transmitted data

To use the decoder you can type, in the Matlab command window:

- ***decode_provor_2_csv*** or ***decode_provor_2_nc*** to decode NKE floats,
- ***decode_apex_2_csv*** or ***decode_apex_2_nc*** to decode TWR floats,
- ***decode_nova_2_csv*** or ***decode_nova_2_nc*** to decode MetOcean floats,
- ***decode_nemo_2_csv*** or ***decode_nemo_2_nc*** to decode OPTIMARE floats.

Doing so, all the float of the `FLOAT_LIST_FILE_NAME` file will be decoded.

You can also choose to decode only few floats by providing their WMO numbers as input parameters: ***decode_provor_2_csv(2902077)*** or ***decode_provor_2_nc(2902089, 2902118, 2902086)*** for example.

The ***decode_*_2_csv*** decoders perform a 'raw' decoding of the input data (decoding of the transmitted float messages). The decoded data are stored in a CSV file (one per decoder session).

The ***decode_*_2_nc*** decoders perform the 'full' decoding of the input data (decoding of the transmitted float messages, processing of derived parameters, application of Argo rules). The decoded data are stored in NetCDF files compliant with the Argo V3.1 (or V3.2 for trajectory file) format.

6.3 Decoder input and output files

The files associated to the decoder are:

- Input files:
  - Decoder configuration file (*_argo_decoder_conf.txt*),
  - Float decoder configuration file (`FLOAT_INFORMATION_FILE_NAME`),
  - Float meta-data file (*wmo_meta.json*),
  - Float transmitted data:
    - Argos data: 'identified' cycle files (stored in `DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`/*Argos_Id*),
    - Iridium data: Iridium files (stored in `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO* or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO*).
- Output files:
  - Log file of the session (stored in `DIR_OUTPUT_LOG_FILE`),
  - For ***decode_*_2_csv*** programs: CSV file of the decoded data (stored in `DIR_OUTPUT_CSV_FILE`),
  - For ***decode_*_2_nc*** programs: Argo NetCDF file of the decoded data (stored in `DIR_OUTPUT_NETCDF_FILE`, `DIR_OUTPUT_NETCDF_TRAJ_3_1_FILE`, `DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE`).

7 Using the DAC decoder

The DAC decoder is designed to be deployed in a Real Time data flux. Its deployment depends on each DAC infrastructure; we will describe what is done in the Coriolis one.

7.1 Decoder input parameters

Remember that the PI decoder is used to decode already received data of a given float. Thus the only input parameter allowed is the WMO numbers of the floats to decode.

The needs of a DAC for real time processing can be more complex, it can be:

- To decode new incoming data (thus HEX Argos data or a list of Iridium files),
- To reprocess all the data received for a given float (thus a float provided by its WMO number).

Moreover, the frequency of the NetCDF file update can depend on Argo file types:

- High frequency for profile files (i.e. each time data is received from the float),
- Low frequency for trajectory, meta-data and technical files (i.e. only when the float has ended its surface transmission).

Consequently, the DAC decoder needs specific input parameters.

These parameters should be provided in pairs: *'parameterName'*, *'parameterValue'*.

The allowed parameters depend on float transmission type.

7.1.1 For Argos floats

- **'processmode'**: this parameter name is used to choose the data you want to decode and the NetCDF files you want to generate. This *'parameterName'* can be associated to one of the 3 following *'parameterValue'*:
    - **'all'**: in this mode the new HEX Argos data file (provided with the **'argosfile'** parameter) and all the already received data (stored in the `DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`/*ArgosId* directory) are processed and all the NetCDF files are generated (according to the `GENERATE_NC_*` configuration flags). After that, the new HEX Argos data file is archived in the `DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`/*ArgosId* directory. This mode is used at Coriolis, once per cycle at the end of the transmission to generate the TRAJ file and to be sure that all the received data has been exploited for the profile.
    - **'profile'**: in this mode only the new HEX Argos data file (provided with the **'argosfile'** parameter) is processed. Moreover, the `GENERATE_NC_TRAJ`, `GENERATE_NC_MULTI_PROF`, `GENERATE_NC_TECH` and `GENERATE_NC_TRAJ` flags are (temporarily) set to 0 and the `GENERATE_NC_MONO_PROF` flag is set to 2. Consequently, only the mono-profile NetCDF file associated to the provided new HEX Argos data file is created or updated (if needed). This mode is used at Coriolis each time (once per hour) HEX data are received from CLS to create a complete version of the corresponding mono-profile file.
    - **'redecode'**: in this mode all the already received data of given float (provided with the **'floatwmo'** parameter) are processed and all the NetCDF files are generated (according to the `GENERATE_NC_*` configuration flags). This mode is used at Coriolis to re-decode the data of dead floats each time a new version of the decoder is

available (with updates or corrections that concern the Argos decoding chain or the NetCDF files generation).

- **'argosfile'**: this parameter name is used to provide the name of the file containing the new incoming HEX Argos data. The associated *'parameterValue'* is the absolute file path name of the concerned file.
- **'floatwmo'**: this parameter name is used to provide a float WMO number. The associated *'parameterValue'* is the float WMO number.

The 'processmode' parameter is mandatory.

When 'processmode' is associated to 'all' or 'profile', the parameter 'argosfile' is mandatory.

When 'processmode' is associated to 'redecode', the parameter 'floatwmo' is mandatory.

7.1.2 For Iridium floats

- **'rsynclog'**: this parameter name is used to select a log generated by the rsync tool used to identify the new incoming Iridium files. This *'parameterName'* can be associated to one of the 2 following *'parameterValue'*:
  - **'all'**: in this mode, all the log files stored in the `DIR_INPUT_RSYNC_LOG` directory are considered.
  - The name (without any path) of one rsync log file stored in the `DIR_INPUT_RSYNC_LOG` directory.
- **'floatwmo'**: this parameter name is used to provide a float WMO number. The associated *'parameterValue'* is the float WMO number.

The 'rsynclog' and 'floatwmo' parameters are mandatory.

Note that when an Iridium float is decoded:

- The list of rsync log files checked is stored in the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO/history_of_processed_data/processed_rsync_log_wmo.txt* file (for `FLOAT_TRANSMISSION_TYPE` #3 or #4 floats) or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO/history_of_processed_data/processed_rsync_log_wmo.txt* file (for `FLOAT_TRANSMISSION_TYPE` #2 floats). These rsync log files will not be considered during the next sessions of the decoder,
- The list of rsync log files used (i.e. that report at least one float mail or SBD file) is stored in the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO/history_of_processed_data/used_rsync_log_wmo.txt* file (for `FLOAT_TRANSMISSION_TYPE` #3 or #4 floats) or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO/history_of_processed_data/used_rsync_log_wmo.txt* file (for `FLOAT_TRANSMISSION_TYPE` #2 floats). These rsync log files are stored for debugging purposes only,
- The list of decoded buffers is stored in the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO/history_of_processed_data/wmo_buffers.txt* file (for `FLOAT_TRANSMISSION_TYPE` #3 or #4 floats) or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO/history_of_processed_data/wmo_buffers.txt* file (for `FLOAT_TRANSMISSION_TYPE` #2 floats). These buffers will be used during the next sessions of the decoder,

Consequently:

- When you need to reprocess a float, you should clean its

`IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO* or
`IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO* directory.

7.1.3 Additional parameters of the decoder

Each configuration value of the *_argo_decoder_conf.json* configuration file can be modified through an input parameter. The input *'parameterName'* should be the name of the configuration variable and the input *'parameterValue'* should be the new associated configuration value.

Two optional parameters can also be used by the decoder:

- **'configfile'**: this parameter name is used to provide the name of the configuration file to use. The associated *'parameterValue'* is the absolute file path name of the .json configuration file you want to use. You can provide more than one configuration file in the parameters of the decoder; in this case it chooses the first that matches the float WMO number (provided by the **'floatwmo'** parameter).
- **'xmlreport'**: this parameter name is used to provide the name of the XML report file. The associated *'parameterValue'* is the file name of the concerned file that will be stored in the directory `DIR_OUTPUT_XML_FILE`. To be consistent with the Coriolis infrastructure and naming conventions, this name is expected to have the pattern *co041404_yyyymmddTHHMMSSZ[_PID].xml*.

  If you don't use this parameter, the log file (*decode_argo_2_nc_rt_yyyymmddTHHMMSSZ.log*) and the XML report file (*co041404_yyyymmddTHHMMSSZ.xml*) will have the same time stamp which corresponds to the start of the decoder (not precisely known by the script that starts the decoding session). This parameter has been added to be sure to get the right XML report for the post-processing operations.

7.2 Decoder input and output files

The files associated to the decoder are:

- Input files:
  - Decoder configuration file (*_argo_decoder_conf.json*),
  - Float decoder configuration file (*wmo_emitterId_info.json*),
  - Float meta-data file (*wmo_meta.json*),
  - Float transmitted data:
    - Argos data:
      - If **'processmode'** = **'all'**:
        - 'identified' cycle files (stored in `DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`/*Argos_Id*),
        - New incoming HEX Argos data (sent through the **'argosfile'** parameter).
      - If **'processmode'** = **'profile'**:
        - New incoming HEX Argos data (sent through the **'argosfile'** parameter).
      - If **'processmode'** = **'redecode'**:
        - 'identified' cycle files (stored in

DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1/*Argos_Id*).

- Iridium data: Iridium files listed in the rsync log files set (sent through the **'rsynclog'** parameter) excluding already processed files.

- Output files:
    - Log file of the session (stored in DIR_OUTPUT_LOG_FILE),
    - Argo NetCDF files of the decoded data (stored in DIR_OUTPUT_NETCDF_FILE, DIR_OUTPUT_NETCDF_TRAJ_3_1_FILE, DIR_OUTPUT_NETCDF_TRAJ_3_2_FILE). They are updated according to GENERATE_NC_* flag configuration values and **'processmode'** input parameter (for Argos floats only),
    - XML report of the session (stored in DIR_OUTPUT_XML_FILE). It summarizes what has been done and report the 'INFO', 'WARNING' and 'ERROR' information listed in the log file. This report is used at Coriolis to monitor the decoding step and to initiate the post-processing steps.

7.3 Deployment of the DAC decoder in the Coriolis infrastructure

The scripts used at Coriolis to deploy the DAC decoder can be sent on demand.

The main philosophy of the processing is the following.

7.3.1 Argos floats processing

The CLS service has been asked to send Argos data through e-mail to Coriolis once per hour (at HH:00).

Each time an Argos e-mail is received from CLS, a process

1. Catches it,
2. Duplicates it in the *archive* directory,
3. Moves it in the *spool_message* directory.

Each hour (at HH:15), a process is launched:

1. It processes the files of the *spool_message* directory:
    a. It split the HEX data of the files according to the Argos Id and create a specific file for this Argos Id Hex data,
    b. These files are stored in the *spool_cycle* directory,
    c. When the transmission stops for more than 18 hours, a new file is created,
    d. The *spool_cycle* directory finally contains files:
        i. With the HEX data of a unique Argos Id,
        ii. With gaps in the transmission less than 18 hours.
2. It processes the files of the *spool_cycle* directory:
    a. It looks for files where the last HEX data have been received for more than 24 hours (assuming that, in this case, the surface transmission ended). These files are process with the **'processmode'** = **'all'** and then processed and archived by the decoder,
    b. It processes the remaining files with the **'processmode'** = **'profile'**.

7.3.2 Iridium floats processing

Each time an e-mail is received from Iridium, a first process:

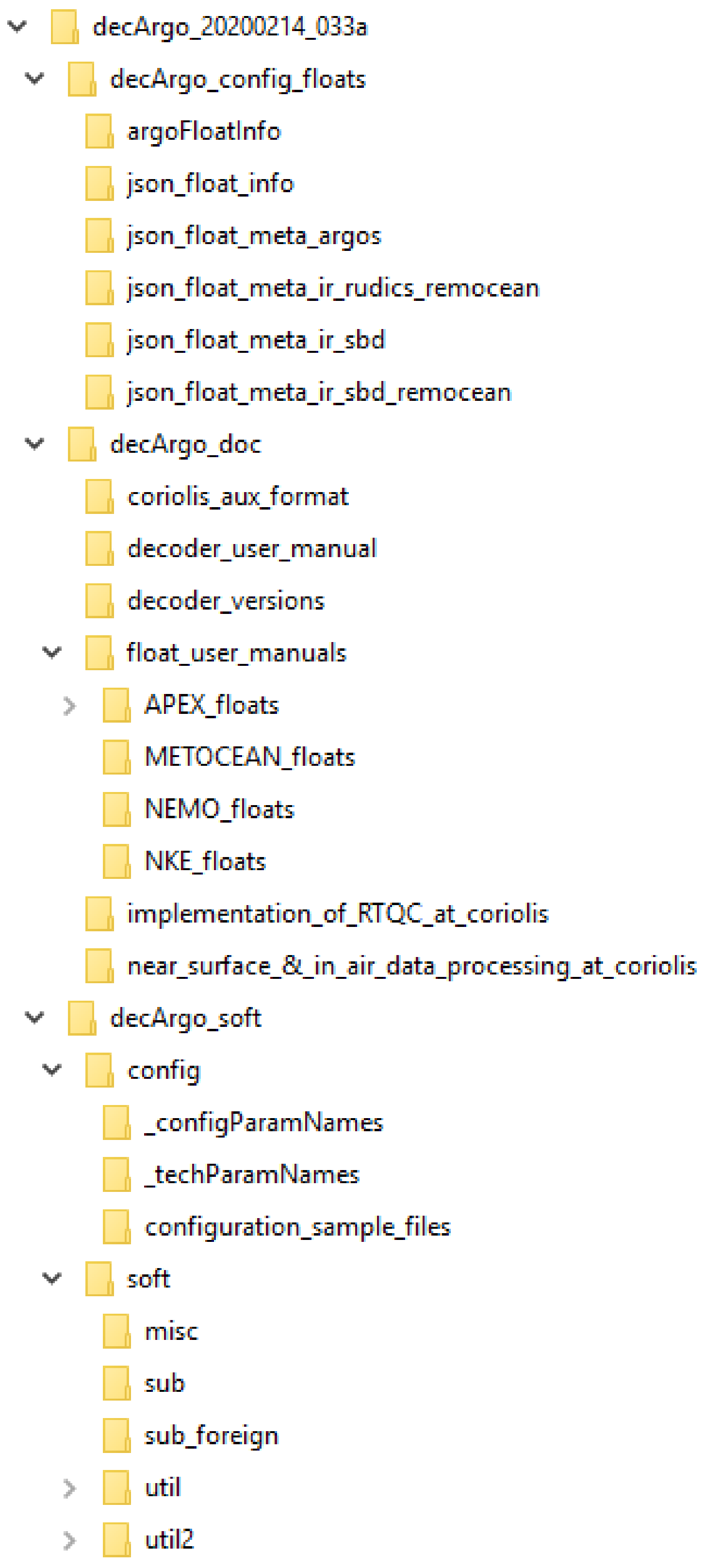
1. Catches it,
2. Duplicates it in the *archive* directory,
3. Moves it in the *spool_message* directory.

Four times per day, a process is launched:

1. It launches a ***rsync*** command on the *spool_message* directory (for Iridium SBD floats) or on the repository of the RUDICS server (for Iridium RUDICS floats),
2. It launches one instance of the decoder for each managed float with the **'rsynclog'** = **'all'** and **'floatwmo'** = concerned float WMO number.

8 ANNEX A: detailed description of the decoder package

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



```
decArgo_20200214_033a
  decArgo_config_floats
    argoFloatInfo
    json_float_info
    json_float_meta_argos
    json_float_meta_ir_rudics_remocean
    json_float_meta_ir_sbd
    json_float_meta_ir_sbd_remocean
  decArgo_doc
    coriolis_aux_format
    decoder_user_manual
    decoder_versions
    float_user_manuals
      APEX_floats
      METOCEAN_floats
      NEMO_floats
      NKE_floats
    implementation_of_RTQC_at_coriolis
    near_surface_&_in_air_data_processing_at_coriolis
  decArgo_soft
    config
      _configParamNames
      _techParamNames
      configuration_sample_files
    soft
      misc
      sub
      sub_foreign
      util
      util2
```

Three main directories are provided: *decArgo_soft*, *decArgo_doc* and *decArgo_config_floats*.

8.1 The decArgo_soft directory

The *decArgo_soft* directory contains the decoder software and configuration.

8.1.1 The decArgo_soft/soft directory

In this directory you can find:

- The decoder programs: ***decode_provor_2_csv***, ***decode_provor_2_nc***, ***decode_apex_2_csv***, ***decode_apex_2_nc***, ***decode_nova_2_csv, decode_nova_2_nc, decode_nemo_2_csv, decode_nemo_2_nc, decode_argo_2_nc_rt*** and ***decode_provor_2_nc_dm***,
- The decoder configuration files: *_argo_decoder_conf.txt* and *_argo_decoder_conf.json*,
- The software sub-directories: *sub* for the implemented functions and *sub_foreign* for the part of code coming from other people than the Coriolis development team,
- The tools directories: *util* for the user tools and *util2* for the specialized operator tools.

8.1.2 The decArgo_soft/config directory

In this directory you can find:

- The *configParamNames* and *techParamNames* sub-directories used to store configuration and technical labels information (see Annex G),
- The *configuration_sample_files* sub-directory which provides examples of configuration files.

8.2 The decArgo_doc directory

The *decArgo_doc* directory contains:

- The *decoder_user_manual* sub-directory to store this User Manual and the *_CoriolisArgoFloatVersions_YYYYMMDD.xlsx* file,
- The *decoder_versions* sub-directory to store information on the different versions of the decoder code,
- The *float_user_manuals* sub-directory to store User Manuals of the managed float versions,
- The *Coriolis_aux_format*, *implementation_of_RTQC_at_coriolis* and *near_surface_&_in_air_data_processing_at_coriolis* sub-directories to store information to explain how the decoder manages these particular topics.

8.3 The decArgo_config_floats directory

The *decArgo_config_floats* directory contains:

- The *argoFloatInfo* sub-directory to store an example of float decoder configuration information file (for the PI decoder),
- The *json_float_info* sub-directory to store examples of float decoder configuration information files (for the DAC decoder),
- The *json_float_meta_** sub-directories to store examples of float meta-data files.

Note that, in these sample files, the IMEI float numbers have been replaced by 'xxxxxxxxxxxxxxx'.

9 ANNEX B: specificities of Iridium data decoder

This Annex aims at describing the management of the Iridium files during a session of the decoder **on NKE float data**.

9.1 Management of Iridium mail files received from FLOAT_TRANSMISSION_TYPE #3 or #4 floats

To be decoded, the set of all received Iridium mail files should be split in sets of files that correspond to one received cycle. We call "buffer (of data)" the SBD attachment files of each such set of mail files.

The management of these buffers depends on the float ability to store non-transmitted data into an internal memory card (Ice floats) or not (non Ice floats).

9.1.1 For non Ice floats

Non Ice floats have no storage capabilities; the data to be transmitted should be sent during the transmission session that follows each cycle surfacing.

In that case, the buffers are virtually managed through two lists of files: *'spool'* and *'buffer'*.

Just before decoding, all the received Iridium mail files are in the '`IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO*/*archive'* directory. This has been done:

- By the operator (with the tool ***copy_iridium_mail_files***, see 6.1.2.2) before using the PI decoder,
- By the DAC decoder (after analysis of the rsync log files to identify new incoming Iridium mail files).

At the beginning of the decoding session, the Iridium mail files dated in the interval [*float_launch_date*, *end_decoding_date*] (see 5.1.1 or 5.2.1) are loaded in the '*spool'* list (the date used is the date of the Iridium session, also stored in the Iridium mail file name, see 6.1.2.1).

During the decoding session, each file of the '*spool'* list is processed in chronological order. The processing steps are the following:

1. Move the mail file to the '*buffer'* list.
2. Process the mail file, that is:
   a. Store the Iridium session information (provided in the mail),
   b. Extract the mail file attachment (the SBD file), if any.
3. Process a 'light' decoding of all the SBD files of the '*buffer'* list.

   This 'light' decoding consists of:
   a. Retrieving from technical information the expected number of messages of each type to be received,
   b. Counting the number of received messages of each type.
4. Check if all expected messages have been received (i.e. if the transmission of the cycle data is

completed).

- If not: apply steps #1 to #4 to the next mail file of the '*spool*' list.
- If yes, the buffer is complete; we can then continue to step #5 below.

5. Process a 'full' decoding of all the SBD files of the '*buffer*' list.
6. Remove the decoded mail files from the '*buffer*' list and delete the decoded SBD files.

This theoretical algorithm works if all the expected data have been received, i.e. if all the buffers are completed. Unfortunately, this is not always the case.

For a non Ice float, not completed buffer can be due to:

- A message that has been lost (because of a float failure), or
- A transmission that is still pending at the time of the decoding session.

Consequently, we introduced a timeout on the 'buffer duration' (time difference between earliest and latest SBD files).

When we move a new mail file to the '*buffer*' list (step #1) the SBD files dated in the interval [*earliest_SBD_file*, *earliest_SBD_file*+`g_decArgo_minSubSurfaceCycleDuration`[(`g_decArgo_minSubSurfaceCycleDuration` = 5 hours) are processed together (steps #5 and #6 of the algorithm) even if the resulting data set is not complete.

Note that, when the float has ended its mission (recovered or lost float), if the last received buffer is not complete, it will never be processed with the algorithm exposed above. To cope with this issue, the user should be informed that, at the end of a decoder session:

- The remaining unprocessed buffers are always processed by the PI decoder,
- The remaining unprocessed buffers can be processed by the DAC decoder with the `PROCESS_REMAINING_BUFFERS` configuration variable.

9.1.2 For Ice floats

Ice floats are equipped with a memory card. When a message cannot be transmitted (in case of a failure of the transmission system or in case the float cannot reach the surface because of Ice coverage), it is stored on the memory card to be transmitted later in delayed mode (during one or more following transmission session(s)). Many transmission sessions can then be needed to transmit the data of a given cycle.

In that case, the buffers are virtually managed through lists of files: a '*spool*' list and one '*buffer#i*' list for each cycle #i under transmission.

Just before decoding, all the received Iridium mail files are in the '`IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO*/*archive*' directory. This has been done:

- By the operator (with the tool ***copy_iridium_mail_files***, see 6.1.2.2) before using the PI decoder,
- By the DAC decoder (after analysis of the rsync log files to identify new incoming Iridium mail files).

At the beginning of the decoding session, the Iridium mail files dated in the interval [*float_launch_date*, *end_decoding_date*] (see 5.1.1 or 5.2.1) are loaded in the '*spool'* list (the date used is the date of the Iridium session, also stored in the Iridium mail file name, see 6.1.2.1).

The processing steps of a decoding session are the following:

1. All the mail files of the *'spool'* list are processed, that is:
    a. Store the Iridium session information (provided in the mail),
    b. Extract the mail file attachment (the SBD file), if any,
    c. Decode the SBD file data.
2. Create the decoding buffers, that is, for each transmission session:
    a. Create a new *'buffer#i'* list for each new cycle data,
    b. Store already seen cycle data in their associated existing *'buffer#i'* list,
    c. For each *'buffer#i'* list, check if its content is completed. If yes, set the *'buffer#i'* as 'completed'.
3. Process the SBD decoded data of the 'completed' *'buffer#i'* lists.

This theoretical algorithm works if all the expected data have been received, i.e. if all the buffers are completed. Unfortunately, this is not always the case.

For an Ice float, not completed buffer can be due to:

- A message that has been lost (because of a float failure or if the memory card is full), or
- A transmission that is still pending at the time of the decoding session.

Consequently, we introduced a timeout on the maximum number of transmission sessions allowed for cycle data to be transmitted (presently set to 3).

Note that, when the float has ended its mission (recovered or lost float), if the last received buffer is not complete, it will never be processed with the algorithm exposed above. To cope with this issue, the user should be informed that, at the end of a decoder session:

- The remaining unprocessed buffers are always processed by the PI decoder,
- The remaining unprocessed buffers can be processed by the DAC decoder with the `PROCESS_REMAINING_BUFFERS` configuration variable.

Note also that, when using the ***decode_provor_2_csv*** program, a CSV file describing each buffer contents is generated in the `DIR_OUTPUT_CSV_FILE` directory.

9.2 Management of Iridium files received from FLOAT_TRANSMISSION_TYPE #2 floats

The Iridium RUDICS data transmitted by `FLOAT_TRANSMISSION_TYPE` #2 floats are received as binary or ASCII files.

The management of these files is thus similar to the one previously exposed (except that we manage binary or ASCII files instead of Iridium mail files).

10 ANNEX C: decode_provor_2_nc_dm, the Delayed Mode DAC decoder

This decoding mode only concerns NKE BIO floats (`FLOAT_TRANSMISSION_TYPE` #2 or #4 floats).

Sometimes, these floats don't transmit all expected messages (or transmit erroneous information), in that case, the real time buffer cannot be processed at the correct time (it is finally processed after a timeout, see Annex B) and the result can be erroneous (particularly when the float is transmitting more than one sub-cycles, i.e. the data collected during more than one ascent).

We haven't succeeded in solving this case in real time, thus we decided to create a Delayed Mode decoding mode.

This DM decoding mode is used by the ***decode_provor_2_nc_dm*** DAC decoder.

Once a float died (or has been retrieved), the set of received float data packets can be definitely grouped in buffers by the operator.

The tool ***split_remocean_sbd_mail_files*** (for `FLOAT_TRANSMISSION_TYPE` #4 floats) or ***split_remocean_rudics_sbd_files*** (for `FLOAT_TRANSMISSION_TYPE` #4 floats) can be used to split the received SBD files in mono-packet files (one file per float packet) and to create a csv file containing the first version of the buffers. The operator can then check this proposal and modify it as needed to create the data sets to be decoded together.

For each float to be decoded in DM mode, we thus create a file to define each buffer contents (each SBD mono-packet file is assigned to a buffer number, or to -1 if the file should be ignored). This ‘buffers definition’ file should be stored in the `DIR_INPUT_DM_BUFFER_LIST` directory.

When the decoder detects that a float should be processed in DM (set in the `DM_FLAG` float decoder configuration information, see 5.2.1):

1. It first checks if the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO/archive_dm* or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO/archive_dm* directory is empty.

   If so (first decoding of the float in DM), it splits the files of the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO*/*archive* or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO*/*archive* directory in mono-packet files and store these files in the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO*/*archive_dm* or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO*/*archive_dm* directory.

2. It then decodes the float according to the ‘buffers definition’ contents.

In this mode, the files stay in the `IRIDIUM_DATA_DIRECTORY`/*IMEI_WMO*/*archive_dm* or `IRIDIUM_DATA_DIRECTORY`/*LoginName_WMO*/`archive_dm` directory and, as the buffers are already defined, the decoding of the float is more efficient.

11 ANNEX D: conditional generation of NetCDF files

The NetCDF files of trajectory data (TRAJ), of profile data (MONO-PROF or MULTI-PROF), of technical data (TECH) and of meta-data (META) are generated according to the configuration flags `GENERATE_NC_TRAJ`, `GENERATE_NC_MONO_PROF`, `GENERATE_NC_MULTI_PROF`, `GENERATE_NC_TECH` and `GENERATE_NC_META` values respectively.

When a flag is set to 0, the corresponding file is not generated.

When a flag is set to 1, the corresponding file is generated (created or updated).

When a flag is set to 2, the corresponding file can be generated or not, depending on the rules detailed below for each float transmission type.

11.1 For Argos floats

If the DAC decoder is used with the parameter **'processmode'** set to **'profile'**:

- The `GENERATE_NC_TRAJ`, `GENERATE_NC_MULTI_PROF`, `GENERATE_NC_TECH` and `GENERATE_NC_TRAJ` flags are set to 0 (i.e. the corresponding files are not generated);
- The `GENERATE_NC_MONO_PROF` flag is set to 2.

When a generation flag is set to 2, the generation rules are the following.

11.1.1 META file

The meta-data file is created if it doesn't exist. It is never updated.

11.1.2 TRAJ, MULTI-PROF and TECH files

The trajectory, multi-profile and technical files are created if they don't exist. They are updated, except when the two following assumptions are true:

- The set of cycle numbers of the Argos HEX data files is identical to the one stored in the CYCLE_NUMBER array of the concerned NetCDF file,
- All the Argos HEX data file system dates are before the DATE_UPDATE date of the NetCDF file.

11.1.3 MONO-PROF file

The rules for generating the mono-profile file are the following:

- If the DAC decoder is used with a parameter **'processmode'** set to **'profile'**:
  - The file is created when the profile has been completed (number of levels equal to the expected one (provided in the technical data)),
  - The file is never updated.
- In all other cases:
  - The file is created if it doesn't exist. If the profile is not complete, a WARNING is written in the log file (giving the number of missing levels),
  - The file can be updated:
    - When the system date of the associated Argos HEX data file is later or equal

to the DATE_UPDATE date of the NetCDF file (i.e. some additional HEX data have been received since the last update of the NetCDF file. The redundancy done with these new data can improve the profile contents),

- When the profile of the NetCDF file was not located but can be now located (from interpolation with the following cycles).

11.2 For Iridium floats

When a generation flag is set to 2, the generation rules are the following.

11.2.1 META file

The meta-data file is created and then updated, each time a decoding buffer contents is processed.

11.2.2 TRAJ, MULTI-PROF an TECH files

The trajectory, multi-profile and technical files are created and then updated, each time a decoding buffer contents is processed.

11.2.3 MONO-PROF file

The mono-profile file is:

- Created when a first decoding buffer contents is processed,
- Updated when the two following assumptions are true:
  - A decoding buffer contents is processed,
  - The profile location is interpolated from a GPS location of the last cycle decoded during the session.

With the DAC DM decoder (***decode_provor_2_nc_dm***) the mono-profile file is created when a first decoding buffer contents is processed, it is never updated.

12 ANNEX E: miscellaneous information

Note that, to correctly use the *loadjson.m* Matlab function on a Linux platform you should set:

```
setenv LANG C
```

in the Linux user configuration.

13 ANNEX F: NITRATE processing

Some BIO floats sample and transmit NITRATE measurements.

In the decoder, NITRATE data is computed from NITRATE sensor raw outputs using the *fitlm* Matlab function (from the Matlab Statistics Toolbox).

If this toolbox is not available in your Matlab distribution, you should use the transmitted NITRATE.

This can be done by setting the `FITLM_MATLAB_FUNCTION_NOT_AVAILABLE` flag to 1 in the Matlab code.

14 ANNEX G: Configuration and Technical label management

14.1 Configuration label management

The configuration labels used in the LAUNCH_CONFIG_PARAMETER_NAME and CONFIG_PARAMETER_NAME variables of the NetCDF META file should be allowed by the Argo project. The approved lists of labels can be found on the Argo data management web page (http://www.argodatamgt.org/Documentation).

The CONF labels used by the decoder are stored in the `DIR_INPUT_JSON_CONF_LABEL_FILE` directory.

There is one file (named *_config_param_name_decId.json*) per decoder Id.

For each configuration parameter entry:

- CONF_PARAM_DEC_ID is the reference of the parameter in the decoder. It should not be modified,
- CONF_PARAM_NAME is the allowed Argo label of the parameter.

14.2 Technical label management

The technical labels used in the TECHNICAL_PARAMETER_NAME variable of the NetCDF TECH file should be allowed by the Argo project. The approved list of labels can be found on the Argo data management web page (http://www.argodatamgt.org/Documentation).

The TECH labels used by the decoder are stored in the `DIR_INPUT_JSON_TECH_LABEL_FILE` directory.

There is one file (named *_tech_param_name_decId.json*) per decoder Id.

For each technical parameter entry:

- TECH_PARAM_DEC_ID is the reference of the parameter in the decoder. It should not be modified,
- TECH_PARAM_NAME is the allowed Argo label of the parameter.

15 ANNEX H: additional tools

A huge (> 200) set of tools is provided with the decoder. Only few of them will be mentioned in this Annex.

Additional information (configuration, usage, results, …) on all the tools can be sent on demand.

15.1 Tools configuration

To configure the access to bathymetric atlases used by the drawing tools, you have:

- To update the line #31 of the *m_etopo2.m* file (of the M_MAP Matlab package) with the ETOPO2 file directory,
- To update the line #77 of the *get_srtm_data.m* file with the SRTM30+ data directory (*SRTM30+\data*).

To configure a given tool you generally have to edit it:

- In the few first lines some items should be updated,
- Some tools use some information of the decoder configuration file (*_argo_decoder_conf.txt*), through the function ***get_config_dec_argo***.

15.2 A selection of useful tools

15.2.1 Visualization tools

The visualization tools are:

- ***nc_trace_disp*** to plot the float displacements,
- ***nc_trace_param*** to plot the profile data,
- ***nc_trace_times_bis*** to plot the cycle timings and profile depths of the entire float mission,
- ***nc_trace_cycle_times*** to plot a “pressure versus time” representation of each cycle.

Configure, start a tool and make the drawing window active. Then press the 'h' key of the keyboard. The help of the tool will appear in the Matlab command window.

15.2.2 NetCDF to CSV conversion tools

The NetCDF to CSV conversion tools are:

- ***nc_meta_2_csv***: for NetCDF META file conversion,
- ***nc_prof_2_csv***: for NetCDF MULTI-PROFILE and MONO-PROFILE files conversion,
- ***nc_prof_adj_2_csv***: for NetCDF MULTI-PROFILE and MONO-PROFILE files conversion (including adjusted values),
- ***nc_tech_2_csv***: for NetCDF TECH file conversion,
- ***nc_traj_2_csv***: for NetCDF TRAJ file conversion,
- ***nc_traj_adj_2_csv***: for NetCDF TRAJ file conversion (including adjusted values).

These tools can be used to convert (part of) Argo NetCDF V3.1 (or V3.2 for trajectory) file contents to CSV files (easy to study using Excel filters for example).

15.2.3 Argos cycle file management tools

The Argos cycle files (stored in `DIR_INPUT_HEX_ARGOS_FILE_FORMAT_1`/*ArgosId*) may need to be modified. A set of tools has been implemented for that:

- ***check_argos_cycle_files***: to check the set of cycle files of a float,
- ***set_cycle_number_of_argos_cycle_file***: to set the cycle number of a cycle file,
- ***freeze_argos_cycle_files***: to exclude a cycle file from the decoding process,
- ***concat_argos_cycle_files***: to concatenate cycle files data,
- ***shift_cycle_number_of_argos_cycle_files***: to shift cycle number of cycle files.

15.2.4 copy_iridium_mail_files, copy_remocean_sbd_files, copy_cts5_files, copy_apx_iridium_rudics_files, copy_apx_apf11_iridium_rudics_files or copy_nemo_files

Tools used to make a copy of Iridium mail files or Iridium RUDICS files from their repository to the directory associated to each float.

15.2.5 nc_add_rtqc_flags_prof_and_traj

The RTQC flags can be added by the decoder just after the decoding (if `APPLY_RTQC = 1`).

You can also apply RTQC tests to existing V3.1 (or V3.2 for trajectory file) NetCDF Argo files using this tool (the tests to be performed should be set to 1 in the `testToPerformList` cell array).

15.2.6 nc_check_file_format

This tool is used to check that the NetCDF decoded files are compliant to the Argo V3.1 format with the GDAC Argo checker (the last version of the checker should be first downloaded at http://usgodae.org/pub/outgoing/argo/etc/FileChecker/ and installed).