argo data management

Argo data management
DOI: http://dx.doi.org/10.13155/29824

Argo DAC Trajectory Cookbook

Version 6.1

November 2022

ARGO

part of the integrated global observation strategy

Argo data management

Argo DAC trajectory cookbook

Authors: Megan Scanderbeg / Scripps Institution of Oceanography, Jean-Philippe Rannou / ALTRAN, Justin Buck / BODC, Claudia Schmid / AOML, John Gilson / Scripps Institution of Oceanography, Dana Swift / University of Washington, Kanako Sato/ JAMSTEC, Tanya Maurer / MBARI

How to cite this document

Megan Scanderbeg / Scripps Institution of Oceanography, Jean-Philippe Rannou / ALTRAN, Justin Buck / BODC, Claudia Schmid / AOML, John Gilson / Scripps Institution of Oceanography. , Dana Swift / University of Washington, **Argo DAC trajectory cookbook**. http://dx.doi.org/10.13155/29824

Table of contents

History of the document

Version	Date	Comment
1.0	June 2012	Original version sent around for comment
1.1	August 2012	Comments from John Gilson, Jean-Philippe Rannou, Justin Buck, Kanako Sato, Mizuho Hoshimoto, Bernie Petolas
1.2	November 2012	Updated measurement code table to three places to allow for many more codes. Added satellite name, error ellipse variables, condensed final questions in preparation for ADMT meeting
1.3	February 2013	Updated with feedback from ADMT-13 meeting
1.4	April 2013	Updated format in anticipation of publication
1.5	April , 2014	Updated code for TET estimation for APEX floats with Argos transmission, condensed float tables in Annex I. Updated cycle definition, added new paragraph describing core-Argo and B-Argo trajectory files. Updated MC codes for TST and AST for APEX floats.
2.0	June 2014	Officially given a DOI and named as Trajectory Cookbook
3.0	September 2014	Large changes to APEX float section after numerous discussions with Dana Swift, author of most of the APEX float software
3.2	February 2015	Move estimation procedures for APEX floats to Annex J. Removed all Annex I tables except PROVOR, ARVOR, NINA
3.3	May 2015	Added instructions on PUMPED and UNPUMPED CTD data for NKE floats Updated SOLO-II tables Updated NOVA tables Updated NINJA tables Added Deep NINJA tables
4.0	June 2015	Removed all sections related to profile files for the new DAC Profile cookbook
4.1	November 2015	Standard Reference ID now optional
5	August 2016	Updated instructions on CONFIG_MISSION_NUMBER for launch and cycle 0
6	August 2019	Updated color scheme on how to assign measurement codes Removed 702,704 for Iridium RUDICs floats Updated definition of 600/ ascent end time Combined APEX APF9i and NAVIS tables Added PROVORCTS3 and Arvor Iridium tables Added Arvor table Added estimated, Iridium and RAFOS positions Added relative measurement codes for surface measurement sequences
6.1	November 2022	Updated Cookbook with instructions on how to include BGC and other measurements stored in the combined v3.2 trajectory files. Generalized sections to surface times and locations, timing measurements, and sensor measurements. Updated introcution and reiterated that Standard Format ID is no longer maintained. Updated Grounded flags section

Preface

This document is still in progress. As such, there are highlighted sections of text throughout that need to be addressed. Yellow highlighting means this is a topic open to discussion - some things are known about this topic, but agreement needs to be reached. Green highlighting signifies a question that needs to be answered by a float expert or float manufacturer. Red highlighting means the issue needs a solution and nothing has been suggested yet.

One proposed entry point into this cookbook is through through the Argo Trajectory Measurement Code tables which are a companion to the DAC Trajectory Cookbook and are available via the same DOI: 10.13155/29824 (doi.org).These tables are organized by float type and model and each row describes how a different measurement code should be filled for that specific float model. This allows DACs to look up their float model and read across that row to find out how to fill each variable. For further explanation, a DAC could go to the section of this cookbook corresponding to the float type of interest and find the same tables, but with more information to help understand how to find and process the data needed for each measurement code.

1 Introduction

This DAC Trajectory Cookbook includes instructions for DACs on how to calculate different variables for the Argo trajectory files. . These files are primarily intended to store information and data related to the timing and positioning of the float at various stages. It is separate from other data manuals because users interested in profile data do not need to understand all these details, but it is important that all DACs calculate the variables in the same manner. For readers who have no familiarity with trajectory files and their associated variables, it is essential to first read section 2.3 of the Argo User Manual (http://dx.doi.org/10.13155/29825) prior to proceeding with this document. Additionally, note that during the ADMT-21 meeting held at LOV in Villefranceh in October, 2019, the proposal to combine the previously separate core-Argo and B-Argo trajectory files was accepted by the community. The processing instructions outlined herein support the new format type v3.2 which accommodates trajectory information related to both core and BGC sensor operations.

Concerning Argo trajectory data specifically, correct data processing requires a good knowledge of the float platform capabilities. Each float type has its own behavior and within a given type, each float version provides specific data useful to trajectory determination. A large part of the document is based on the work done since 2007 on Argo trajectory data in the framework of the ANDRO project by Jean-Philippe Rannou and Michel Ollitrault. Dana Swift has also provided input on APEX float timing. He has written much of the firmware on APEX floats and was able to provide detailed information on APEX floats.

The algorithms proposed by Jean-Philippe Rannou and Michel Ollitrault are included in the cookbook, but are not mandatory. While they have been designed to be efficient and robust enough to be deployed in a real time data flow without a visual check, the timing information they provide is an estimate.

If DACs are interested in PROVOR and ARVOR floats manufactured through 2014, the best entry point is through ANNEX H: Cookbook entry point. In the ANNEX, the presented tables, one can find, for a given float version, all useful information that can be decoded, computed or estimated from transmitted float data and a link to the concerned paragraph(s) in the document.

For all PROVOR and ARVOR floats manufactured after 2014 and all other float types, it is suggested to look for the appropriate float type in each category of Section 2 (2.1: surface fixes, 2.2: cycle timing, 2.4: sensor measurements). Each float type has a section, often with tables to explain how to fill the which measurement codes for each float type and version. . A companion to this is the Argo Trajectory Measurement Code Tables (accessible via the same DOI as the DAC Trajectory Cookbook)

which summarize this information in tables and can be referred to for an overview of which measurement codes are filled and how for each float type.

To matchthe Format ID in Annex H to the float version list, please access this spreadsheet established by the Mathieu BELBEOCH and stored at OceanOPS: https://docs.google.com/spreadsheet/ccc?key=0AitL8e3zpeffdENUQmszRlY3djYweGZhbnBZSU1fTFE&usp=sharing. If the link does not work when clicking on it, please copy and paste it into your browser. This list was created in 2012 and accurately describes the float types available at that time. Since then, most float types have been updated and is no longer actively maintained. If you have questions about this list, please contact Victor TURPIN (vturpin@ocean-ops.org) or Megan SCANDERBEG (mscanderbeg@ucsd.edu) .

Many of the concerned float versions are obsolete for real time processing. However, even if these floats are no longer active, it is important to document how to decode the float data and create trajectory files for historical purposes.

The float types presented in this document include:

- The PROVOR float including PROVOR, PROVOR-MT and ARVOR floats in their Argos and Iridium versions
- APEX
- NINJA, Deep NINJA
- SOLO
- SOLO-II
- NEMO
- NAVIS
- NOVA

1.1 Cook book usage and update

Each new float version must be fully studied, decoded and the results analyzed (in a "trajectory" point of view) before being added in the DAC Trajectory Cookbook and the accompanying Argo Trajectory Measurement Code Tables.

This detailed information should provide each DAC with the ability to homogeneously process NetCDF TRAJ file contents no matter whether or not the DAC has prior knowledge about a float's trajectory data.

For many float types, the ANNEX H tables are not up to date and should be used with caution. The PROVOR and ARVOR tables are updated through 2014 and are the most reliable.

For float types other than PROVOR and ARVOR manufactured before 2014, it is suggested to search for the float type in Section 2. Section 2.1 describes how to fill surface positions and focuses largely on the type of positioning systems: Argos, GPS, estimations and RAFOS. Section 2.2details how to fill in the cycle timing variables by float type.

1.2 Real time TRAJ file expected contents

Three main types of data are expected to be stored in the real time NetCDF TRAJ files:

- Surface fixes: Argos, GPS, Iridium or estimated locations of the float surface trajectory,
- Cycle timings: The dated main cycle events associated (when available) with their sensor measurements,

- Other sensor measurements; including:
    - Drift phase CTD and/or BGC measurements: possibly dated measurements sampled during the drift phase at parking depth (used to determine ie the deep displacement immersion),
    - CTD and/or BGC measurements made outside the drift phase: depending on float versions, sensor make measurements during ascent, descent or on the surface; if these measurements include timing information, it helps define the vertical movements of the float and the associated rates.

1.2.1 Duplicated times

The cycle timing dates must be duplicated in the TRAJ files. They should be stored in the N_CYCLE arrays **and** in the N_MEASUREMENT arrays with the associated MEASUREMENT_CODE value. All Primary and Secondary Measurement Code (MC) events (see Annex F) that **are experienced by the float** are required to be present in the N_MEASUREMENT array and redundantly in the N_CYCLE variables. All other codes are voluntary.

If the float experiences an event **but the time is not able to be determined**, then a *_STATUS = '9' is used. This indicates that it might be possible to estimate in the future and acts as a placeholder.

In the N_CYCLE variables, if the float does not experience an event then *_STATUS = 'FillValue' is used. Only events that are experienced by a float are recorded in the N_MEASUREMENT array so status='FillValue' is not used in those variables.

1.2.2 Data resolution

The decoded data can sometimes have unusual resolutions depending on the measurement code. Store the nominal resolution in the <PARAM>:resolution attribute. If the resolution differs by measurement code, provide this information in the "COMMENT_ON_RESOLUTION" attribute for the concerned variable and as a global attribute.

See section 2.3.5.1 of the Argo User's Manual for a more detailed example.

Examples of differing resolutions are as follows:

- Dates: reported timestampes might have a 1 minute or 6 minutes resolution,
- Pressures: the PROVOR technical and spy pressures are sometimes given in bars rather than decibars, this is also the case for APEX descending pressure marks.

1.2.3 N_CYCLE Array

For the N_CYCLE array, there are a few details to mention here about the float launch and cycle 0. Specifically, launch information should not be included in the N_CYCLE array. For cycle 0, CONFIG_MISSION_NUMBER should be fill value, but the other N_CYCLE variables should be filled appropriately, including the GROUNDED flag.

1.2.4 Cycle number management in RT TRAJ

Cycle numbers

A cycle is defined as a series of actions made by a float and includes either a descending profile or an ascending profile (or, rarely, both); it may also include immersion drift or surface drift. An Argo cycle starts with a descent toward deep water, usually from the surface. It ends with the next ascent to shallow water and data transmission (in some situations or for some floats, data transmission may not

always occur). Each cycle of a float has a unique cycle number, increased by one after each ascent to shallow water. For most floats, this will be the cycle number transmitted by the float. In some cases, this number will need to be calculated by the operator. Simple checks on cycle number can be performed in real time.

For floats that provide cycle number, DACs should compare the provided cycle number with the expected cycle number. If they agree, the provided cycle number will be stored in CYCLE_NUMBER and CYCLE_NUMBER_INDEX variables. If they disagree, cycle number should be computed to be coherent with time versus cycle duration. Care should be taken not to overwrite a current cycle.

For floats that do not provide cycle number, cycle number should be computed to be coherent with time versus cycle duration. These cycle numbers should be stored in CYCLE_NUMBER in real time.

Both CYCLE_NUMBER and CYCLE_NUMBER_INDEX need to be filled in real time. The cycle number in CYCLE_NUMBER must match the profile cycle number, which is the number recorded in the CYCLE_NUMBER variable in the profile file. If a mismatch is detected between a trajectory cycle number and a profile cycle number, the trajectory cycle number must be changed to match the profile file cycle number and replaced on the GDAC.

As mentioned previously, launch information should not be in the N_CYCLE array or in CYCLE_NUMBER_INDEX.

CYCLE_NUMBER_INDEX indicates which cycle number information is contained in that index of the N_CYCLE array. For example, CYCLE_NUMBER_INDEX(4)=3 means the 4th element of all N_CYCLE variables is associated with the WMO_003.nc profile file. Additionally, all the elements of the N_MEASUREMENT variables for which CYCLE_NUMBER = 3 are likewise associated with the 4th N_CYCLE elements and with the WMO_003.nc profile file. This stops confusion over which index in the N_CYCLE array corresponds to which cycle number in the N_MEASUREMENT array.

The CYCLE_NUMBER_ADJUSTED and the CYCLE_NUMBER_INDEX_ADJUSTED variables will contain a cycle numbering which has been assessed and may be adjusted to be correct, especially for the purpose of trajectory calculations.

If a cycle is recovered during delayed mode, DACs must choose to either (a) create a new profile file and renumber all profile files accordingly and then rewrite the trajectory file with the changes in CYCLE_NUMBER & CYCLE_NUMBER_INDEX to match the profile files OR (b) not create a new profile file and add the new cycle into the CYCLE_NUMBER_ADJUSTED and CYCLE_NUMBER_ADJUSTED_INDEX variables. Two examples of case (b) are below.

The first example is where cycle number 5 is recovered either in delayed mode. The cycle number variables must be rewritten as follows:

CYCLE_NUMBER	1, 2, 3, 4, _, 6, 7, 8, 9, 10, 11,… ,
CYCLE_NUMBER_INDEX	1, 2, 3, 4, _, 6, 7, 8, 9, 10, 11,…,
CYCLE_NUMBER_ADJUSTED	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, _
CYCLE_NUMBER_ADJUSTED_INDEX	1, 2, 3, 4, 5, 6, 7, 8 9, 10, _

Here, FillValue is added to CYCLE_NUMBER and CYCLE_NUMBER_INDEX to indicate that no profile files exist with cycle number 5. The trajectory file must be rewritten to add in the new cycle number information and any other information recovered for that profile.

A second example of errors that might be discovered in cycle number in delayed mode involves floats that do not send cycle number and for which cycle number must be calculated. In this situation, there

are times when cycle numbers are incorrectly skipped. Here, cycle number 5 was incorrectly skipped in real time and added back in delayed mode:

CYCLE_NUMBER	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12,…
CYCLE_NUMBER_INDEX	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12,…
CYCLE_NUMBER_ADJUSTED	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, _, _
CYCLE_NUMBER_ADJUSTED_INDEX	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, _, _

Missing cycles

A cycle is defined as a series of actions, including collection of data, made by a float that ends with transmission of data, or the attempt to transmit data. If the float fails to collect or transmit data, a cycle has not occurred and can be defined as missing.

Missing cycles should NOT be stored in the TRAJ file. No place holders are necessary and will not work with the new TRAJ file.

1.2.5 Clock offset

Some Argo float versions provide times for dated events or dated measurements. Over time, the float's clock may drift. Clock drift can be defined as the drift of the clock in hours/ minutes/ seconds per year. To correct for this, we must apply a clock offset where clock offset is defined as a measurement, done at a given time, of the offset of the clock due to clock drift. Thus a clock offset should be estimated for each of these float times.

Note that clock offset can also embrace a clock that has not been correctly set or a clock that has been set in local time. Of course, in these cases, clock offset is not only revealing a drift of the float clock...

Float clock offset is defined as: Float clock offset = Float time - UTC time.

A good estimate of the clock offset can be obtained when the float transmits its Real Time Clock (RTC) time in the technical data. It can then be compared to the time from Argos of the corresponding message to compute a clock offset for all the float times of the concerned cycle.

Unfortunately this is not always the case, some floats do not transmit their RTC time and even if they do, this RTC time is not always received.

Here are some remarks on RTC time transmitted by Argo float versions:

- For APEX Argos floats: the float times come from float versions which send the RTC time only once in the test message (thus around the launch time),
- For APEX Iridium floats: the float times are given with the RTC time for every cycle, thus they can always be corrected,
- For PROVOR and ARVOR Argos floats: the RTC time is in the technical message. If we do not receive the technical message, the float times cannot be computed for this cycle. Thus, when we cannot compute the clock offset, there are no float times to correct from clock offset,
- For PROVOR Iridium floats: the RTC is set each cycle (using GPS time). Thus clock offset is considered to be equal to zero,
- For NINJA Argos floats: the RTC time is provided each cycle but the corresponding message can be missing (not received),
- For SIO and WHOI SOLO floats: no float time is transmitted, but the SIO float clock is reset each time, making clock offset essentially non-existent.
- For SOLO-II: float time is transmitted. In addition the float clock is reset each surfacing, making clock offset essentially non-existent.

- For NEMO Argos floats: the decoding has been done by Optimare and we do not know how they manage clock offsets.
- For NEMO Iridium floats: the RTC time is in the technical message. If we receive the technical message, we can correct RTC time by using GPS time.

1.2.5.1 How to put clock offset into trajectory file in real time

If a float can be corrected for clock offset in real time, DACs should determine the drift and adjust the time (inclusive of adjustment of zero). The corrected time should go in the JULD_ADJUSTED (N_MEASUREMENT) variable.

The JULD_ADJUSTED_STATUS should be set to "3" if the clock offset is computed from the RTC time.

The JULD_ADJUSTED_QC should also be filled.

Simultaneously, the DATA_MODE should be marked as "A" indicating an adjusted float, and the CLOCK_OFFSET (N_CYCLE) variable should be appropriately filled.

If a float cannot be corrected for clock offset in real time, the JULD_ADJUSTED* variables and the CLOCK_OFFSET variable should all be fill value.

1.2.5.2 How to put clock offset in trajectory file in delayed mode

If the float is corrected for clock offset in delayed mode, the corrected time should go in the JULD_ADJUSTED (N_MEASUREMENT) variable.

The JULD_ADJUSTED_STATUS should be set to "3" if the clock offset is computed from the RTC time or to "1" if it is estimated using information sent by the float or if it is estimated using procedures that rely on typical float behavior.

The JULD_ADJUSTED_QC should also be filled.

The clock offset itself goes in the CLOCK_OFFSET(N_CYCLE) variable and the DATA_MODE should be marked as "D" indicating a delayed mode correction.

If a float cannot be corrected for clock offset in delayed mode, the CLOCK_OFFSET variable should be fill value. The JULD_ADJUSTED* variables may be filled if other estimates are done on the timing information not related to clock offset.

2 Trajectory files

2.1 Surface fixes

2.1.1 Launch position and time

The launch position and time values should be duplicated from the META file to the TRAJ file.

They should be stored as the first LATITUDE, LONGITUDE and JULD of the N_MEASUREMENT array with:

- CYCLE_NUMBER = -1,
- POSITION_QC = 0,
- POSITION_ACCURACY = _FILLValue,

- MEASUREMENT_CODE = 0
- JULD_STATUS = 4 - determined by satellite if that can be done, otherwise ‘9’ if the JULD is FILLValue

The launch time should be as reliable as possible (because it is used in Argos surface location selection, see §**Erreur ! Source du renvoi introuvable.**). Therefore, methods, based on transmitted information, can be used to check the launch time. Once the launch position has been checked, its QC should be set to 1.

Launch information should not be duplicated in the N_CYCLE array. It goes only in the N_MEASUREMENT array.

2.1.2 For Argos APEX floats

Argos Apex floats send the information "Time from startup" in the test message. This information can be used to compute the STARTUP_DATE (in the metafile) of the float (using the time of the Argos message used for "Time from startup" information decoding).

For APF9a/t floats, all firmware revisions are capable of self-activation via the pressure-activation mechanism. For floats that self-activate, the “time from startup” is AFTER launch time. For floats that were manually started while still on-board the ship, the start-time is before launch time.

For APF8 floats, some firmware revisions had the pressure-activation mechanism while others did not. This means the “time from startup” can either be before launch time, if manually started, or after launch time if self-activated.

2.1.3 Other surface location fixes

All surface locations **occurring after the launch time** should be stored with their full resolution (some DACs do not store the seconds of the location time) in the TRAJ file.

Required data:

- JULD
- JULD_QC
- JULD_STATUS
- JULD_DATA_MODE
- LATITUDE
- LONGITUDE
- POSITION_ACCURACY
- POSITION_QC
- SATELITE_NAME
- AXIS_ERROR_ELLIPSE

2.1.3.1 Argos surface locations

Some Argos locations can be computed twice by CLS in (near) real time and the improved results sent again. The Argos location data set of a given cycle should be updated from all CLS incoming data at least 2 days after the theoretical end of the Argos transmission (*cf. ADMT12 action#53*).

The real time quality control test #20 (Questionable Argos Position test, also in ANNEX G: Implementation of the JAMSTEC trajectory quality control method) should be used on surface Argos locations to define the position QCs.

2.1.3.2 GPS surface locations

GPS position(s) and time(s) should be included as the position fixes and times (see required data in section 2.1.3) when available.

- POSITION_ACCURACY = ‘G’
- JULD_STATUS = ‘2’: transmitted by float
- AXES_ERROR_ELLIPSE variables can be filled the estimated error.

All the GPS positions should be stored in the TRAJ file with a measurement code of 703

For APEX floats

GPS locations provided in log file **and** message files should be merged. One suggested way to find the fixes for APEX 001087 floats, is to parse the log file using:

GPS_FIX = ['GpsServices() Profile ' sprintf('%d', a_cycleNum) ' GPS fix obtained in'];

For PROVOR floats

The time of the GPS position provided in the technical message is the float's time and date (also provided in this technical message).

2.1.3.3 Iridium surface locations

Iridium fixes can be included in the trajectory files. In that case:

- POSITION_ACCURACY = ‘I’
- AXES_ERROR_ELLIPSE variables can be filled with the estimated error
- POSITIONING_SYSTEM = ‘GPS’ (because this is not an N_MEASUREMENT array variable, it needs to be recorded as ‘GPS’ and users can determine it is an Iridium location based on the POSITION_ACCURACY)
- POSITION_QC would be a ‘1’ or ‘2’

All the Iridium positions should be stored in the TRAJ file with a measurement code of 703.

2.1.3.4 Estimated positions

The Argo Program has decided to allow estimated positions in trajectory files as of ADMT-20 in December 2019. This means that if no positions are available for a profile/surface interval, DACs and DM operators can choose to enter in an estimated position. In v3.1 and v3.2, the POSITIONING_SYSTEM variable is not N_MEASUREMENT long, so it will not be possible to indicate for each position which system, or lack thereof, was used to determine the position. So, the following guidelines have been developed to try and provide as much information to the user as possible to indicate an estimated position.

If no positions are available, but the DAC or DM operator chooses to put in an estimated position with MC = 703:

- LATITUDE and LONGITUDE
- POSITION_ACCURACY would be ‘U’
- POSITION_QC would be ‘8’
- AXES_ERROR_ELLIPSE variables can be filled with an estimated error
- POSITIONING_SYSTEM will record ‘GPS’ or ‘Argos’ or whichever is the main positioning system of the float

If there are GPS fixes, Iridium positions and estimated positions all in one trajectory file, POSITIONING_SYSTEM will still record ‘GPS’.

2.1.3.5 RAFOS positions

If RAFOS positions are calculated during the float’s drift phase, those positions should be stored as follows:

- LATITUDE and LONGITUDE
- POSITION_ACCURACY would be ‘R’
- POSITION_QC would be ‘1’, ‘2’, ‘3’ or ‘4’
- POSITIONING_SYSTEM would be main surface positioning system (Argos, GPS, etc)
- AXES_ERROR_ELLIPSE variables can be filled with estimated errors
- JULD should be filled with the corresponding time
- MEASUREMENT CODE would be 275 to indicate it took place during drift

Intermediate information used to calculate the RAFOS positions should be stored in the AUX directory

2.2 How to calculate cycle timing variables

Each Argo float cycle is composed of programmed events. Depending on float type, some of these events can be dated and associated sensor measurements can be provided. The following figure shows an example cycle, with the times ordered for Argos satellite communications. For Iridium floats, the order of surface events may be different.

The sixteen following timed events can be highlighted.

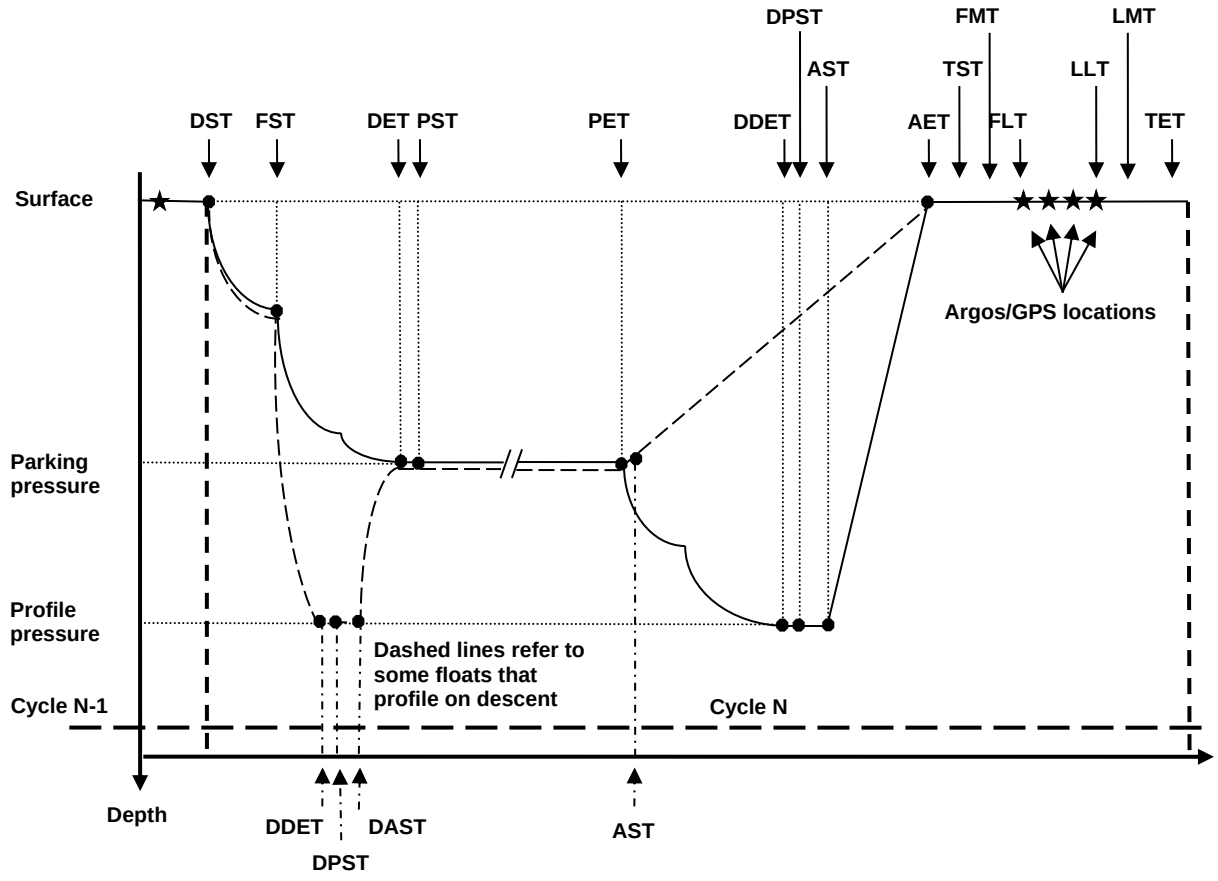


Figure 1: Figure showing float cycle and the cycle timing variables. Floats can profile either on descent or ascent. Most floats profile on ascent. Their path is shown with a solid black line. Some floats profile on descent. One such float, the new SOLO-II Deep float, has a cycle as shown by the dashed line.

Floats that profile on ascent would have the following primary cycle timings:

DST, DET, PET, DDET, AST, AET, TST, all surface times and TET

Floats that profile on descent might have the following cycle timings:

DST, DDET, DAST, DET, PET, AST, AET, TST, all surface times and TET

NOTE: if a float is programmed to experience a primary cycle timing event, but no timing information is sent back and no estimate is possible, fill value should be inserted in the JULD array with the measurement code corresponding to the primary cycle timing event. Examples of this include:

- SOLO and APEX APF8 floats which send back no cycle timing information should have fill value for the primary measurement codes unless an estimated time can be determined

- Ice detection floats that detect ice at the surface and then do not surface should have fill value for measurement codes 600-800 for the affected cycles. For some Iridium floats, a time is available for 600.

NOTE: the diagram above shows the chronological order of timing events for an Argos float. Iridium floats have a different chronological order of timing events. In either case, times in the JULD variable should be arranged chronologically. The only exception to this if a clock offset has been applied and

then the JULD variable may have an inversion, but the JULD_ADJUSTED variable must be arranged chronologically.

Time	MC	Long name	N_CYCLE variable name	Description
DST	100	Descent Start Time	JULD_DESCENT_START JULD_DESCENT_START_STATUS	Time when float leaves the surface, beginning descent.
FST	150	First Stabilization Time	JULD_FIRST_STABILIZATION JULD_FIRST_STABILIZATION_STATUS	Time when a float first becomes water-neutral.
DET	200	Descent End Time	JULD_DESCENT_END JULD_DESCENT_END_STATUS Note: Float may approach drift pressure from above or below.	Time when float first approaches within 3% of the eventual drift pressure. Float may be transitioning from the surface or from a deep profile. This variable is based on pressure only and can be measured or estimated by fall-rate. In the case of a float that overshoots the drift pressure on descent, DET is the time of the overshoot.
PST	250	Park Start Time	JULD_PARK_START JULD_PARK_START_STATUS	Time when float transitions to its Park or Drift mission. This variable is based on float logic based on a descent timer (i.e. SOLO), or be based on measurements of pressure (i.e. Provor).
Note on DET and PST: DET and PST might be near in time or hours apart depending on float model and cycle-to-cycle variability. PI has judgment call whether DET~=PST.				
PET	300	Park End Time	JULD_PARK_END JULD_PARK_END_STATUS	Time when float exits from its Park or Drift mission. It may next rise to the surface (AST) or sink to profile depth (DDET)
DDET	400	Deep Descent End Time	JULD_DEEP_DESCENT_END JULD_DEEP_DESCENT_END_STATUS	Time when float first approaches within 3% of the eventual deep drift/profile pressure. This variable is based on pressure only and can be measured or estimated by fall-rate.
DPST	450	Deep Park Start Time	JULD_DEEP_PARK_START JULD_DEEP_PARK_START_STATUS	Time when float transitions to a deep park drift mission. This variable is only defined if the float enters a deep drift phase (i.e. DPST not defined in cases of constant deep pressure due to bottom hits, or buoyancy issues)
DAST	550	Deep Ascent Start Time	JULD_DEEP_ASCENT_START JULD_DEEP_ASCENT_START_STATUS	Time when float begins its rise to drift pressure. Typical for profile-on-descent floats.
AST	500	Ascent Start Time	JULD_ASCENT_START JULD_ASCENT_START_STATUS	Time when float begins to return to the surface.
AET	600	Ascent End Time	JULD_ASCENT_END JULD_ASCENT_END_STATUS	Time when float ends the ascending profile phase and starts the next phase of the cycle.
TST	700	Transmission Start Time	JULD_TRANSMISSION_START JULD_TRANSMISSION_START_STATUS	Time when float begins transmitting.
FMT	702	First Message Time	JULD_FIRST_MESSAGE JULD_FIRST_MESSAGE_STATUS	Earliest time of all messages received by telecommunications system
FLT	703	First Location Time	JULD_FIRST_LOCATION JULD_FIRST_LOCATION_STATUS	Earliest location of all float locations.
LLT	703	Last Location Time	JULD_LAST_LOCATION JULD_LAST_LOCATION_STATUS	Latest location of all float locations.
LMT	704	Last Message Time	JULD_LAST_MESSAGE JULD_LAST_MESSAGE_STATUS	Latest time of all messages received by telecommunications system
TET	800	Transmission End Time	JULD_TRANSMISSION_END JULD_TRANSMISSION_END_STATUS	Time when floats stops transmitting.

Table 1: Descriptions of cycle times shown in the previous figure

All these times are in both the N_MEASUREMENT and the N_CYCLE variable groups of the TRAJ file. These times should be included in chronological order in both the cases. This means events may not occur in the same order as in the table above, as it is developed around an Argos float. For an Iridium float with a GPS fix taken before starting Iridium transmission, the measurement codes at the surface might look like this: 703, 700, 702, 704, 800

The main times of a cycle can be separated in two parts:

- Positioning and transmission system times: FMT, FLT, LLT and LMT,
- Times of float events: the other ones.

The trajectory file checker will be checking for ascending times in JULD and JULD_ADJUSTED. Due to float clock drift, the times from the float cannot be compared to times reported by the satellite systems to check for ascending times in JULD and JULD_ADJUSTED. Therefore, two ascending time checks will be done on JULD and JULD_ADJUSTED. One check for ascending times in JULD and JULD_ADJUSTED for times associated with the float (ie, *STATUS flags equal to 1, 2, or 3). The second check would be for ascending times in JULD and JULD_ADJSUTED for times not associated with the float (ie, *STATUS flags equal to 4).

2.2.1 Positioning system and transmission system times

The FMT, FLT, LLT and LMT times only depend on the positioning system and the transmission system used by the float. Additionally, the order these times and positions occur in chronologically depends on the system being used. The order is completely different for Argos than for Iridium and in many cases of Iridium usage, there is only a single position and time fix during the entire surfacing period. In addition, for Iridium floats with RUDICS, there is usually no additional timing information available detailing when Iridium first sent a transmission or when the final transmission was sent. So, for Iridium floats which send back only timing associated with the position fix(s) measurement codes (FMT/702 and LMT/704) are not used. For Iridium floats which send back additional timing information like when the first transmission started or when the last transmission ended, please fill all FMT/702 and LMT/704.

2.2.1.1 For Argos floats

See Annex A for Argos message time and Argos location time illustration as received from CLS. Status variables should be a "4 (value is determined by satellite)".

2.2.1.1.1 First and last message times

All Argos message times should be collected and the maximum and minimum values stored as FMT and LMT (*we cannot assume that data are received from CLS in chronological order*).

If only one message has been received for a given cycle, its time should be duplicated in FMT and LMT.

NOTE: Some DACs already store FMT and LMT in the TRAJ file but some of them are erroneous because they correspond to ghost Argos messages.

As reliable FMTs and LMTs are crucial for other times estimation (such as APEX DST), we must think of a robust method to reject these ghost messages in real time.

The best method for now is for floats which use a CRC in their Argos messages, to use in FMT and LMT only Argos messages that passed the CRC check. *From AOML: [For known cycle times one can use that information plus an analysis of all cycles in the raw data to identify ghosts. We developed a program for this, but are not yet using it in operations. We could share that program with others once we are convinced it works.]*

2.2.1.1.2 First and last location times

All Argos location times should be collected and the maximum and minimum values stored as FLT and LLT (*we cannot assume that data are received from CLS in chronological order*).

If only one location has been computed for a given cycle, its time should be duplicated in FLT and LLT.

2.2.1.2 For Iridium floats

The chronological order of the events on the surface does not follow the numerical order from the measurement code table which was designed more with the Argos system in mind. For example, usually the GPS fix comes first and then the float begins transmitting to Iridium. Often several messages are sent to and received from Iridium and then the float stops transmitting. If there are times associated with the transmissions sent by Iridium (SBD floats and some others), , following the chronological order of the times, the measurement codes would be 703, 700, 702, 704, 800. For Iridium floats that do not send times associated with transmissions sent by Iridium, (usually RUDICS), the measurement codes would be 703,700, 800.

2.2.1.2.1 First and last message times

For NEMO and SOLO floats

Use the "Time of Session" information, provided in all the Iridium e-mails received for each cycle, as the float message time.

For NAVIS floats

For Iridium, there are two values transmitted that replace the Argos transmission times. When the float reaches the surface, it acquires a GPS position. The time to do this is represented by TTFF (in seconds). After the GPS is acquired, then the Iridium transceiver is activated. The SBDT is the transmission time of the first Iridium packet (housekeeping packet). This is to give an indication of the transmission throughput as the housekeeping is a constant size as opposed to the other packets. After completion of the transmission, a satellite check is done to look for incoming commands. If there is one, it is processed and then the float starts its next profile. Note that SBDT refers to the previous profile, not the current one, as it is calculated AFTER the Iridium transmission takes place.

First Message Time is TST + TTFF.

Last Message Time is the same as LLT.

For NOVA floats

For Iridium, there are two values transmitted that replace the Argos transmission times. When the float reaches the surface, it acquires a GPS position. The time to do this is represented by TTFF (in seconds). After the GPS is acquired, then the Iridium transceiver is activated. The SBDT is the transmission time of the first Iridium packet (housekeeping packet). This is to give an indication of the transmission throughput as the housekeeping is a constant size as opposed to the other packets. After completion of the transmission, a satellite check is done to look for incoming commands. If there is one, it is processed and then the float starts its next profile. Note that SBDT refers to the previous profile, not the current one, as it is calculated AFTER the Iridium transmission takes place.

First Message Time is TST + TTFF.

Last Message Time is the same as the FMT - there is only one GPS fix.

For PROVOR and APEX floats

For Iridium SBD floats, use the "Time of Session" information, provided in all the Iridium e-mails received for each cycle, as the float message time.

For Iridium RUDICS floats, no first and last message time is available, so these are not included in the trajectory file.

2.2.1.2.2 First and last location times

All Iridium location times should be collected and the maximum and minimum values stored as FLT and LLT.

If only one location has been collected for a given cycle, its time should be duplicated in FLT and LLT.

2.2.2 Times of float events

Each float type, and sometimes each model of float type, has different instructions on how to fill in the timing variables in the trajectory file. Remember that all mandatory cycle timing variables must be filled and are in both the N_MEASUREMENT and N_CYCLE arrays. If it is not possible to fill this time, even by an estimation, fill value must be used in both arrays.

2.2.3 APEX floats

2.2.4 APEX floats with the APF8 controller board

The cycle timing information transmitted by APEX floats with the APF8 controller board is limited, so **no event times are directly available**.

To compute or estimate the cycle times for APF8 floats, we must use "external" methods based on the float functioning. Some of the same methods can be used for the APF9a or APF9t floats, but should not replace the transmitted times.

Two main methods have developed (from work done for ANDRO) to be efficient and may be robust enough to be implemented in real time.

- The first one, based on float functioning, can be used to estimate TET,
- The second one, based on float transmission strategy, can be used to compute TST.

The first method to compute TET seems less robust for real time application and some DACs may choose not to estimate TET in real time. If this is the case, then the delayed mode operator may choose to estimate a TET in delayed mode where more time can be spent visually inspecting each float's TET estimate. Even though this is a mandatory time, if the DAC or delayed mode operator feels that no time can be estimated accurately enough in delayed mode, the time should be left as fill value.

The second method, based on float transmission strategy, relies on the raw Argos messages the DAC receives. This method is also an estimate, but it is currently being implemented in some manner at the DACs. There are a few different methods to make the estimate and thus, DACs may do it differently. This document is including the recommended method that improves on the one from TWR.

Other event times can be (roughly for some of them) estimated from TET or TST and float parameters and/or in situ data statistical results.

This is the case for:

- DST, DET and PET which are determined from TET,
- AET and AST which are determined from TST.

Again, if the TET is fill value in real time or delayed mode, then DST, DET and PET will be also fill value. This is determined by the DACs and delayed mode operator.

If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET-(N_CYCLE) variable and the JULD_ADJUSTED variables so users know it has been applied.

Argo program measurement codes (MC) for APEX APF8 floats in REAL TIME				
Code (timing)	APF8 Variable	Description	Units	JULD_STATUS

0	Float does not know when it is launched. If the launch time and location are available from the ship, enter that time and location If the launch time and location are not available, use fill value.	Launch time and location	Time, position	0: value is estimated from pre-deployment information found in the metafile Or 9: value is not immediately known, but believe it can be estimated later
100 (DST)	TET from previous cycle OR Fill Value	If TET is estimated in real time, use the TET from previous cycle. OR If TET is not estimated in real time, use FillValue	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
200 (DET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
250 (PST)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
During the drift phase, the APF8 makes drift measurements. Common codes are listed below. See 3.4.1.1 for CTD measurements during drift for APEX floats				
296	Average pressure Average temperature	Any averaged measurements made during drift	Pressure Temp	9: value is not immediately known, but believe it can be estimated later
297	Minimum pressure Minimum temperature	Minimum value taken during drift	Pressure Temp	9: value is not immediately known, but believe it can be estimated later
298	Maximum pressure Maximum temperature	Maximum value taken during drift	Pressure Temp	9: value is not immediately known, but believe it can be estimated later
End of drift measurements				
300 (PET)	Not available, so use Fill Value CTD performed at end of drift		Time P, T, S	9: value is not immediately known, but believe it can be estimated later
301	Average pressure during drift	Best estimate of drift depth. See section 2.4.2 for more details	Pressure	9: value is not immediately known, but believe it can be estimated later
400 (DDET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
500 (AST)	If PARK and PROFILE depths are equal and TET is estimated in real time: AST(i)=TET(i) – UP TIME OR FillValue	See 2.2.4.10	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
501	DownTimeEpoch/UNIX epoch when the down-time expired	Down-time end time – time out	Time	2: value is transmitted by the float
502	Time of profile initiation provided in auxiliary engineering data See 2.2.4.9 $AST_{FL} = DTET_{FL} + TPI$ minutes		Time	3: value is directly computed from relevant, transmitted float information

600 (AET)	Float does not know when it reaches the surface, so Fill Value		Time	9: value is not immediately known, but believe it can be estimated later
602	Time of MC=701 minus 10 minutes		Time	3: value is directly computed from relevant, transmitted float information
700 (TST)	See section 2.2.4.11 & 4	Based on Argos messages	Time	3: value is directly computed from relevant, transmitted float information
701 TST sent by APEX floats	TST_{FL} = $DTET_{FL}$ + TOTPI minutes	See 2.2.4.13	Time	3: value is directly computed from relevant, transmitted float information
702 (FMT)	Earliest time of all Argos messages received	Time	Time	4: value is determined by satellite
703 (ST)	All Argos times and locations		Time, Position	4: value is determined by satellite
704 (LMT)	Latest time of all Argos messages received		Time	4: value is determined by satellite
800 (TET)	2.2.4.1 and ANNEX B: Transmission End Time estimation for an APEX Argos float OR FillValue	DACs can choose to make this estimate in real time or not. Annex B explains how to make the estimate. 2.2.4.1 gives guidance how to implement the method in Annex B	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later

Argo program measurement codes (MC) for APEX APF8 floats in DELAYED MODE				
Code (timing)	**APF8 Variable**	**Description**	**Units**	**JULD_STATUS**
0	Float does not know when it is launched. If the launch time and location are available from the ship, enter that time and location If the launch time and location are not available, use fill value.	Launch time and location	Time, position	0: value is estimated from pre-deployment information found in the metafile Or 9: value is not immediately known, but believe it can be estimated later
100 (DST)	TET from previous cycle OR Fill Value	If TET is estimated in delayed mode, use the TET from previous cycle. OR If TET is not estimated in real time, use FillValue	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
200 (DET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
250 (PST)	PST estimated from 2.2.4.4 OR FillValue	See 2.2.4.4 for details	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
During the drift phase, the APF8 makes drift measurements. Common codes are listed below. See 3.4.1.1 for CTD measurements during drift for APEX floats				
296	Average pressure Average temperature	Any averaged measurements made	Pressure Temp	9: value is not immediately known, but believe it can be

		during drift		estimated later
297	Minimum pressure Minimum temperature	Minimum value taken during drift	Pressure Temp	9: value is not immediately known, but believe it can be estimated later
298	Maximum pressure Maximum temperature	Maximum value taken during drift	Pressure Temp	9: value is not immediately known, but believe it can be estimated later
End of drift measurements				
300 (PET)	PET estimated from 2.2.4.5 OR Fill Value CTD performed at end of drift	See 2.2.4.5 for details on how to estimate PET	Time P, T, S	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
301	Average pressure during drift	Best estimate of drift depth. See section 2.4.1.1for more details	Pressure	9: value is not immediately known, but believe it can be estimated later
400 (DDET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
500 (AST)	If PARK and PROFILE depths are equal and TET is estimated: AST(i)=TET(i) – UP TIME OR AST estimated from 2.2.4.10 OR FillValue	See 2.2.4.10 OR 5	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
501	DownTimeEpoch/UNIX epoch when the down-time expired	Down-time end time – time out	Time	2: value is transmitted by the float
502	Time of profile initiation provided in auxiliary engineering data. See 2.2.4.9 AST_{FL} = $DTET_{FL}$ + TPI minutes		Time	3: value is directly computed from relevant, transmitted float information
600 (AET)	Float does not know when it reaches the surface, so Fill Value		Time	9: value is not immediately known, but believe it can be estimated later
602	Time of MC=701 minus 10 minutes		Time	3: value is directly computed from relevant, transmitted float information
700 (TST)	See section 2.2.4.11 & 4	Based on Argos messages	Time	3: value is directly computed from relevant, transmitted float information
701 TST sent by APEX floats	TST_{FL} = $DTET_{FL}$ + TOTPI minutes	See 2.2.4.13	Time	3: value is directly computed from relevant, transmitted float information
702 (FMT)	Earliest time of all Argos messages received	Time	Time	4: value is determined by satellite
703 (ST)	All Argos times and locations		Time, Position	4: value is determined by satellite
704 (LMT)	Latest time of all Argos		Time	4: value is determined by

	messages received			satellite
800 (TET)	2.2.4.1 and ANNEX B: Transmission End Time estimation for an APEX Argos float OR FillValue	Delayed mode operators can choose to make this estimate in real time or not. Annex B explains how to make the estimate. 2.2.4.1 gives guidance how to implement the method in Annex B	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later

2.2.4.1 Transmission End Time determination – APEX APF8 floats

The TET can be estimated with the method proposed in §3.3. In this Annex B, two methods are proposed for finding TET depending on whether or not clock offset has been estimated. In order to ensure that these estimation methods are as robust as possible, the metadata information going into them (cycle time, whether the float is a Deep Profile First float, etc) must be correct. A list has been compiled by J.P. Rannou of corrected meta data for floats used in the ANDRO Atlas work. This file, found at ftp://ftp.ifremer.fr/ifremer/argo/etc/coriolis-custom/argo-andro-data/metadata_admt14/) (see http://www.argodatamgt.org/Documentation/Delayed-mode-trajectories-recovered-from-Andro-project for details), may be a good place to start to confirm the metadata for many APEX floats. **If the DAC and/or delayed mode operator wishes to estimate the TET, thefollowing steps should be taken:**

- Correct clock drift at launch for clocks which have not been correctly set,
- Estimate TET using the first algorithm (without clock drift estimation) for the first 32 cycles,
- From cycle #33, estimate the clock drift:
    - If it is less than 20 minutes per year, estimate TET using the second algorithm (with clock drift estimation) for all float cycles,
    - If it is greater than 20 minutes per year, there was an unexpected behavior of the float and the TET should not be estimated.

For more details, refer to Annex B.

Regardless of whether clock offset has been estimated during the TET determination, the resulting values should be stored in the JULD_ADJUSTED variable in the N_MEASUREMENT array with the measurement code set to 800 and STATUS set to 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behavior

N_CYCLE arrays: TET value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 1.

If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If the DAC and/or delayed mode operator does not choose to estimate the TET, then fill value should be inserted into the JULD variable in the N_MEASUREMENT array with the measurement code set to 800 and the STATUS set to 9.

N_CYCLE arrays: fill value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 9.

2.2.4.2 Descent Start Time determination – APEX APF8

DST = TET from previous cycle for all APEX floats.

2.2.4.3 Descent End Time determination – APEX APF8

APEX floats do not measure or estimate pressure on their descent, so the time when the float first approaches within 3% of the eventual drift pressure cannot be calculated. Nothing is entered into the JULD variable in the N_MEASUREMENT array because the float cannot measure or estimate the pressure on descent.

If, during delayed mode processing, it is determined that the float overshoots the drift pressure on descent, DET is the time of the overshoot. This time can be entered into the JULD variables in the N_MEASUREMENT array with an MC=200 and a STATUS equal to 2: value is transmitted by the float.

2.2.4.4 Park Start Time determination – APEX APF8 in Delayed Mode

For non APF9 APEX floats, the PST can be **roughly** estimated using the DST and the duration of the descent to PARKING depth. One way to estimate the PST is to use the mean descent rate as described in Annex I. **DACs and delayed mode operators can choose if they want to use this method to estimate the PST, but it should be done in delayed mode**.

If estimated in delayed mode, the PST value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 1 (estimated using procedures that rely on typical float behavior).

If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

In real time or if no estimate is made, fill value should be stored in the JULD variable in the N_MEASUREMENT array with the measurement code set to 250 and the STATUS set to 9.

N_CYCLE arrays: fill value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 9.

2.2.4.5 Park End Time determination – APEX APF8

For Argos APEX floats, PET can be computed from TET. This is not done in real time, but can be estimated later. See Annex I for more details.

2.2.4.6 Deep Descent End Time determination – APEX APF8

For Argos APEX floats, DDET could be estimated from PET and the mean descent velocity estimated for DET determination.

However:

- Deep descent velocity is not necessarily the same as the mean velocity between the surface and the PARKING depth,
- We have no in situ pressure measurements between PET and DDET from APEX floats,
- DDET is not as important as DET.

Consequently, DDET should not currently be required to be estimated for APEX Argos floats. However, because DDET might be estimated at a later date, fill value should go in the N_MEASUREMENT array with an MC = 400 and STATUS code of 9: value is not immediately known, but believe it can be estimated later

For the N_CYCLE array, JULD_DEEP_DESCENT_END = fill value as does JULD_DEEP_DESCENT_END_STATUS.

.

2.2.4.7 Ascent Start Time determination – APEX APF8

2.2.4.8 Argos APEX floats that do not provide this time

If the PARKING and PROFILE depths are equal for cycle #i, then:

AST(i) = TET(i) - UP TIME

If not, we can however **roughly** estimate AST using AET and the profile duration. See Annex J for more details.

If estimated, the AST value should also be stored in the JULD_ADJUSTED variables with an MC = 500 and STATUS set to 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour. Apply clock offset if it has been determined.

For the N_CYCLE array, the AST value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 1. If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If the AST value is not estimated, fill value should be stored in the JULD variable with an MC=500 and STATUS set to 9.

N_CYCLE arrays: fill value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 9.

2.2.4.9 Ascent Start Time provided by APEX floats

Some float directly provide the time at the end of DOWN TIME period ($DTET_{FL}$).

These float versions also provide, in the Auxiliary Engineering Data (AED), the "Time of profile initiation". This information is defined as the time difference, in minutes, between profile start and end of DOWN TIME (negative for start before expiration and positive for start after expiration, thus in this latter case, necessarily when TOD feature has been set).

The Auxiliary Engineering Data are not always transmitted (depending on the remaining space in the last Argos message) but if received, this "Time of profile initiation" (TPI) can be used to compute a second value of AST provided by the float (AST_{FL}).

$AST_{FL} = DTET_{FL}$ + TPI minutes

AST_{FL} value computed from $DTET_{FL}$ (corrected from clock offset) does not need to be corrected from clock offset but the information should be set in the AST_{FL} storage.

AST_{FL} is stored in the JULD_ADJUSTED N_MEASUREMENT arrays with the MC = 502 and the STATUS equal to 3: value is computed from information transmitted by the float. Clock offset has been applied in the $DTET_{FL}$ variable.

2.2.4.10 Ascent End Time determination – APEX APF8

For APEX APF8 floats, the AET is not fully determined and cannot be estimated with a simple formula.

Fill value should be stored in the JULD variables in the N_MEASUREMENT array with an MC = 600 and STATUS set to 9.For the N_CYCLE array, fill value should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END set to 9.

2.2.4.11 Transmission Start Time determination – APEX APF8

Some APEX floats provide the time at the end of the DOWN TIME period. For these float types, it is possible to compute a TST based on the DOWN TIME and to compute a TST based on Argos transmission or GTS fixes. In real time, it is difficult to know which method to compute TST is better because there may be problems with either the onboard clock (affecting the DOWN TIME) or the actual Argos transmission (affecting the alternate method to calculate TST). Therefore, it is best to compute TST using both methods in real time (going into MC=700 and MC=701). In delayed-mode, an expert can examine the TST values and determine which is best. The best value of TST should always go in MC=700 in delayed mode. This might mean copying over the value in MC=701 if that value is determined to be better in delayed mode. Both methods are outlined below.

2.2.4.12 Argos APEX floats

The TST can be computed with the method proposed in §4.2.

The TST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 700 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

For the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 3. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.4.13 Transmission Start Time provided by APEX Argos floats

Some float versions directly provide the time at the end of the DOWN TIME period ($DTET_{FL}$).

If the float clock offset has been estimated during the TET determination, $DTET_{FL}$ value should first be corrected for clock offset and the information should also be set in the $DTET_{FL}$ storage.

$DTET_{FL}$ is included in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 701 STATUS set to 2: value is transmitted by the float.

These float versions also provide the time, in minutes, of telemetry phase initiation relative to $DTET_{FL}$ (TOTPI).

Thus a TST, provided by the float, can be computed: $TST_{FL} = DTET_{FL}$ + TOTPI minutes

TST_{FL} value computed from $DTET_{FL}$ (corrected for clock offset) does not need to be corrected for clock offset but the information should be set in the TST_{FL} storage.

TST_{FL} is included in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 701 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

2.2.5 APEX floats with the APF9a or APF9t controller

APEX floats equipped with most APF9a or APF9t controller boards have begun to address the lack of transmitted cycle timing by transmitting timing information both in the mission prelude (cycle 0) and each cycle. Some of this timing information may be in the Auxiliary Engineering data. The four timing specifications that are transmitted during cycle 0 are:

- ParkDescentPeriod
- DeepProfileDescentPeriod
- DownIntervals
- UpIntervals

In addition, two time stamps are transmitted from each cycle:

- DownTimeEpoch
- TelemetryEpoch

These times should be used to calculate other cycle timing variables and stored in the TRAJ file with the appropriate *_STATUS flag to reflect that the timing information is transmitted or calculated from transmitted information. If none of this timing information is available in the Auxiliary Engineering data and the APF9 float is an early version, please refer back to the APF8 instructions.

2.2.5.1 Auxiliary Engineering Data (AED)

The Auxiliary Engineering Data (AED) includes the "Time of profile initiation". This information is defined as the time difference, in minutes, between profile start and end of DOWN TIME (negative for start before expiration and positive for start after expiration, thus in this latter case, necessarily when TOD feature has been set).

The AED are not always transmitted (depending on the remaining space in the last Argos message) but if received, this "Time of profile initiation" (TPI) can be used to compute the value of AST provided by the float (AST_{FL}).

AST =DownTimeEpoch(ToD for down-time expiration)minutes + TPI minutes

The AST value computed from DownTimeEpoch (corrected from clock offset) does not need to be corrected from clock offset but the information should be set in the AST storage.

AST is stored in the JULD_ADJUSTED N_MEASUREMENT arrays with the MC = 500 and the STATUS equal to 3: value is computed from information transmitted by the float. Clock offset has been applied in the DownTimeEpoch variable.

The descending pressure marks are also included in the Auxiliary Engineering Data and they are measured in bar. These pressure marks can be entered in the JULD (N_MEASUREMENT) variable with a measurement code of 189 or 190 and a STATUS flag of 2: value is transmitted by the float.

APEX APF9a and APF9t floats provide the time at the end of the DOWN TIME period (DownTimeEpoch or ToD for down-time expiration). For these float types, it is possible to compute a TST based on the DOWN TIME and to compute a TST based on Argos transmission (described in 6.2). In real time, it is difficult to know which method to compute TST is better because there may be problems with either the onboard clock (affecting the DOWN TIME) or the actual Argos transmission (affecting the alternate method to calculate TST). Therefore, it is best to compute TST using both methods in real time (going into MC=700 and MC=701). In delayed-mode, an expert can examine the TST values and determine which is best. The best value of TST should always go in MC=700 in

delayed mode. This might mean copying over the value in MC=701 if that value is determined to be better in delayed mode.

Argo program measurement codes (MC) for APEX APF9a or APF9t				
Code (timing)	**APF9a or APF9t Variable**	**Description**	**Units**	**JULD_STATUS**
0	Float does not know when it is launched. If the launch time and location are available from the ship, enter that time and location If the launch time and location are not available, use fill value.	Launch time and location	Time, position	0: value is estimated from pre-deployment information found in the metafile Or 9: value is not immediately known, but believe it can be estimated later
100 (DST)	DownTimeEpoch - DownIntervals	Down TimeExpiredEpoch from current message cycle minus Down Time interval from cycle 0 message. Notice one time is in days another in hours or minutes.	Time DownTimeEpoch from N Down Time interval from Cy 0	3: value is directly computed from relevant, transmitted float information
If an APEX isopycnal float				
189	Descent() Pressure: XX.X Found in Auxiliary Engineering data, so sometimes not available every cycle	Descending CTD measurements starting at a programmed time after DST (often six hours) and following every 60 minutes	Time Pressure (bars)	3: value is directly computed from relevant, transmitted float information OR 9: value is not immediately known, but believe it can be estimated later
Endif an APEX isopycnal float				
190	Descent() Pressure: XX.X Found in Auxiliary Engineering data, so sometimes not available every cycle	Descending CTD measurements starting at a programmed time after DST (often six hours) and following every 60 minutes. See section 2.4.3.2 for more details	Time Pressure (bars)	3: value is directly computed from relevant, transmitted float information OR 9: value is not immediately known, but believe it can be estimated later
200 (DET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
250 (PST)	DST + ParkDescentPeriod	Descent Start Time from above plus Park Descent Period from Cycle 0. Notice one time is in days another in hours. Note that this is a time-out value and does not indicate when float actually stabilizes at drift pressure.	Time DST from N Park Descent Period from Cy 0	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour
During the drift phase, the APF9a and APF9t floats measure time pressure and temperature hourly, but these are not reported. Instead, a statistical pack of information is sent and the following measurement codes may apply. Please check all the relative MCs for transitioning towards MC300 to see which are appropriate				
296	Average pressure Average temperature	Any averaged measurements made during drift	Pressure Temp	9: value is not immediately known

286	Any supporting measurements for average measurements. An example is pressure at average temperature		Pressure Temp	9: value is not immediately known
295	Median pressure Median temperature	Median of measurements made during drift	Pressure Temp	9: value is not immediately known
285	Any supporting measurements for the median measurements. An example is pressure at median temperature		Pressure Temp	9: value is not immediately known
294	Standard deviation of measurements taken during drift		Pressure Temp	9: value is not immediately known
297	Minimum pressure Minimum temperature	Minimum value taken during drift	Pressure Temp	9: value is not immediately known
298	Maximum pressure Maximum temperature	Maximum value taken during drift	Pressure Temp	9: value is not immediately known
287	Pressure at minimum temperature		Pressure	9: value is not immediately known
288	Pressure at maximum temperature		Pressure	9: value is not immediately known
End of drift measurements				
300 (PET)	DOWN TIME EXPIRED EPOCH (GMT) minus DEEP PROFILE DESCENT PERIOD (HOURS)	Down TimeExpiredEpoch from current message cycle minus Deep Profile Descent Period from cycle 0 message. Notice one time is in days another in hours.	Time DST from N Deep Profile Descent Period from Cy 0	3: value is directly computed from relevant, transmitted float information
301	Average pressure during drift	Best estimate of drift depth	Pressure	9: value is not immediately known, but believe it can be estimated later
400 (DDET)	Same as AST		Time	2: value is transmitted by the float
500 (AST)	DOWN TIME EXPIRED EPOCH (GMT) plus TIME INITIATED TO EPOCH (MINUTES) Sent in Auxiliary engineering data which is not always available. See section 2.2.5.1	Time that float actually starts ascending; Can be the same as the DownTimeEpoch if the float times out before reaching profile pressure. Otherwise, float begins to ascend as soon as profile pressure is reached	Time	2: value is transmitted by the float
501	DownTimeEpoch/UNIX epoch when the down-time expired/ToD for downtime expiration	Down-time end time – time out	Time (minutes)	2: value is transmitted by the float
600 (AET)	700 minus 10 minutes	Assume that float finishes ascent ten minutes before transmission start time	Time N	9: value is not immediately known, but believe it can be estimated later
602	701 – 10 minutes		Time	3: value is directly computed from relevant, transmitted float information

700 (TST)	See section 4	Based on Argos messages	Time	3: value is directly computed from relevant, transmitted float information
701 TST sent by APEX floats	DOWN TIME EXPIRED EPOCH (GMT) Plus START OF TRANSMISSION FROM EPOCH	See 2.2.4.13	Time	3: value is directly computed from relevant, transmitted float information
702 (FMT)	Earliest of all Argos messages received: +99.999 +999.999 2012/12/04 7:34:55 000 A 0	FillValue for LATITUDE FillValue for LONGITUDE JULD (First time from Argos) Unknown SATELLITE_NAME FillValue for POSITION_ACCURACY	Degrees Degrees Time Unknown N/A N/A N	4: value is determined by satellite
703 (ST)	All Argos times and locations. -32.440 -141.872 2012/12/04 7:39:39 015 A 2 -32.443 -141.881 2012/12/04 8:31:22 011 L 1 -32.439 -141.889 2012/12/04 9:18:39 010 A 1 -32.448 -141.874 2012/12/04 9:33:48 015 P 2 -32.446 -141.871 2012/12/04 10:09:44 007 N 1 -32.449 -141.855 2012/12/04 11:18:52 013 P 2 -32.448 -141.850 2012/12/04 11:49:51 009 N 2 -32.445 -141.830 2012/12/04 13:33:44 005 N 1 -32.441 -141.835 2012/12/04 13:34:57 006 K 1 -32.434 -141.807 2012/12/04 15:16:00 005 K 1 -32.423 -141.796 2012/12/04 16:19:01 007 M 3 -32.403 -141.789 2012/12/04 18:13:44 007 L 2 -32.402 -141.788 2012/12/04 18:55:13 011 A 3	LATITUDE LONGITUDE JULD Unknown SATELLITE_NAME POSITION_ACCURACY	Degrees Degrees Time Unknown N/A N/A N	4: value is determined by satellite
704 (LMT)	Latest time of all Argos messages received +99.999 +999.999 2012/12/04 18:59:50 000 A 0	FillValue for LATITUDE FillValue for LONGITUDE JULD (Last time from Argos) Unknown	Degrees Degrees Time Unknown N/A N/A N	4: value is determined by satellite

		SATELLITE_NAME FillValue for POSITION_ACCURACY		
800 (TET)	DOWN TIME EXPIRED EPOCH (GMT) Plus UP TIME INTERVALS (HOURS)		Time	3: value is directly computed from relevant, transmitted float information

2.2.6 APEX floats with the APF9i controller and Seabird NAVIS floats

2.2.6.1 Transmitted time stamp information from msg files

Seabird Navis and some APEX floats equipped with the firmware revisions 072314 or later, telemeter 6 time stamps in the msg file from each cycle and these can be slotted directly into 6 Argo cycle timing variables.

For each cycle, the 6 transmitted time stamps are:

1. **TimeStartDescent (DST)**: This timestamp marks the initiation of buoyancy reduction with the intent to descend from the surface. This means the start of the piston retraction at the beginning of the profile cycle. The float cannot detect when it actually descends below the surface. This closely matches DST, but can be wrong in some pathological cases like when a float never descends. If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2. **TimeStartPark (PST)**: This timestamp marks the phase transition from the park-descent phase to the park phase. During the park-descent phase, there is no attempt to autoballast the float – it is simply in free fall for a user-specified period of time. Therefore, there is not necessarily any connection to when the float approaches neutral buoyonacy. So it matches up with PST, but not DET.

3. **TimeStartProfileDescent (PET)**: This timestamp is defined only if the Park-n-Profile (PnP) feature is enabled. This timestamp marks the initiation of buoyancy reduction with the intent to descend from the park pressure to the profile pressure. This means the start of the piston retraction at the beginning of the profile-descent phase. The float cannot detect when it actually begins to descend. If PnP is enabled, this timestamp matches PET well.

4. **TimeStartProfile (AST)**: This timestamp marks the transition to the profile phase of the profile cycle. This means the start of the initial piston extension by a user-specified amount. The float cannot detect when it actually begins to ascend. In some pathological cases, the float may continue to descend until the ascent-control algorithm's feedback mechanism compensates for any lingering negative buoyancy. This timestamp comes closest to matching AST, but there are differences.

5. **TimeStopProfile (AET)**: This timestamp marks the termination of the profile phase. The phase transition is induced by one of two conditions: either the surface detection algorithm has returned true or else the ascent timeout period has expired. Typically this phase transition happens a few meters below the sea surface but this is not guaranteed. In unusual cases the

phase transition can happen after surfacing or far below the surface. This timestamp matches AET, but it is noted that it has requirement for being at the surface.

6. **TimeStartTelemetry**: This timestamp marks the initiation of the telemetry phase of the profile cycle and it can occur while the float is still subsurface. The telemetry phase consists of well-defined telemetry cycles that are spaced apart by a user-defined period. Each cycle starts with acquisition of a GPS fix and then an attempt to upload data. Cycles continue until all data is successfully uploaded or until the telemetry phase times-out. This most closely matchs TST, but is assigned MC = 701 since it is possible to determine the TST based on the time from the satellite messages.

These transmitted information should be used to calculate other cycle timing variables (if the data is not available in the log files) and stored in the TRAJ file with the appropriate *_STATUS flag to reflect that the timing information is transmitted or calculated from transmitted information (see following section for details and table below).

2.2.6.1 Descent Start Time

DescentInit() from log file or TimeStartDescent from msg file (if available)

If a time stamp is not available, DST can be calculated as described in the table below.

2.2.6.2 Descent End Time (DET)

From the log file, 'Descent()' time and pressure stamps.

Use the first pressure value within 3% of the configured drift pressure, or if a float overshoots the configured pressure, use the first depth/time of the overshoot. In this case, the DET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 200 and STATUS is then set to 2: Value is transmitted by the float.

Alternatively, DET can be interpolated from a line fit to the curve of Descent() pressure vs time. Calculate time when pressure is within 3% of drift pressure. In this case, the DET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 200 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_END variable and the JULD_DESCENT_END_STATUS set to 3 (computed from information transmitted directly by the float). If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If the log file is unavailable, the DET = PST. Or, calculate DET from the .msg file using the values output with ParkDescentP. ParkDescentP[1] is always made at DST+1796 seconds. ParkDescentP[n] measurements are then made every 1800 seconds. The output is pressure rounded to the nearest bar.

The most accurate way to calculate fall rate is from the change in pressure and the change in net seconds from the Descent() lines in the .log file. But it can be estimated from ParkDescentP[n] in the .log file. Pressure samples 1 to n are made 1800 seconds (30 minutes) apart. The value output by ParkDescentP is pressure rounded to the nearest bar. The time between ParkDescentP[0] and ParkDescentP[1] should not be used as the timing between the two samples is variable.

2.2.6.3 Park End Time (PET)

GoDeepInit() Or TimeStartProfileDescent

The PET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 300 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.6.4 Ascent Start Time (AST) and Deep Descent End Time (DDET)

DDET is the same as AST (see next section). The DDET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 400 and STATUS set to 2 or 3 depending on how AST is calculated.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2 or 3 depending on how AST is calculated. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.6.5 Ascent Start Time

ProfileInit() Sample 0 initiated at XXXXX from the log file. Or TimeStartProfile from the msg file. AST can be calculated if neither is available (see table below).

The AST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 500 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2: value transmitted by float. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

When the time is estimated, the AST value should also be stored in the JULD_ADJUSTED variables in the N_MEASUREMENT arrays with an MC = 500 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

For the N_CYCLE array, the AST value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 1. If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.6.6 Ascent End Time (AET)

SurfaceDetect() in the log file or TimeStopProfile in the msg file (if available).

The AET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 600 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2: value transmitted by float. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.6.7 Transmission Start Time (TST)

CLogin() or login() from the log file, or TimeStartTelemetry from the msg file (if available)

The TST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 700 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2: value is transmitted by float. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.6.8 Transmission End Time (TET)

This is the last time that the Telemetry() command is recorded

The TET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 800 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 2: value is transmitted by float. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
0 (launch)	Best option is time, lat, long taken from metadata records on ship launch date/time. If not available, from *.000.msg file (this may be hours before float actually launched): # GPS fix obtained in 38 seconds. # lon lat mm/dd/yyyy hhmmss nsat Fix: 119.906 -60.007 12/15/2015 043710 10	Launch time and location Occurs only once in a trajectory file.	Time, position n = 0	0 if taken from ship metdata 2 if taken from 000.msg file
100 DST transmitted (first choice)	Best option from APEX and NAVIS log files: (Jan 14 2016 04:45:15, 7 sec) DescentInit() Deep profile 5 initiated at mission-time 853679sec. Alternative option from NAVIS msg file: TimeStartDescent=1478131066 Nov 02 2016 23:57:46 *In some older Navis floats, the TimeStartDescent was not available and cannot be used.	Descent start time	Time n	2. value is transmitted by the float
100 DST calculated (800 TET calculated)	if PST (code 250) is not empty: DST = PST – ParkDescentTime/24/60 Obtain ParkDescentTime from msg file: $ ParkDescentTime(300) [min] Can be applied to TET for n-1 profile.	Descent start time	Time n	3. value is directly computed from relevant, transmitted float information
190 DSP	Taken preferably from log file: (Feb 10 2018 05:03:38, 724 sec) Descent() Pressure: 49.8	Descending CTD measurements	Time, Pressure n	2. value is transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	(Feb 10 2018 05:51:38, 3604 sec) Descent() Pressure: 408.8 (Feb 10 2018 06:51:38, 7204 sec) Descent() Pressure: 731.5 (Feb 10 2018 07:51:38, 10804 sec) Descent() Pressure: 938.4 (Feb 10 2018 08:51:38, 14404 sec) Descent() Pressure: 993.9 (Feb 10 2018 09:51:38, 18004 sec) Descent() Pressure: 993.9 A non-time stamped, low-resolution version (need to multiply by 10) is available from the msg file: ParkDescentP[0]=5 ParkDescentP[1]=41 ParkDescentP[2]=73 ParkDescentP[3]=94 ParkDescentP[4]=99 ParkDescentP[5]=99			
189	Optional buoyancy adjustment times when float begins descent: (Nov 22 2008 02:25:22, 0 sec) DescentInit() Deep profile 3 initiated at mission-time 878770sec. (Nov 22 2008 02:25:24, 2 sec) DescentInit() Surface pressure: 0.2dbars. (Nov 22 2008 02:25:29, 7 sec) PistonMoveAbsWTO() 211->070	Active adjustments to buoyancy on descent	Time, n	2. value is transmitted by the float
During the drift phase				
200 DET transmitted (best choice)	From the log file: For a float that overshoots: (Feb 06 2018 05:23:02, 620 sec) Descent() Pressure: 8.1 (Feb 06 2018 05:42:45, 1804 sec) Descent() Pressure: 168.8 (Feb 06 2018 06:12:45, 3604 sec) Descent() Pressure: 413.4 (Feb 06 2018 06:42:45, 5404 sec) Descent() Pressure: 632.8 (Feb 06 2018 07:12:44, 7204 sec) Descent() Pressure: 837.5 (Feb 06 2018 07:42:44, 9004 sec) Descent() Pressure: 997.9 (Feb 06 2018 08:12:44, 10804 sec) Descent() Pressure: 1120.7 DET, time of overshoot (Feb 06 2018 08:42:43, 12604 sec) Descent() Pressure: 1227.7 (Feb 06 2018 09:12:43, 14404 sec) Descent() Pressure: 1271.7 (Feb 06 2018 09:42:42, 16204 sec) Descent() Pressure: 1301.6 For a non-overshoot float: (Feb 10 2018 05:03:38, 724 sec) Descent() Pressure: 49.8 (Feb 10 2018 05:51:38, 3604 sec) Descent() Pressure: 408.8 (Feb 10 2018 06:51:38, 7204 sec) Descent()	Descent end time. Time when float first approaches within 3% of the CONFIGURED drift pressure. Float may be transitioning from the surface or from a deep profile. This variable is based on pressure only and can be measured only and can be estimated by fall-rate (see below). In the case of a float that overshoots the drift pressure on descent, DET is the time of the overshoot.	Time n	2. Value is transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	Pressure: 731.5 (Feb 10 2018 07:51:38, 10804 sec) Descent() Pressure: 938.4 (Feb 10 2018 08:51:38, 14404 sec) Descent() Pressure: 993.9 DET, within 3% of mission park (1000db) (Feb 10 2018 09:51:38, 18004 sec) Descent() Pressure: 993.9 (Feb 10 2018 09:51:38, 18004 sec) ParkInit() For NAVIS floats and newer APEX floats, CONFIGURED drift pressure is found in the .msg file at the start $ ParkPressure(1000) [dbar].			
200 DET estimated (use if transmitted DET is not available)	DET = PST or you can calculate DET based on the fall rate using times associated with descending park measurements. For Navis (and Apex?) floats, you can calculate DET from the .msg file using the values output with ParkDescentP. ParkDescentP[1] is always made at DST+1796 seconds. ParkDescentP[n] measurements are then made every 1800 seconds. The output is pressure rounded to the nearest bar. The most accurate way to calculate fall rate is from the change in pressure and the change in net seconds from the Descent() lines in the .log file. But it can be estimated from ParkDescentP[n] in the .log file. Pressure samples 1 to n are made 1800 seconds (30 minutes) apart. The value output by ParkDescentP is pressure rounded to the nearest bar. The time between ParkDescentP[0] and ParkDescentP[1] should not be used as the timing between the two samples is variable.	See above Fall rate is not output, only can be calculated from Descent() in .log files*	Time n	3. value is directly computed from relevant, transmitted float information
250 PST	From the log file: (Feb 06 2018 09:42:44, 16206 sec) ParkInit() In some floats, eg Navis, the msg file may contain: TimeStartPark=1517910164 Feb 06 2018 09:42:44 If either of the above is unavailable, use first park point from msg file: ParkPts: Feb 06 2018 09:43:08 1517910188 16230 1301.58 5.0065 34.4284	Time of park start Time when float transitions to its Park or Drift mission. This variable is based on float logic	Time n	2. Value is transmitted by the float
290 PTM	If float is programmed to take one sample, 299, otherwise use 290. Usually Navis floats only record one ParkPts during profile *.001 From the msg file: ParkPts: Feb 10 2018 09:52:10 1518256330 18036 993.93 4.8352 34.3890 ParkPts: Feb 10 2018 10:52:07 1518259927 21633 997.28 4.7839 34.3902 ParkPts: Feb 10 2018 11:52:07 1518263527	A series of pressure measurements taken daily during drift.	Time, pressure, temp, PSAL, etc. n	

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	25233 998.61 4.7770 34.3908 ParkPts: Feb 10 2018 12:52:07 1518267127 28833 993.30 4.7229 34.3888 ParkPts: Feb 10 2018 13:52:07 1518270727 32433 994.89 4.6988 34.3971 ParkPts: Feb 10 2018 14:52:07 1518274327 36033 997.07 4.6915 34.3956 ParkPts: Feb 10 2018 15:52:07 1518277927 39633 1000.34 4.7448 34.3929 ParkPts: Feb 10 2018 16:52:07 1518281527 43233 996.92 4.7109 34.3895 ParkPts: Feb 10 2018 17:52:07 1518285127 46833 991.47 4.6570 34.3980 ParkPts: Feb 10 2018 18:52:07 1518288727 50433 995.81 4.6700 34.3968 ParkPts: Feb 10 2018 19:52:07 1518292327 54033 997.65 4.6712 34.3970 ParkPts: Feb 10 2018 20:52:07 1518295927 57633 1005.26 4.8114 34.3890 ParkPts: Feb 10 2018 21:52:07 1518299527 61233 998.14 4.7660 34.3889 ParkPts: Feb 10 2018 22:52:07 1518303127 64833 991.14 4.6677 34.3995 ParkPts: Feb 10 2018 23:52:07 1518306727 68433 1000.03 4.7145 34.3938 ParkPts: Feb 11 2018 00:52:07 1518310327 72033 1004.18 4.7084 34.3950 ParkPts: Feb 11 2018 01:52:07 1518313927 75633 995.84 4.7796 34.3916 ParkPts: Feb 11 2018 02:52:07 1518317527 79233 1000.93 4.8089 34.3897 ParkPts: Feb 11 2018 03:52:07 1518321127 82833 1002.78 4.7672 34.3917 .... ParkPts: Feb 19 2018 07:52:10 1519026730 788436 998.07 4.1888 34.4374 ParkPts: Feb 19 2018 08:52:10 1519030330 792036 1007.68 4.2876 34.4312 ParkPts: Feb 19 2018 09:52:10 1519033930 795636 1010.83 4.3498 34.4274 ParkPts: Feb 19 2018 10:52:10 1519037530 799236 1008.37 4.3494 34.4277 ParkPts: Feb 19 2018 11:52:10 1519041130 802836 1005.17 4.2945 34.4317 $ Profile 9640.087 terminated: Mon Feb 19 23:53:36 2018			
299	From the log file: (Feb 19 2018 12:52:00, 806426 sec) ParkTerminate() Piston Position:74 Vacuum:79 Vq:196 Aq:5 Vsbe:181 Asbe:9 (Feb 19 2018 12:52:28, 806454 sec) ParkTerminate() PTS: 999.3dbars 4.2391C 34.4367PSU From the msg file, no date/time stamp available: $ Profile 9640.087 terminated: Mon Feb 19 23:53:36 2018 $ Discrete samples: 2 $ p t s 999.30 4.2391 34.4367 (Park Sample)	Any measurement recorded during transition toward PET.	Time, pressure, temperature, salinity n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	2000.29 2.2072 34.6579 # Feb 19 2018 23:59:06 Sbe41cpSerNo[6437] NSample[21714] NBin[998] * For Navis floats, the discrete sample labelled (Park Sample) is always taken at the termination of the parking phase. If there is a second discrete sample, it is taken at the start of profiling, but it is taken after ascent start and after 500 (AST) is output in the .log file as ProfileInit() or in the .msg file as TimeStartProfile = ***. This second discrete sample should go in as 503 since continuous profiling will start later.*			
300 PET transmitted (best choice)	From the log file: (Feb 19 2018 12:52:28, 806454 sec) ParkTerminate() PTS: 999.3dbars 4.2391C 34.4367PSU (Feb 19 2018 12:52:28, 806454 sec) GoDeepInit() Moving piston. If the GoDeepInit line is not there, use the ParkTerminate line. For NAVIS floats: from the msg file, there may be: TimeStartProfileDescent=1518719438 Feb 15 2018 18:30:38 If neither is available, use the last park time from msg file.	Time when float exits from its Park or Drift mission. It may next rise to the surface (AST) or sink to profile depth (DDET)	Time n	2: value transmitted by the float
300 PET calculated (second choice)	If PET is not available from methods above, it can be estimated which relies on DST being known. PET = DST + DownTime/60/24 - DeepProfileDescentTime/60/24; From msg file, get DownTime and DeepProfileDescentTime: $ DownTime(14030) [min] $ DeepProfileDescentTime(270) [min]	Time when float exits from its Park or Drift mission. It may next rise to the surface (AST) or sink to profile depth (DDET)	Time n	3. value is directly computed from relevant, transmitted float information
301	Average of hourly pressure measurements from MC 290.	Representative park pressure	Pressure n	3: value is directly computed from relevant, transmitted float information
End of drift measurements				
400 DDET	Same as AST (code 500)	Time when float first approaches within 3% of the eventual deep profile pressure. This variable is based on pressure only and can be	Time n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
		measured or estimated by fall-rate.		
500 AST transmitted (best choice)	From the log file: ProfileInit() PrfId:098 Pressure:1964.6dbar pTable[1]:1950dbar For Navis floats: from the msg file, there may be: TimeStartProfile=1518727187 Feb 15 2018 20:39:47 * For Navis floats, the time in TimeStartProfile in the .msg file corresponds to the time the GoDeep() command is executed because it actually executed the ProfileInit() command*	Ascent start time	Time n	2: value transmitted by the float
500 AST calculated (second choice)	If the above values are not available, AST can be estimated. However, relies on DST being known. * Analysis of Navis data found that when you do calculate AST, the mean difference is about 12 hours. The difference changes throughout the course of the deployment as the ascent and descent rates are optimized.* AST = DST + DownTime[min]/60/24 From the msg file: $ DownTime(13740) [min] If the calculated AST using value above is earlier than PET (eg, in the case the float hits the bottom), then use PET + DeepProfileDescentTime as the AST estimate. From the msg file: $ DeepProfileDescentTime(300) [min] if AST < PET or AST > AET AST = PET + DeepProfileDescentTime/60/24; if AST < PET or AST > AET AST = NaN; end end If TimeOfDay setting is enabled in mission (BGC floats): *Yes, this is an option available in Navis BGCi floats. If it is in the firmware you will see the following line in the n.msg file. In this case TimeOfDay is disabled.* $ TimeOfDay(DISABLED) [min] DownTime will be affected if TimeOfDay value is enabled and set. To enable a float to surface at a particular time	Ascent start time	Time n	3: value computed from relevant, transmitted float information

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	of day, DownTime should be set to one day less than the entire cycle length, then the TimeOfDay value added to the DownTime Value; Eg: For a 10-day cycle with the float scheduled to surface around midday GMT: DownTime = 12960 min (9 days), and TimeOfDay = 100 min (or number of minutes after midnight local time to ensure arrival at the surface at time required). If time of profile start (AST) has to be calcuated use either: AST = DST + DownTime/60/64 + TimeOfDay/60/64;			
503	For both APEX and Navis floats, where there are 2 discrete samples, the second one is Sample 0 at profile initiation time and should be recorded in MC 503 as it occurs after the AST. From the msg file, no date/time stamp available: $ Profile 9640.087 terminated: Mon Feb 19 23:53:36 2018 $ Discrete samples: 2 $ p t s 999.30 4.2391 34.4367 (Park Sample) 2000.29 2.2072 34.6579 # Feb 19 2018 23:59:06 Sbe41cpSerNo[6437] NSample[21714] NBin[998] From the log file: (Feb 19 2018 17:46:38, 824104 sec) GoDeep() Sequence point detected at 2000.3dbar. (Feb 19 2018 17:46:41, 824107 sec) ProfileInit() PrfId:087 Pressure:2000.3dbar pTable[0]:2000dbar (Feb 19 2018 17:47:19, 824145 sec) Profile() Sample 0 initiated at 2000.3dbars for bin 0 [2000dbars]. PTS: 2000.3dbars 2.2072C 34.6579PSU	Deepest bin reached during ascending profile	Time, Pressure, Temp, Salinity n	2: value transmitted by the float
589	Optional. Buoyancy adjustment times on the Ascent. (Feb 19 2018 17:46:41, 824107 sec) ProfileInit() PrfId:087 Pressure:2000.3dbar pTable[0]:2000dbar (Feb 19 2018 17:47:19, 824145 sec) Profile() Sample 0 initiated at 2000.3dbars for bin 0 [2000dbars]. PTS: 2000.3dbars 2.2072C 34.6579PSU	Active adjustments to buoyancy on ascent	Time n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	(Feb 19 2018 17:47:19, 824145 sec) PistonMoveAbsWTO() 020->042 021 022 023 024 [30sec, 12.5Volts, 0.830Amps, CPT:1013sec] (Feb 19 2018 17:47:59, 824185 sec) PistonMoveAbsWTO() 024->042 025 026 027 028 [30sec, 12.6Volts, 0.822Amps, CPT:1043sec] (Feb 19 2018 17:50:07, 824313 sec) Sbe41cpStartCP() Continuous profile started. (Feb 19 2018 17:50:07, 824313 sec) PistonMoveAbsWTO() 029->042 030 031 032 033 [30sec, 12.6Volts, 0.866Amps, CPT:1073sec] (Feb 19 2018 17:50:49, 824355 sec) PistonMoveAbsWTO() 033->042 034 035 036 037 [30sec, 12.6Volts, 0.842Amps, CPT:1103sec] (Feb 19 2018 17:51:31, 824397 sec) PistonMoveAbsWTO() 037->042 038 039 040 041 [30sec, 12.6Volts, 0.846Amps, CPT:1133sec] (Feb 19 2018 17:52:13, 824439 sec) PistonMoveAbsWTO() 041->042 042 [2sec, 12.6Volts, 0.846Amps, CPT:1135sec] (Feb 19 2018 18:38:15, 827201 sec) AscentControlAgent() Bouyancy nudge to 52 (v=0.078dbar/sec).			
590	For some Navis bio floats, the discrete sample information is included in the log files. (Jul 10 2016 20:34:13, 13162 sec) Profile() Sample 0 initiated at 1601.5dbars for bin 8 [1600dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1597.1dbars,3.0014C,34.5900PSU / 26.211u,1.091539V / 63,466,213 / 130827,132055,131162,131012 / 130820,131443,131096,131516 / 16959,16959,16959 / 15547 -0.02200 (Jul 10 2016 20:43:24, 13713 sec) Profile() Sample 2 initiated at 1501.5dbars for bin 10 [1500dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1497.3dbars,3.2007C,34.5710PSU / 26.072u,1.085170V / 66,444,207 / 130835,132049,131168,130998 / 130812,131448,131090,131518 / 16959,16959,16959 / 15547 -0.02200 (Jul 10 2016 20:53:07, 14296 sec) Profile() Sample 4 initiated at 1401.3dbars for bin 12 [1400dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1397.4dbars,3.4660C,34.5450PSU / 25.717u,1.076850V / 61,473,203 / 130838,132037,131166,131000 / 130825,131444,131110,131520 / 16959,16959,16959 / 15546 -0.02100 (Jul 10 2016 21:03:23, 14912 sec) Profile() Sample 6 initiated at 1301.4dbars for bin 14 [1300dbars].	Series of measurements transitioning towards AET	Time, Pressure, Temp, Salinity n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1297.6dbars,3.7270C,34.5230PSU / 25.302u,1.068646V / 63,448,200 / 130831,132047,131152,131003 / 130806,131441,131106,131520 / 16959,16959,16959 / 15547 -0.02200 (Jul 10 2016 21:14:24, 15573 sec) Profile() Sample 8 initiated at 1200.5dbars for bin 16 [1200dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1197.1dbars,4.0622C,34.5010PSU / 24.853u,1.058171V / 62,451,194 / 130836,132036,131163,130999 / 130812,131444,131105,131514 / 16959,16959,16959 / 15548 -0.02200 (Jul 10 2016 21:26:08, 16277 sec) Profile() Sample 10 initiated at 1101.1dbars for bin 18 [1100dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 1098.0dbars,4.4999C,34.4720PSU / 24.226u,1.044719V / 64,444,189 / 130844, 132039,131158,131003 / 130808,131435,131106,131517 / 16959,16959,16959 / 15546 -0.02100 (Jul 10 2016 21:39:23, 17072 sec) Profile() Sample 12 initiated at 1000.7dbars for bin 20 [1000dbars]. PTS/O2Ph,O2T/FlBbCd/OcrI/OcrR/EcoFlBbCd/Crv: 997.8dbars,5.1208C,34.4510PSU / 23.597u,1.025704V / 58,448,179 / 130844,132038,131152,130992 / 130801,131438,131112,131505 / 16959,16959,16959 / 15548 -0.02200			
600 AET	From log file: (Feb 06 2018 03:42:10, 856511 sec) SurfaceDetect() SurfacePressure:-0.0dbars Pressure:4.0dbars BuoyancyPosition:556 (Feb 06 2018 03:42:10, 856512 sec) Sbe41cpStopCP() Continuous profile stopped. * For Navis floats, SurfaceDetect() triggers the ProfileTerminate() command, which then writes TimeStopProfile then $Profile lines to the .msg file. This is when the CTD turns off. From msg file: $ Profile 0800.021 terminated: Fri Feb 16 03:12:43 2018 From msg file, some floats (eg Navis), may have: TimeStopProfile=1518750763 Feb 16 2018 03:12:43 *This is written by the same command. See comment above	Time float switches from ascent mode to surface mode	Time n	2: value transmitted by the float
Float is on the surface				
Notes on how float behaves	From log file, (matches with the TimeStartTelemetry indicated in Navis msg files):			

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
when it reaches the surface	(Feb 06 2018 03:48:41, 856902 sec) TelemetryInit() Profile 20. (Npf ARGO FwRev: 170210) ** TelelmetryInit() starts a sequence where the bladder is inflated to get the antenna mast above the surface, then it tries to get a GPS position, the first GPS position is defined by gga, then it starts trying to connect to the modem with CLogin, and doesn't make the connection until the login() command is executed with the Login successful message. *			
703 ST	From the log file (fixes are from the PREVIOUS cycle, and should be added to the n-1 cycle): (Jan 14 2016 03:15:47, 848318 sec) GpsServices() Profile 4 GPS fix obtained in 26 seconds. (Jan 14 2016 03:15:47, 848318 sec) GpsServices() lon lat mm/dd/yyyy hhmmss nsat (Jan 14 2016 03:15:48, 848318 sec) GpsServices() Fix: 118.333 -59.042 01/14/2016 031520 10 From the msg file (fixes relate to the CURRENT cycle, add to n cycle) # GPS fix obtained in 167 seconds. # lon lat mm/dd/yyyy hhmmss nsat Fix: 118.333 -59.042 01/14/2016 031520 10	Satellite times and locations. One for each fix, in chronological order.	Time, Position n-1 if from the log file, n if from the msg file.	4. value is determined by satellite
700 TST	When the float first connects to the modem: ((Dec 17 2016 21:48:18, 47230 sec) login() Login successful. * For Navis and APEX floats, the login() command with the following text is the true time telemetry starts* If this time is not available, can use: - Time of last GPS fix (Apr 12 2018 09:16:35, 11053 sec) GpsServices() Fix: 154.3182 -33.8677 04/12/2018 091700 9 - Time of TelemetryInit() (Apr 12 2018 09:15:16, 10975 sec) TelemetryInit() Profile 0. (Npf ARGO FwRev: 170425) - TimeStartTelemetry (some NAVIS msg files) TimeStartTelemetry=1523524515 Apr 12 2018 09:15:15	Time of change of float phase to telemetry.	Time n-1 if from the log file, n if from the msg file.	2. Value transmitted by the float
702 FMT, and 704 LMT	Not necessary for Iridium floats	Time of first/last iridium message		
800 TET transmitted (best choice)	For consistency, it might be best to use the last time that the Telemetry() command is executed. The TelemetryTerminate() is also executed during ice evasion. This could lead to confusing results since the float never surfaces when evading ice.	Time of the end of transmission for the float.	Time, Position n-1	2. Value transmitted by the float

Argo trajectory file measurement codes (MC) for APF9i and Seabird NAVIS floats				
Code (timing)	**Name in float data output**	**Description**	**Units and data profile number**	**JULD_STATUS**
	(Feb 06 2018 05:12:40, 861943 sec) Telemetry() Telemetry cycle complete: PrfId=20 ConnectionAttempts=4 Connections=4			
800 TET estimated (second choice)	TET can be estimated using the final position fix from 703 ST.	Time of the end of transmission for the float.	Time, Position n-1	2. Value transmitted by the float
903	Taken from surface pressure in msg file: SurfacePressure=0.01 Or from log file: (Dec 22 2008 18:49:46, 0 sec) DescentInit() Deep profile 6 initiated at mission-time 888601sec. (Dec 22 2008 18:49:48, 2 sec) DescentInit() Surface pressure: 0.3dbars *This is the pressure in dbar as sampled by the CTD. This is in all versions of Navis firmware. The only tricky bit is that when the floats are in ice evasion mode and not coming to the surface, the SurfacePressure is not updated until the float surfaces again. So the value of SurfacePressure is the pressure taken the last time the float surfaced. In some case this could be several months prior.*	Surface pressure offset value	Pressure n	2. Value transmitted by the float

2.2.7 Apex APF11 Argos floats with firmware version 2.8.0 or 2.10.4

Argo trajectory file measurement codes (MC) for Apex APF11 Argos floats				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
0 (launch position)	Provided by PI (from deployment team).	From Coriolis Excel deployment file provided at Coriolis DAC by float PI.	Time, latitude, longitude. Cycle #-1 (convention).	4 (value is determined by satellite)
100 (DST)	Duplication of TET of the previous cycle.		Time. Cycle #N.	3 (value is directly computed from relevant, transmitted float information)
Start of the drift phase				
287 = PET-13 (minimum meas of park-level PT samples supporting meas)	'Pressure associated with Tmin of park-level PT samples' and 'Minimum temperature of park-level PT samples'.	Argos float data message.	PRES, TEMP. Cycle #N.	No time

288 = PET-12 (maximum meas of park-level PT samples supporting meas)	'Pressure associated with Tmax of park-level PT samples' and 'Maximum temperature of park-level PT samples'.	Argos float data message.	PRES, TEMP. Cycle #N.	No time
293 = PET-7 (mean PRES diff of park-level PT samples)	'Mean pressure diff of park-level PT samples'.	Argos float data message.	PRES. Cycle #N.	No time
294 = PET-6 (standard deviation of meas of park-level PT samples)	'Standard deviation of pressure of park-level PT samples' and 'Standard deviation of temperature of park-level PT samples'.	Argos float data message.	PRES, TEMP. Cycle #N.	No time
296 = PET-4 (mean TEMP of park-level PT samples)	'Mean temperature of park-level PT samples'.	Argos float data message.	TEMP. Cycle #N.	No time
297 = PET-3 (minimum meas of park-level PT samples)	'Minimum pressure of park-level PT samples' and 'Minimum temperature of park-level PT samples'.	Argos float data message.	PRES, TEMP. Cycle #N.	No time
298 = PET-2 (maximum meas of park-level PT samples)	'Maximum pressure of park-level PT samples' and 'Maximum temperature of park-level PT samples'.	Argos float data message.	PRES, TEMP. Cycle #N.	No time
300 (PET)	PET is not set when PARK_PRESSURE = PROFILE_PRESSURE (since PET = AST) otherwise PET = TET – UpTime – DeepDescentTimeout.		Time. Cycle #N.	3 (value is directly computed from relevant, transmitted float information)
301 (representative park measurement)	Average value of measurements stored with MC=287 and MC=288. REPRESENTATIVE_PARK_PRESSURE_STATUS = 5.		PRES, TEMP. Cycle #N.	No time
End of drift measurements				
501	DOWN_TIME_END = 'RTC time when down time expired'.	Argos float data message.	Time. Cycle #N.	2 (value is transmitted by the float)
400 (DDET)	DDET = AST.		Time. Cycle #N.	2 (value is transmitted by the float)
Start of profile				
500 (AST)	AST = DOWN_TIME_END.		Time. Cycle #N.	2 (value is transmitted by the float)
503 (deepest measurement)	Deepest value of PTS profile.	Argos float data message.	PRES, TEMP. Cycle #N.	No time
Float is on the surface				
700 (TST)	Computed from Argos satellite times (and float transmission strategy).	Argos float data message and CLS information.	Time. Cycle #N.	1 (value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour)
701 (TST transmitted by the float)	Computed from 'Time when telemetry phase was initiated relative to down time end'. This time is stored in Real Time only; in Delayed Mode, once checked this value should	Argos float data message.	Time. Cycle #N.	3 (value is directly computed from relevant, transmitted float information)

	replace TST.			
702 (FMT)	Earliest time of current cycle Argos messages.	CLS information (Argos message dates).	Time. Cycle #N.	4 (value is determined by satellite)
703 (surface location)	All Argos fixes provided.	CLS information (Argos fixes estimated by CLS).	Time, latitude, longitude, location class. Cycle #N.	4 (value is determined by satellite)
704 (LMT)	Latest time of current cycle Argos messages.	CLS information (Argos message dates).	Time. Cycle #N.	4 (value is determined by satellite)
800 (TET)	TET = DOWN_TIME_END + UpTime. If UpTime is unknown (and couldn't be estimated) TET is estimated from LMTs.		Time. Cycle #N.	3 (value is directly computed from relevant, transmitted float information)

2.2.8 Apex APF11 floats with Iridium

Coriolis decoder for Apex APF11 Iridium floats (Firmware version 2.10.1 or 2.11.1, Coriolis version 2.10.1 or 2.11.1, Decoder Id 1321 or 1322).

Notes:

1. Cycle times DST, PST, PET and AET are provided in science_log and system_log files (associated timestamps may be slightly different however). Coriolis decided to keep the science_log file ones so that one can consistently associate a pressure to each time using the CTD_P measurements provided in this file.

2. Cycle time adjustment: cycle times can be adjusted in real time from float clock drift that can be estimated from 'GPS Skew' information provided in system_log file.

Coriolis chooses to include as much data as possible from the floats which is great, but not some measurements codes are not required including buoyancy adjustments. These are highlighted in grey in the following table.

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
0 (launch)	Ideally would come from Ship deployment records/logs. From zero cycle file system log **000** In pressure activated mode – time of activation: 20171212T112421|5|IDLE|Activation pressure detected: 168.99 dbar 20171212T112421|5|go_to_state|Mission state IDLE -> PRELUDE	Time of launch Position at end of prelude cycle	Time, position N = 0	0 if taken from ship metdata 2 if taken from 000.msg file

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	Later GPS fix appears in the file twice: 20171212T121045|5|update_almanac|Updating GPS Almanac 12/12/2017 12:10 20171212T122549|3|read_str|COM6 RX Overrun Detected! 20171212T122550|5|RMC|Set Clock: 12/12/2017 12:25:50 20171212T122550|5|GPS|GPS TimeToFix: 2 secs 20171212T122550|5|GPS|GPS Skew: -2 secs 20171212T122550|5|GPS|GPS Fix: 12/12/2017 12:25:50,-28.79842,-158.99509,9 20171212T122550|5|wait_for_done|GPS time/location set 20171212T122550|5|test|RF Board Max Current: 37.4 mA 20171212T122550|5|test|Battery Min Voltage: 15.1 V 20171212T122550|5|test| GPS Test : <<PASS>> 20171212T122552|5|update_almanac|Updating GPS Almanac 12/12/2017 12:25 20171212T124058|5|update_offset|Surface Offset Pressure: 0.0300 20171212T124058|5|PRELUDE| Self Test : <<PASS>> 20171212T124059|5|RMC|Set Clock: 12/12/2017 12:40:59 20171212T124059|5|GPS|GPS TimeToFix: 2 secs 20171212T124059|5|GPS|GPS Skew: 0 secs 20171212T124059|5|GPS|GPS Fix: 12/12/2017 12:40:59,-28.79814,-158.99414,9 Do we include these positions as launch, or as TST? Would need to expand this for manual activation – need an example			
90 = DST – 10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled after deployment and before DST.	science_log file	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
100 DST	Change in state to park descent. From *.system.log file: 20180202T081820|5|go_to_state|Mission state SURFACE -> PARKDESCENT OR From *.science_log file: Message,20180202T081826,Park Descent Mission*********	System_log file Or Science_log file	Time, PRES Cycle #N	2. value is transmitted by the float
189 = DET-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) From the *.system_log.txt file: 20180201T214841|5|ASCENT|Adjusting Buoyancy to 578	Buoyancy adjustments made during descent (between DST	Time, PRES Cycle #N	2. value is transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	20180201T214842|5|buoyancy_engine_task|Buoyancy Start Position: 280 20180201T215035|5|buoyancy_engine_task|Buoyancy engine destination 578 reached after 01:52 . 20180201T215610|5|start_profile|Continuous Profile Started********** 20180201T221848|5|ASCENT|Ascending Too Slowly: 0.079 dbar/sec @ 1851.3 dbar 20180201T221854|5|ASCENT|Ascending Too Slowly: 0.079 dbar/sec @ 1850.9 dbar 20180201T221907|5|ASCENT|Ascending Too Slowly: 0.079 dbar/sec @ 1849.8 dbar 20180201T221907|5|do_ascent|target_position: 742.000000	and DET) in system_log file		
190 = DET-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between DST and DET. From *.science_log file: Message,20171115T174117,Park Descent Mission********** CTD_P,20171115T175601,25.99 CTD_P,20171115T175603,26.24 CTD_P,20171115T185606,524.39 CTD_P,20171115T185608,524.58 CTD_P,20171115T195613,843.10 CTD_P,20171115T195615,843.19 CTD_P,20171115T205620,1021.85 CTD_P,20171115T205622,1021.90	Descending CTD measurements Science_log file	Time, all available measurements. Cycle #N.	2. value is transmitted by the float
200 DET	**Best option and set this MC here only if it occurs before PST:** Computed, from science_log file CTD_P data, as the first time the float enters in the [PARK_PRES-3%;PARK_PRES+3%] interval. Associated pressure is the first CTD_P of [PARK_PRES-3%;PARK_PRES+3%] interval. Data looks like this: Message,20171212T131937,Park Descent Mission********* CTD_P,20171212T133305,27.990 CTD_P,20171212T143309,516.390 CTD_P,20171212T153314,881.230 CTD_P,20171212T163319,1095.690 Message,20171212T163325,Park Mission********** **2nd option and set this MC here only if it occurs before PST:** When the float has reached park depth. From the *.system_log.txt file: 20171212T163320|5|PARKDESCENT|Reached Park Depth: 1095.80 dbar	Descent end time. Time when float first approaches within 3% of the configured drift pressure. Float may be transitioning from the surface or from a deep profile. This variable is based on pressure only and can be measured or estimated by fall-rate. In the case of a float that overshoots the drift pressure on descent, DET is the time of the overshoot. Science_log file Or	Time n	3 (value is directly computed from relevant, transmitted float information) Or 2. Value is transmitted by the float Or 2. Value is transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	3rd option and set this MC after PST if a timeout: If 'Reached Park Depth' not available in the *.system_log.txt file due to a timeout, look for: 20180112T001404|4|PARKDESCENT|Park Descent Timeout: 314 min @ 962.1 dbar	system_log file		
239 = PST-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) Buoyancy action (time and pressure) recorded in system_log file between DET and PST.	Buoyancy adjustments made during descent (between DET and PST) system_log file	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
240 = PST-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between DET and PST.	science_log file	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
Start of the drift phase				
250 PST	From 'Park Mission' of science_log file. Associated pressure is the CTD_P measurement with the same timestamp if exists; otherwise the average value of the 2 surrounding CTD_P measurements. OR Based on float logic. A statement that it has reached the depth. From the *.system_log.txt file: 20171116T090439|5|go_to_state|Mission state PARKDESCENT -> PARK	Time of park start Time when float transitions to its Park or Drift mission. This variable is based on float logic. Science_log file OR System_log file	Time n	2. Value is transmitted by the float
289 = PET-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) Buoyancy action (time and pressure) recorded in system_log file between PST and PET. From the *.system_log.txt file: 20180118T221551|5|PARK|Adjusting Buoyancy to 1091	Time of active buoyancy adjustment during park phase system_log file	Time, PRES. n	2 (value is transmitted by the float)
290 = PET-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between PST and PET. Message,20171116T090444,Park Mission********** CTD_P,20171116T090447,1026.53 CTD_PTSH,20171116T090509,1027.10,1.5959,34.7070,-0.977961 O2,20171116T090512,231.35490,52.98427,1.60344,42.96755,40.87555,49.02606,8.15051,488.42319,802.80371,648.11133	A series of pressure measurements taken daily during drift. Can assign JULD values for APF11 floats. Science_log file	Time, all available measurements. n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	CTD_P,20171116T090514,1027.25 CTD_P,20171116T100518,1045.72 CTD_PTSH,20171116T100541,1045.70,1.5607,34.7050,-0.978141 O2,20171116T100544,235.17410,53.80836,1.56928,42.78669,40.69469,48.86159,8.16690,485.97198,803.06158,649.16498 CTD_P,20171116T100546,1045.68 CTD_P,20171116T110550,1054.54 Message,20171212T163830,Deep Descent Mission**********			
300 PET	From the *.system_log.txt file: 20171116T193820|5|go_to_state|Mission state PARK -> DEEPDESCENT In the case of a float going from park to ascent, it might look like this: 20171116T193820|5|go_to_state|Mission state PARK -> ASCENT OR From the science_log file: From 'Deep Descent Mission' (or 'Profiling Mission' in the case of a float going from park to ascent) of science_log file. Associated pressure is the CTD_P measurement with the same timestamp if exists; otherwise the average value of the 2 surrounding CTD_P measurements.	Time when float exits from its Park or Drift mission. It may next rise to the surface (AST) or sink to profile depth (DDET) System_log file or science_log file	Time, PRES. n	2: value transmitted by the float
301 (representative park measurement)	Averaged values of CTD_PTS, CTD_PTSH and O2 measurements sampled during the [Park start time;Park end time] time interval (MC = 290). REPRESENTATIVE_PARK_PRESSURE_STATUS = 1.	science_log file	All available measurements. n	3: value is directly computed from relevant, transmitted float information
End of drift measurements				
389 = DDET-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) Buoyancy action (time and pressure) recorded in system_log file between PET and DDET. From the system_log.txt file 20180518T162610|5|DEEPDESCENT|Adjusting Buoyancy to 187	Buoyancy adjustments made during deep descent (between PET and DDET) system_log file	Time, PRES. n.	2: value transmitted by the float
390 = DDET-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between PET and DDET. Measurements between:	Time stamped measurements collected down to deep descent. science_log file	Time, all available measurements. n.	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	Message,20171116T193828,Deep Descent Mission********** And Message,20171116T222642,Profiling Mission********** From the *.science_log.csv file: Message,20171116T193828,Deep Descent Mission********** CTD_P,20171116T194757,1088.93 CTD_P,20171116T194759,1089.14 CTD_P,20171116T195304,1133.28 CTD_P,20171116T195306,1133.51 CTD_P,20171116T195811,1177.30 CTD_P,20171116T195813,1177.49 CTD_P,20171116T200318,1219.82 CTD_P,20171116T200320,1220.02 CTD_P,20171116T200825,1260.75 CTD_P,20171116T200827,1260.97 CTD_P,20171116T201332,1301.30 CTD_P,20171116T201334,1301.51			
400 DDET	Calculate this from the measurements between: Message,20171116T193828,Deep Descent Mission********** And Message,20171116T222642,Profiling Mission********** 400 is the first value within 3% of configured deep depth. **This MC is set only if it occurs before AST.** From the *.science_log.csv file (continued on from the snippet in code 399, above): CTD_P,20171116T220101,1922.43 399 CTD_P,20171116T220606,1940.07 400 DDET (first measurement within 3% of 2000db) Where do we put Pressure value? 490? CTD_P,20171116T220608,1940.15 490 (measurements between 400 DDET and 500 AST) CTD_P,20171116T221113,1956.34 490 CTD_P,20171116T221115,1956.43 490 CTD_P,20171116T221620,1971.78 490 CTD_P,20171116T221622,1971.86 490 CTD_P,20171116T222127,1986.43 490 CTD_P,20171116T222129,1986.53 490 CTD_P,20171116T222634,1999.88 490 CTD_P,20171116T222636,1999.92 490 Message,20171116T222642,Profiling Mission**********	Time when float first approaches within 3% of the eventual deep profile pressure. This variable is based on pressure only and can be measured or estimated by fall-rate. science_log file	Time, PRES. n.	3 (value is directly computed from relevant, transmitted float information) OR 2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
489 = AST-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) Buoyancy action (time and pressure) recorded in system_log file between DDET and AST.	Buoyancy adjustments made during descent (between DDET and AST) system_log file	Time, PRES. n.	2 (value is transmitted by the float)
490 = AST-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between DDET and AST. From the *.science_log.csv file (same snippet from code 400, above): CTD_P,20171116T220608,1940.15 490 (measurements between 400 DDET and 500 AST) CTD_P,20171116T221113,1956.34 490 CTD_P,20171116T221115,1956.43 490 CTD_P,20171116T221620,1971.78 490 CTD_P,20171116T221622,1971.86 490 CTD_P,20171116T222127,1986.43 490 CTD_P,20171116T222129,1986.53 490 CTD_P,20171116T222634,1999.88 490 CTD_P,20171116T222636,1999.92 490 Message,20171116T222642,Profiling Mission**********	A series of measurements transitioning towards 500, AST. science_log file	Time, all available measurements. n.	2: value transmitted by the float
500 AST	From *.system_log.txt file: 20171116T222637|5|go_to_state|Mission state DEEPDESCENT -> ASCENT In the case of a float going from park to ascent, it might look like this: 20171116T193820|5|go_to_state|Mission state PARK -> ASCENT OR From the science_log file: From 'Profiling Mission' of science_log file. Associated pressure is the CTD_P measurement with the same timestamp if exists; otherwise the average value of the 2 surrounding CTD_P measurements.	System_log file Or Science_log file	Time n	2: value transmitted by the float
503 (deepest measurement)	Deepest value from CTD_CP, CTD_CP_H or profile CTD_PTS, CTD_PTSH measurements.	science_log file	Time (if available), all available measurements. Cycle #N.	2: value transmitted by the float
589 = AET-11 (buoyancy actions)	Could put buoyancy adjustments in here if you wanted to (optional) Buoyancy action (time and pressure) recorded in system_log file between AST and AET.	Buoyancy adjustments made during descent (between AST and AET)	Time, PRES. n.	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	From the *.system_log.txt file: 20180518T101453|5|ASCENT|Adjusting Buoyancy to 570	system_log file		
590 = AET-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between AST and AET. Cyan rows are the profile data and go in the profile netcdf. They can be duplicated in the trajectory file if desired. Green rows are rise rate and go in traj file with code 590. From the *.science_log.csv file: Message,20171116T222642,Profiling Mission********** CTD_P,20171116T222645,2000.28 CTD_P,20171116T222646,2000.33 CTD_P,20171116T222846,1998.49 CTD_PTSH,20171116T222908,1996.50,0.7812,34.6920,-0.981964 O2,20171116T222911,244.32410,54.70848,0.79020,42.75586,40.66386,48.78920,8.12534,482.74139,807.42743,673.09949 CTD_P,20171116T222913,1995.97 CTD_P,20171116T223017,1990.00 CTD_P,20171116T223019,1989.82 CTD_P,20171116T223123,1983.93 CTD_P,20171116T223125,1983.82 CTD_P,20171116T223229,1977.94 CTD_P,20171116T223231,1977.76 CTD_P,20171116T223335,1972.03 CTD_P,20171116T223337,1971.84 CTD_P,20171116T223441,1966.22 CTD_P,20171117T045001,4.32 CTD_P,20171117T045017,3.41 CTD_P,20171117T045019,3.34 Message,20171117T045024,Surface Mission**********	Point sample pressures on ascent to surface. science_log file	Time, all available measurements. n.	2: value transmitted by the float
600 AET	From the *.system_log.txt file: 20171117T045019|5|go_to_state|Mission state ASCENT -> SURFACE OR From the science_log file: From Surface Mission' of science_log file. Associated pressure is the CTD_P measurement with the same timestamp if exists; otherwise the average value of the 2 surrounding CTD_P measurements.	Time of change to surface mission. Will likely be before the float reaches the surface. System_log file OR science_log file	Time n	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
Float is on the surface				
690 = TST-10 (relative series of measurements)	All timestamped measures provided in science_log files except O2 relative ones (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, etc… depending on the sensors mounted on the float) that have been sampled between AET and TST.	science_log file	Time, all available measurements. Cycle #N.	2: value transmitted by the float
700 TST	From the *.system_log.txt file: 20171117T050104|5|connect|Received CONNECT Or From first occurrence of 'Found sky.' in system_log file. For RBR prototype float, this is the n profile for this float. 20180519T031207|5|network_quality|Modem Quality = 5 %taken from the end of the file. Not the start, which is associated with n-1 and is not used for trajectory files.	Time of first z-modem activity. System_log file	Time **Cycle n-1** Or Time. Cycle n.	2: value transmitted by the float
703 ST	Multiple fixes may appear and all should be included here. The first fix in the science_log and system_log files is the location for profile n-1, the last fix is the location for profile n. Every fix is additional to any from previous profile files. Also, the float may produce additional files for any given profile if the float has trouble connecting. These files will be smaller but contain more GPS fixes and surface pressure measurements and they will apply to profile n. EG: -rw-rw-rw- 1 18157 argo-hf 240 Feb 23 10:56 f8157.007.20180202T081336.vitals_log.bin -rw-rw-rw- 1 18157 argo-hf 433 Feb 23 10:55 f8157.007.20180202T081336.vitals_log.csv -rw-rw-rw- 1 18157 argo-hf 15236 Feb 23 10:56 f8157.007.20180202T081758.science_log.bin -rw-rw-rw- 1 18157 argo-hf 39696 Feb 23 10:56 f8157.007.20180202T081758.science_log.csv -rw-rw-rw- 1 18157 argo-hf 12716 Feb 23 10:56 f8157.007.20180212T073604.system_log.txt -rw-rw-rw- 1 18157 argo-hf 169 Feb 23 10:56 f8157.007.20180212T073834.vitals_log.bin -rw-rw-rw- 1 18157 argo-hf 361 Feb 23 10:56 f8157.007.20180212T073834.vitals_log.csv -rw-rw-rw- 1 18157 argo-hf 247 Feb 23 10:56 f8157.007.20180212T082954.science_log.bin -rw-rw-rw- 1 18157 argo-hf 550 Feb 23 10:56 f8157.007.20180212T082954.science_log.csv -rw-rw-rw- 1 18157 argo-hf 6155 Feb 23 10:55 f8157.007.20180212T110402.system_log.txt From the *.system_log.txt file:	Satellite times and locations. One for each fix.	Time, Position n-1 and n	4. value is determined by satellite

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	20180202T081758|5|GPS|GPS Skew: 0 secs 20180202T081758|5|GPS|GPS Fix: 02/02/2018 08:17:58,-31.62194,-164.89915,77 Fix for profile n-1 20180202T081758|5|wait_for_done|GPS time/location set 20180202T081820|5|go_to_state|Mission state SURFACE -> PARKDESCENT 20180212T073341|5|stop_profilers|Stopping profilers 20180212T073352|5|update_offset|Surface Offset Pressure: 0.4900 20180212T073603|5|RMC|Set Clock: 02/12/2018 07:36:03 20180212T073603|5|GPS|GPS TimeToFix: 126 secs 20180212T073603|5|GPS|GPS Skew: 0 secs 20180212T073603|5|GPS|GPS Fix: 02/12/2018 07:36:03,-31.71048,-165.08408,67 Fix for profile n 20180212T073603|5|wait_for_done|GPS time/location set 20180212T073603|5|SURFACE|Completing Mission No.: 7 From the *.science_log.csv file: Message,20180202T081758,Firmware: 03/06/17 21:21:20 APF11-2MB-v2.5.2 Message,20180202T081758,FloatId/Username: f8157 GPS,20180202T081758,-31.6219,-164.8992,7 Fix for profile n-1 CTD_P,20180202T081759,0.110 Message,20180212T073055,Surface Mission********** CTD_P,20180212T073057,0.410 CTD_P,20180212T073310,0.390 CTD_P,20180212T073341,0.280 CTD_P,20180212T073352,0.490 GPS,20180212T073603,-31.7105,-165.0841,6 Fix for profile n.			
702 FMT, and 704 LMT	Don't use for APF11 floats	Time of first/last iridium message	Time n-1	4. value is determined by satellite
790 = TET-10 (relative series of measurements)	All timestamped measures provided in science_log files (CTD_P, CTD_PT, CTD_PTS, CTD_PTSH, O2, etc… depending on the sensors mounted on the float) that have been sampled between TST and TET.	science_log file	Time, all available measurements. Cycle #N.	2: value transmitted by the float
800 TET	End of upload of files to the modem. Applies to cycle n-1. From the *.system_log.txt file: 20180202T081728|5|zmodem_upload_files|Uploaded: f8157.006.20180123T085222.science_log.bin.gz 20180202T081728|5|zmodem_upload_files|Uploaded: f8157.006.20180202T081104.system_log.txt.gz 20180202T081728|5|zmodem_upload_files|Uploade	Time of last z-modem activity	Time n**-1**	2: value transmitted by the float

Argo trajectory file measurement codes (MC) for APF11				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
	d: f8157.006.20180123T084118.vitals_log.bin.gz 20180202T081728|5|zmodem_upload_files|Uploaded 12489 bytes in 99 secs at 126.1515 bytes/sec			
710 (in-water samples, part of surface sequence)	For O2 floats only. All O2 relative measurements sampled between AET and 'Inflating air bladder' time (provided in system_log file).	science_log file	Time, all available measurements. Cycle #N.	2: value transmitted by the float
711 (in-air samples, part of surface sequence)	For O2 floats only. All O2 relative measurements sampled between 'Inflating air bladder' time (provided in system_log file) and TST.	science_log file	Time, all available measurements. Cycle #N.	2: value transmitted by the float
901 (grounded cycle)	Grounding detection (time and pressure) recorded in system_log file.	system_log file	Time, PRES. Cycle #N.	2: value transmitted by the float
903	Surface pressure measurements. From the *.system_log.txt file: 20180518T135447|5|update_offset|Surface Offset Pressure: 0.0900	Surface pressure offset value	Pressure n	2: value transmitted by the float

2.2.9 HM2000 floats

Argo program measurement codes (MC) for HM2000 floats				
Code (timing)	**HM2000 Variable**	**Description**	**Units**	**JULD_STATUS**
0 (launch)	Launch time and location as recorded by deployer If not recorded, use Fill Value	Time and location		0: value is estimated from pre-deployment information found in the metafile 9: value is not immediately available but may be estimated at a later date
100 (DST)	Estimated from engineering data		YYYY/MM/DD HH:MM:SS	3: value is directly computed from relevant, transmitted float information
150 (FST)	HM2000 float doesn't transmit FST, use Fill Value			
189	None, use Fill Value	Next two stabilization times given as hours and minutes elapsed since DST.		
200 (DET)	Estimated from engineering data		YYYY/MM/DD HH:MM:SS	3: value is directly computed from relevant, transmitted float information
250 (PST)	Use Fill Value			

During the drift phase, HM2000 floats measure pressure, temperature and salinity every 3 or 6 hours				
290	Series of pressure	A series of pressure measurements taken daily during drift. No time can be assigned to these pressures, so use Fill Value in JULD	Pressure	2: value is transmitted by the float
End of drift measurements				
300 (PET)	Use Fill Value			
301	Estimated from all drift depth	Best estimate of drift depth	Pressure	3: value is directly computed from relevant, transmitted float information
400 (DDET)	Use Fill Value			
500 (AST)	Estimated from engineering data		YYYY/MM/DD HH:MM:SS	3: value is directly computed from relevant, transmitted float information
590	none	Transmitted data is of the elapsed time for each vertical slice of ascent (from the max pressure to 2000 db for the first slice; and for each 100 dbar think slice until the surface)		
600 (AET)	Computed from TST - 3 min (fixed)		YYYY/MM/DD HH:MM:SS	3: value is directly computed from relevant, transmitted float information
700 (TST)	starting acquisition date of the BDS/GPS fix(es).		Time	2: value is transmitted by the float
702 (FMT)	Earliest time of all BDS messages received		YYYY/MM/DD HH:MM:SS	4: value is determined by satellite
703 (ST)	All times from BDS or GPS		Time, Position	4: value is determined by satellite
704 (LMT)	Last time of all BDS messages received		Time	4: value is determined by satellite
800 (TET)	Last time of float transmission		Time	2: value is transmitted by the float

The positioning system used by HM2000 float can be switched between BDS (BeiDou Satellite) and GPS. At present, all the floats deployed are using BDS for positioning.

When using the BDS system, when the float hits the surface, it tries to get a BDS position, then it transmits its messages to BDS and then it tries to get another BDS position. Please put the measurement codes in the correct chronological order depending on when the positions were obtained from BDS.

The manufacture provides a windows-based software to decode BDS messages and generate a data file for each profile.

The prototype of HM2000 float does not have a fixed cycle time (does not have time-outs just like Apex float) because the engineers from the manufacture (HSOE) didn't recognize the importance of trajectory information for ocean current estimation. The future products will be designed to have a fixed cycle time and record a time at each action (e.g. Descent start time, Descent end time, Ascent start time, Ascent end time, Transmission start time, Transmission end time...). Currently all the times except for satellite fixed times are estimated from technical information.

Ascent Start Time

The AST is estimated from the AET - time consumed by air pump (3 minutes) - time consumed by ascent phase. Users can get it from each data file.

Ascent End Time

The AET is computed from the corresponding TST. AET=TST - 3 minutes. (Present version of HM2000 float will wait 3 minutes for getting a BDS position after it hits the surface)

Transmission Start Time

The TST is obtained after the float hits the surface (3 minutes). It is transmitted by the float, and users can get the TST from each data file.

Transmission End Time

The TET is obtained after the float finishes transmitting messages. It is transmitted by the float, and users can get it from each data file.

Descent Start Time

The DST is computed as the TST - time consumed at each phase under water. Users can get it from each data file.

Descent End Time

The DET is computed as the AET - time consumed from its start drifting phase. Users can get it from each data file.

2.2.10 NEMO floats

Argo program measurement codes (MC) for NEMO floats				
Code (timing)	**NEMO Variable**	**Description**	**Units**	**JULD_STATUS**
0 (launch)	Launch time and location as recorded by deployer If not recorded, use Fill Value	Time and location		0: value is estimated from pre-deployment information found in the metafile 9: value is not immediately known, but believe it can be estimated later
100 (DST)	Descent_start_time OR Descent_starttime	See section 3.2.2.3.1	Time Time	2: value is transmitted by the float 2: value is transmitted by the float
200 (DET)	Not usually available, so use Fill Value unless timeout error is triggered. If timeout occurs, enter time of abort	If float doesn't reach parking depth in time, the descent is aborted and a timeout error is reported. If this happens, enter this value into DET. If not, use Fill Value. See section 3.2.2.3.3		9: value is not immediately known, but believe it can be estimated later 2: value is transmitted by the float
250 (PST)	Parking_start_time Not available, so use Fill Value	Only available for newer floats. Do not enter this if timeout error occurs. Use Fill Value in that case. See section 3.2.2.3.3	Time	2: value is transmitted by the float 9: value is not immediately known, but believe it can be estimated later
During the drift phase, NEMO floats measure time pressure and temperature				
290	What kind of drift measurements are made?? A series, an average??			
End of drift measurements				
300 (PET)	Upcast_start_time	Only available for newer floats. See section 3.2.2.3.4		2: value is transmitted by the float
301	Average pressure during drift	Best estimate of drift depth	Pressure	3: value is directly computed from relevant, transmitted float information
500 (AST)	Ascent_start_time Or Ascent_starttime	See section 3.2.2.3.6 Is this a time out value?? Does float start ascending if it hits profile pressure?	Time Time	2: value is transmitted by the float 2: value is transmitted by the float
600 (AET)	Surfacingtime Or	See section 3.2.2.3.7 Is this a time out	Time	2: value is transmitted by the float

	Ascent_end_time	value or does float know it is at the surface?	Time	2: value is transmitted by the float
700 (TST)	End_of_profile_time Or Surface_start_time	See section 3.2.2.3.8	Time Time	2: value is transmitted by the float 2: value is transmitted by the float
702 (FMT)	Earliest time of all Argos messages received		Time	4: value is determined by satellite
703 (ST)	All Argos times and locations		Time, Position	4: value is determined by satellite
704 (LMT)	Latest time of all Argos messages received		Time	4: value is determined by satellite
800 (TET)	Not available, so use Fill Value	See section 3.2.2.3.9		9: value is not immediately known, but believe it can be estimated later

2.2.10.1.1 Descent Start Time - NEMO

DST is called **descent_start_time** or **descent_starttime**.

The DST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 100 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.10.1.2 First Stabilization Time - NEMO

FST is not measured by the float and should be excluded from the N_MEASUREMENT array.

In the N_CYCLE array, the value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to fill value.

2.2.10.1.3 Descent End Time & Park Start Time - NEMO

PST is called **parking_start_time** and is not available for all floats. For newer floats with serial numbers >113 this time will be in the recorded technical data. The time recorded here is either when the floats reaches programmed parking depth under a controlled descent (actually measuring the pressure) or at a programmed count of the pump for a parkcount descent. Both of these times are based either on an actual measurement of pressure or on a descent timer, so they are characterized as Park Start Time rather than Descent End Time. The PST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 250 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

The only exception is for a timeout error when the float, during a controlled descent, is not able to reach the parking depth and aborts the procedure after a predetermined time. In this case the time of the abort is recorded. If this timeout occurs, the time of abort should be recorded in the Descent End Time variable as it is a timeout value. The DET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 200 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_END variable and the JULD_DESCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

In the N_CYCLE array, fill value should be stored in the JULD_DESCENT_END, JULD_DESCENT_END_STATUS, JULD_PARK_START and JULD_PARK_START_STATUS variables.

2.2.10.1.4 Park End Time - NEMO

PET is called **upcast_start_time**. This variable is available for all floats with serial number >113. While this may seem like an odd name, the float manufacturer chose this following their internal logic.

The PET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 300 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If the **upcast_start_time** is not available, it should be excluded from the N_MEASUREMENT array.

In the N_CYCLE array, fill value should be stored in the JULD_PARK_END and the JULD_PARK_END_STATUS variables.

2.2.10.1.5 Deep Descent End Time - NEMO

DDET is not an event for this float, so it should be excluded from the N_MEASUREMENT array.

In the N_CYCLE array, fill value should be stored in the JULD_DEEP_DESCENT_END and the JULD_DEEP_DESCENT_END_STATUS variables.

2.2.10.1.6 Ascent Start Time - NEMO

AST is called **ascent_start_time** or **ascent_starttime**.

The AST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 500 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.10.1.7 Ascent End Time - NEMO

AET is called **surfacingtime** or **ascent_end_time**.

The AET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 600 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.10.1.8 Transmission Start Time - NEMO

TST is called **end_of_profile_time** or **surface_start_time**.

The TST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 700 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.10.1.9 Transmission End Time - NEMO

TET is not known for the actual profile, the float starts descending immediately after transmission. But the DST for the next profile will be the TET of the current profile. Before this time is known, the TET is a fill value with a status of "9".

The TET should be set to fill value for the current cycle in the JULD variable in the N_MEASUREMENT array with an MC = 800 and STATUS set to 9: value is not immediately known, but believe it can be estimated later.

When the next cycle arrives, the TET should be set to the DST of current profile in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with an MC = 800 and STATUS set to 2: value is transmitted by float.

For the N_CYCLE array, fill value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 9 for the current cycle.

Once the DST of the next profile occurs, and hence the TET of the previous profile is known, the TET can be filled in the previous cycle.

The value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11 NINJA floats

There are two types of NINJA floats: those deployed in 2002 - 2007 and those deployed in 2008. For the floats deployed in 2002 - 2007, some times are directly provided by the float (the day (day number in the current month), hours, minutes and seconds of the event are transmitted). Other times must be computed from technical information.

All these times must be corrected for clock offset before storage in the N_CYCLE and N_MEASUREMENT arrays.

Argo program measurement codes (MC) for NINJA 300001, 300002, 300003 floats				
Code (timing)	**NINJA Variable**	**Description**	**Units**	**JULD_STATUS**
0 (launch)	Launch time and location as recorded by deployer If not recorded, use Fill Value	Time and location		0: value is estimated from pre-deployment information found in the metafile 9: value is not immediately available but may be estimated at a later date
100 (DST)	Descent_Start_Day	See section 3.2.2.4.1.1	Day number in the month, hours, minutes and seconds	2: value is transmitted by the float
150 (FST)	First Stabilization Time Pressure provided with time	First of three stabilization times provided as hours and minutes elapsed since DST. See section 3.2.2.4.1.2	Time (hours and minutes since DST) Pressure	2: value is transmitted by the float 2: value is transmitted by the float
189	Second and Third Stabilization Times Pressure provided with time.	Next two stabilization times given as hours and minutes elapsed since DST. See section 3.2.2.4.1.2	Time (hours and minutes since DST) Pressure	2: value is transmitted by the float 2: value is transmitted by the float
200 (DET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
250 (PST)	Parking_Depth_in_Time Pressure	See section 3.2.2.4.1.3	Day number in the month, hours, minutes and seconds Pressure	2: value is transmitted by the float 2: value is transmitted by the float
During the drift phase, NINJA 300001, 300002, 300003 floats measure pressure daily.				
290	Series of pressure	A series of pressure measurements taken daily during drift. No time can be assigned to these pressures, so use	Pressure	2: value is transmitted by the float

		Fill Value in JULD		
End of drift measurements				
300 (PET)	Not available, so use Fill Value Pressure	See section 3.2.2.4.1.4	 Pressure	9: value is not immediately known, but believe it can be estimated later
301	Average pressure during drift	Best estimate of drift depth	Pressure	3: value is directly computed from relevant, transmitted float information
400 (DDET)	Not available, so use Fill Value	See section 3.2.2.4.1.5		9: value is not immediately known, but believe it can be estimated later
500 (AST)	Ascent_Start_Day	See section 3.2.2.4.1.6	Day number in the month, hours, minutes and seconds	2: value is transmitted by the float
590	Times associated with ascending CTD measurements	Transmitted data is of the elapsed time for each vertical slice of ascent (from the max pressure to 2000 db for the first slice; and for each 100 dbar think slice until the surface)	Time	2: value is transmitted by the float
600 (AET)	AST + profile duration	See section 3.2.2.4.1.6	Day number in the month, hours, minutes and seconds	3: value is directly computed from relevant, transmitted float information
700 (TST)	ARGOS_Start_day	See section 3.2.2.4.1.8	Day number in the month, hours, minutes and seconds	2: value is transmitted by the float
702 (FMT)	Earliest time of all Argos messages received		Time	4: value is determined by satellite
703 (ST)	All Argos times and locations		Time, Position	4: value is determined by satellite
704 (LMT)	Latest time of all Argos messages received		Time	4: value is determined by satellite
800 (TET)	Not available, so use Fill Value	See section 3.2.2.4.1.9		9: value is not immediately known, but believe it can be estimated later

2.2.11.1.1 Dated events for NINJA 300001, 300002 and 300003 versions

2.2.11.1.1.1 Descent Start Time - NINJA

The DST is directly provided by these NINJA versions (**Descent_Start_Day**): the day (day number in the month), hours, minutes and seconds of DST are transmitted.

The DST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 100 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.2 First Stabilization Time - NINJA

Three Stabilization Times are provided by these NINJA versions (as hours and minutes elapsed since DST) with the associated pressures.

The **three** Stabilization Times and pressures should be stored in the JULD (or JULD_ADJUSTED if clock offset has been applied) and PRES variables in the N_MEASUREMENT array with the MC set to 150 for the first stabilization and 189 for the next two. STATUS should be set to 2: value transmitted by float.

In the N_CYCLE array, the first stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.3 Park Start Time - NINJA

The PST is directly provided by these NINJA versions (**Parking_Depth_in_Time**): the day (day number in the month), hours, minutes and seconds of DST are transmitted.

The associated pressure is also transmitted.

The time and pressure should be stored in the JULD (or JULD_ADJUSTED if clock offset has been applied) and PRES variables in the N_MEASUREMENT array with the MC set to 250 and STATUS set to 2: value transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.4 Park End Time - NINJA

NINJA 30001 can only observe given information on the parking depth (parking depth = profile depth) just after it is deployed. Then, the times from PET to AST are not in the technical message. NINJA 30002 and 300003 can observe from profile depth which is deeper than parking depth, but their firmware are only updated a little from 30001. The times from PET to AST are not added in the technical message. Therefore, we cannot know the times from PET to AST.

Unfortunately, there is no constant amount of time from AST. Since the PET is an event that occurs for the float, without the time known, it will be fill value in the JULD & JULD_ADJUSTED variables in the N_MEASUREMENT array with an MC = 300 and STATUS set to 9 as it might be estimated at a later time.

In the N_CYCLE array, fill value should be stored in the JULD_PARK_END and the JULD_PARK_END_STATUS should be set to 9.

2.2.11.1.1.5 Deep Descent End Time - NINJA

NINJA floats do not record this time, so use fill value in the JULD and JULD_ADJUSTED variables in the N_MEASUREMENT array with an MC=400 and JULD_STATUS and JULD_ADJUSTED_STATUS of ‘9’.

In the N_CYCLE array, fill value should be stored in the JULD_DEEP_DESCENT_END variable and the JULD_DEEP_DESCENT_END_STATUS should be set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.6 Ascent Start Time - NINJA

The AST is directly provided by these NINJA versions (**Ascent_Start_Day**): the day (day number in the month), hours, minutes and seconds of AST are transmitted.

The AST should be stored in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with the MC set to 500 and STATUS set to 2: value transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.7 Ascent End Time - NINJA

The AET is not directly provided by these NINJA versions.

However, these floats provide the elapsed time for each vertical slice of ascent (from the max pressure to 2000 dbar for the first slice; and for each 100 dbar thick other slices until the surface).

The cumulative sum of these times is thus the profile duration and can be used to compute AET from AST.

The AET should be stored in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with the MC set to 600 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 3. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

AET is stored as Ascent_0db_time

2.2.11.1.1.8 Transmission Start Time - NINJA

The TST is directly provided by these NINJA versions: the day (day number in the month), hours, minutes and seconds of TST are transmitted. The variable is called “ARGOS_START_Day”.

The TST should be stored in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with the MC set to 700 and STATUS set to 2: value transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.1.9 Transmission End Time - NINJA

TET is not known. Since the TET is an event that occurs for the float, without the time known, it will be fill value in the JULD & JULD_ADJUSTED variables in the N_MEASUREMENT array with an MC = 800 and STATUS set to 9 as it might be estimated at a later time.

In the N_CYCLE array, fill value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.11.1.2 2008 NINJA floats

No timing information is available in real time or in delayed mode. Nothing should be included in the N_MEASUREMENT array. All cycle timing variables and their status flags should be fill value in the N_CYCLE array.

2.2.12 Deep NINJA floats

Deep NINJA floats profile to deeper than 2000db and the accuracy of the CTD data below 2000 db is not yet well understood. Currently, Argo is labeling these data with lower quality flags (2 and 3) in real time. So, if any pressure measurements included in the trajectory file are below 2000 db, they should be flagged with a 2 or 3 in real time.

Argo program measurement codes (MC) for DeepNINJA floats				
Code (timing)	**DeepNINJA Variable**	**Description**	**Units**	**JULD_STATUS**
0 (launch)	Launch time and location as recorded by deployer If not recorded, use Fill Value	Time and location		0: value is estimated from pre-deployment information found in the metafile 9: value is not immediately available but may be estimated at a later date
100 (DST)	Descent_Start_Time Pressure provided with time.	Time when float starts descending to the parking depth from sea surface.	Date and Time Pressure (dbar)	2: value is transmitted by the float 2: value is transmitted by the float
150 (FST)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
200 (DET)	Descent_End_Time Pressure provided with time.	Time when float reaches the parking depth and start drifting	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
250 (PST)	Use DET	Time when float reaches the parking depth and start drifting	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
During the drift phase, DeepNINJA floats measure time pressure, temperature and salinity hourly.				
290	Series of time, pressure, temperature and salinity measured during drift	A series of pressure measurements taken daily during drift.	Date and Time Pressure (dbar) Temperature (°C) Salinity(psu)	2: value is transmitted by the float
End of drift measurements				
300 (PET)	Deep_Descent_Start_Time	Time when float start descending from the parking depth to the profile depth.	Date and Time	2: value is transmitted by the float

	Pressure provided with time.		Pressure(dbar)	2: value is transmitted by the float
301	Average pressure during drift	Best estimate of drift depth	Pressure	3: value is directly computed from relevant, transmitted float information
400 (DDET)	Deep_Descent_End_Time Pressure provided with time.	Time when float reaches the profile depth.	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
500 (AST)	Ascent_Start_Time Pressure provided with time.	Time when float starts ascending to the sea surface	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
600 (AET)	Ascent_End_Time Pressure provided with time.	Time when float reaches the sea surface and stop ascending.	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
703 (ST)	GPS fix		Time, Position	2: value is transmitted by the float
700 (TST)	First Message Time Pressure provided with time	Time when the float transmits the first message	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float
702 (FMT)	Time when the first message is received		Date and Time	4: value is determined by satellite
704 (LMT)	Time when the last message is received		Date and Time	4: value is determined by satellite
800 (TET)	Transmit_END_Time Pressure provided with time.	Time when float stops transmitting.	Date and Time Pressure(dbar)	2: value is transmitted by the float 2: value is transmitted by the float

2.2.13 NOVA floats

The housekeeping data packet has most of the cycle timing variables in it. The variable names and how they are calculated are described below.

Argo program measurement codes (MC) for NOVA floats				
Code (timing)	**NOVA Variable**	**Description**	**Units**	**JULD_STATUS**

0 (launch)	Time of last GPS fix – PARAM 12 setting (accurate to +/-5 minutes) Time of activation (within one hour of GPS fix)	Launch time and location. See section 3.2.2.5.1 Launch time and location. See section 3.2.2.5.1	Time (seconds), position Time (seconds), position	3: value is directly computed from relevant, transmitted float information 2: value is transmitted by the float
100 (DST)	NVS/3 + time stamp of previous Iridium transmission	See section 3.2.2.5.2 DST not transmitted by float	NVS (no unit) Time of Iridium message	3: value is directly computed from relevant, transmitted float information
150 (FST)	FST	See section 3.2.2.5.3	Time (hours)	2: value is transmitted by the float
190	CTD taken during first descent after activation	CTD only taken on descent only after activation	Time (hours) Pressure (dbar) Temp (deg C) Salinity (psu)	2: value is transmitted by the float
200 (DET)	Deepest Temp,Pres pair taken during first descent after activation	CTD only taken on descent only after activation	Time (hours) Pressure (dbar) Temp (deg C) Salinity (psu)	2: value is transmitted by the float
250 (PST)	EDT	Float recognizes when it has stabilized at depth and changes into park phase. See section 3.2.2.5.4	Time (hours)	2: value is transmitted by the float
During the drift phase, NOVA floats measure time pressure and temperature at variable times. Choose the measurement code below that most appropriately describes what types of measurements are taken:				
290	Series of pressure Series of temperature Series of salinity	A series of CTD measurements taken during drift at user specified times	Time (hours) Pressure (dbar) Temp (deg C) Salinity (psu)	2: value is transmitted by the float
297	Minimum pressure	Minimum pressure recorded during drift phase	Pressure (dbar)	2: value is transmitted by the float
298	Maximum pressure	Maximum pressure recorded during drift phase	Pressure (dbar)	2: value is transmitted by the float
End of drift measurements				
300 (PET)	DDST	See section 3.2.2.5.5	Time (hours)	3: value is directly computed from relevant, transmitted float information
301	Average pressure during drift	Best estimate of drift depth	Pressure	3: value is directly computed from relevant, transmitted float information
400 (DDET)	DDET	See section 3.2.2.5.6	Time (hours)	2: value is transmitted by the float
500 (AST)	SAT	See section 3.2.2.5.7	Time (hours)	2: value is transmitted by the float
600 (AET)	EAT	See section 3.2.2.5.8	Time (hours)	2: value is transmitted by the float
703 (surface fixes)	All GPS fixes		Time, Position	2: value is transmitted by the float

700 (TST)	AET + SBDT from previous message	See section 3.2.2.5.9	Time (AET in hours, SBDT in seconds)	3: value is directly computed from relevant, transmitted float information
702 (FMT)	AET + SBDT from previous message	See section 3.2.2.5.9	Time (AET in hours, SBDT in seconds)	3: value is directly computed from relevant, transmitted float information
704 (LMT)	TST + SBDT from previous cycle	See section 3.2.2.5.10	Time (TST in hours, SBDT in seconds)	3: value is directly computed from relevant, transmitted float information
800 (TET)	TST + SBDT from previous cycle	See section 3.2.2.5.10	Time (TST in hours, SBDT in seconds)	3: value is directly computed from relevant, transmitted float information

2.2.13.1.1 Launch time

Once the magnet is removed, the float looks at the PARAMETER 12 setting (the delay before mission which can be set to between 0 and up to 1 hour). Once that time has elapsed, the float will acquire a GPS location and other diagnostic information and send a housekeeping message (time and location stamped) prior to beginning its mission. The message can be identified by its CYCLE COUNT = 255. Depending on how accurate you want to be:

a) Activation can be calculated from the housekeeping message:
Time of Last GPS fix – PARAM 12 setting = activation time to +/-5 minutes
b) The time of the activation will be within 1 hour of housekeeping message for cycle 255

2.2.13.1.2 Descent Start Time - NOVA

DST is not transmitted directly by the float, but must be calculated from the NVS variable. NVS is the number of valve activations at the surface. There is no unit on this and the minimum value is zero and the maximum value is 255. The start byte is 10 and the bit length is 8. The decoding equation is y = x.

DST = NVS/3 + time stamp of previous Iridium transmission.

The DST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 100 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 3.. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.3 First Stabilization Time - NOVA

FST is called FST and is the time in the day when the float first activated the valve during its descent. It is measured in hours, had a minimum value of zero and a maximum value of 23.9. The start byte is 5 and the bit length is 8. The decoding equation is y = 0.1 * x.

The FST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 150 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the first stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to

2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.4 Park Start Time - NOVA

PST is called EDT and is the time in the day when the float ended its descent to parking. It is measured in hours, has a minimum value of zero and a maximum value of 23.9. The start byte is 4 and the bit length is 8. The equation to calculate it is y = 0.1 * x. This time can change from cycle to cycle because the float recognizes it is stable at the parking depth.

The PST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 250 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.5 Park End Time - NOVA

PET is called PET and is the time in the day when the float started its descent to profile depth. The unit is hours, has a minimum value of zero and a maximum value of 23.9. The start byte is 6 and the bit length is 8. The equation to calculate it is y = 0.1 * x.

The PET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 300 and STATUS set to 2: value is transmitted by float.

2.2.13.1.6 Deep Descent End Time - NOVA

DDET is called DDET and is the time in the day when the float achieved its profile depth. The unit is hours, has a minimum value of zero and a maximum value of 23.9. The start byte is 7 and the bit length is 8. The decoding equation is y = 0.1 * x. The float recognizes when it is at the profile pressure and and reports this time as DDET. It can change from profile to profile.

The DDET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 400 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_DEEP_DESCENT_END variable and the JULD_DEEP_DESCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.7 Ascent Start Time - NOVA

AST is called SAT and is the time in the day when the float started its ascending profile. The unit is hours, has a minimum value of zero and a maximum value of 23.9. The start byte is 8 and the bit length is 8. The decoding equation is y = 0.1 * x. This time is set in PARAMETER 2, but should be sufficiently after DDET.

The AST should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 500 and STATUS set to 2: value is transmitted by float.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.8 Ascent End Time - NOVA

AET is called EAT and is the time in the day when the float ended its ascending profile. The unit is hours, has a minimum value of zero and a maximum value of 23.9. The start byte is 9 and the bit length is 8. The decoding equation is y = 0.1 * x. Once the CTD stops profiling at 6 db, the internal bladder is emptied and the float rises to the surface. At that time, the EAT is recorded. Afterwards, GPS acquisition starts and then Iridium transmission.

The AET should be in the JULD (or JULD_ADJUSTED if clock offset has been applied) variables in the N_MEASUREMENT array with an MC = 600 and STATUS set to 2: value is transmitted by float.

2.2.13.1.9 Transmission Start Time & First Message Time- NOVA

For Iridium, there are two values transmitted. When the float reaches the surface, it acquires a GPS position. The time to do this is represented by TTFF (in seconds). After the GPS is acquired, then the Iridium transceiver is activated. The time to do this is represented by SBDT (again in seconds). After completion of the transmission, a satellite check is done to look for incoming commands. If there is one, it is processed and then the float starts its next profile. Note that SBDT refers to the previous profile, not the current one, as it is calculated AFTER the Iridium transmission takes place. TST and FMT should be the same for NOVA floats

TST = AET + SBDT from previous cycle.

The TST & FMT should be set to fill value for the current cycle in the JULD variable in the N_MEASUREMENT array with an MC = 700 and STATUS set to 9: value is not immediately known, but believe it can be estimated later.

When the next cycle arrives, the TST & FMT should be filled in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with an MC = 700 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

For the N_CYCLE array, fill value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 9 for the current cycle.

Once the next profile occurs, and hence the TST & FMT of the previous profile is known, the TST & FMT can be filled in the previous cycle.

The value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 3. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.13.1.10 Transmission End Time & Last Message Time- NOVA

TET = TST + SBDT from previous cycle.

The TETand the LMT should be set to fill value for the current cycle in the JULD variable in the N_MEASUREMENT array with an MC = 800 and STATUS set to 9: value is not immediately known, but believe it can be estimated later.

When the next cycle arrives, the TET & LMT should be filled in the JULD (or JULD_ADJUSTED if clock offset has been applied) variable in the N_MEASUREMENT array with an MC = 800 and STATUS set to 3: value is directly computed from relevant, transmitted float information.

For the N_CYCLE array, fill value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 9 for the current cycle.

Once the next profile occurs, and hence the TET & LMT of the previous profile is known, the TET & LMTcan be filled in the previous cycle.

The value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS set to 3. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14 PROVOR floats

PROVOR floats directly provide, in the technical message, cycle timing information.

More precisely the provided times can be decoded to obtain the hours and minutes of a timed event but the corresponding day must be obtained by other means (see §2.2.14.1.8).

For PROVOR Argos floats, these times must be corrected for clock offset whereas for PROVOR Iridium floats the clock is set each cycle, thus the clock offset can be neglected.

Most of the times have an unusual time resolution (see §2.2.14.1.9) which must be stored in the data.

PROVOR floats do not provide a quick, easy real time estimate for Transmission End Time, but the float actually experiences this event. Therefore, in the N_MEASUREMENT array TET (MC = 800) should be fill value and its status flag should be a "9".

In the N_CYCLE array, fill value should be stored in the JULD_TRANSMISSION_END variable and the JULD_TRANSMISSION_END_STATUS should be set to 9.

PROVOR floats do experience both JULD_DESCENT_END (DET) and JULD_DEEP_DESCENT_END (DDET), but there is no way to know these times right now. Therefore, in the N_MEASUREMENT array DET (MC = 200) and DDET (MC = 400) should be fill value and its status flag should be a "9".

In the N_CYCLE array, fill value should be stored in the JULD_DESCENT_END and JULD_DEEP_DESCENT_END variables and JULD_DESCENT_END_STATUS and JULD_DEEP_DESCENT_END_STATUS should be set to 9.

2.2.14.1.1 Timed events for PROVOR 101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018 and 101019 versions

All status flags should be a "2" since they come directly from the float.

2.2.14.1.1.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "descent start time".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "float First Stabilization Time" (or "first float First Stabilization Time" for PROVOR 101018 and 101019 versions).

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "float stabilization pressure" (or "first float stabilization pressure" for PROVOR 101018 and 101019 versions).

The stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "end of descent time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.4 Park End Time

The hours and minutes of the PET are provided, in the technical message, by the technical parameter "profile descent start time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.5 Deep Park Start Time

The hours and minutes of the DPST are provided, in the technical message, by the technical parameter "profile descent stop time".

In the N_CYCLE array, the value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.6 Ascent Start Time

The hours and minutes of the AST are provided, in the technical message, by the technical parameter "profile ascent start time".

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.1.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "time at end of ascent".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018 and 101019 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Descent Start Time without clock offset applied (2.2.14.1.1.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.1.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	First Stabilization Time without clock offset applied (2.2.14.1.1.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.1.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Park Start Time without clock offset applied (2.2.14.1.1.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.1.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Park End Time without clock offset applied (2.2.14.1.1.4)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.1.4)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Deep Park Start Time without clock offset applied (2.2.14.1.1.5)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Deep Park Start Time with clock offset applied (2.2.14.1.1.5)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Ascent Start Time without clock offset applied (2.2.14.1.1.6)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.1.6)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	101011, 101012, 101014, 101015, 101013, 100001, 101017,	Ascent End Time without clock offset applied (2.2.14.1.1.7)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.1.7)	2: value is transmitted by float	

	101018, 101019			
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Transmission Start Time without clock offset applied (2.2.14.1.1.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.1.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	101011, 101012, 101014, 101015, 101013, 100001, 101017, 101018, 101019	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.2 Timed events for PROVOR 102002, 102003 and 102004 versions

2.2.14.1.2.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "descent start time".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "float First Stabilization Time".

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "float stabilization pressure" (except for PROVOR 102004 version).

In the N_CYCLE array, the stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "end of descent time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.4 Park End Time

The hours and minutes of the PET are provided, in the technical message, by the technical parameter "profile descent start time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.5 Deep Park Start Time

The hours and minutes of the DPST are provided, in the technical message, by the technical parameter "profile descent stop time".

In the N_CYCLE array, the value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.6 Ascent Start Time

The hours and minutes of the AST are provided, in the technical message, by the technical parameter "profile ascent start time".

In the N_CYCLE array, the corresponding number should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 14 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.2.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "time at end of ascent".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 102002, 102003 and 102004 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	102002, 102003, 102004	Descent Start Time without clock offset applied (2.2.14.1.2.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.2.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	102002, 102003, 102004	First Stabilization Time without clock offset applied (2.2.14.1.2.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.2.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	

250 PST	102002, 102003, 102004	Park Start Time without clock offset applied (2.2.14.1.2.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.2.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	102002, 102003, 102004	Park End Time without clock offset applied (2.2.14.1.2.4)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.2.4)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	102002, 102003, 102004	Deep Park Start Time without clock offset applied (2.2.14.1.2.5)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Deep Park Start Time with clock offset applied (2.2.14.1.2.5)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	102002, 102003, 102004	Ascent Start Time without clock offset applied (2.2.14.1.2.6)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.2.6)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	102002, 102003, 102004	Ascent End Time without clock offset applied (2.2.14.1.2.7	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.2.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	102002, 102003, 102004	Transmission Start Time without clock offset applied (2.2.14.1.2.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.2.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	102002, 102003, 102004	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.3 Timed events for PROVOR 101009, 101006, 101008 and 101010 versions

2.2.14.1.3.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "descent start time".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "float First Stabilization Time".

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "float stabilization pressure".

In the N_CYCLE array, the stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "end of descent time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.4 Park End Time

The PET is deduced from AST by the following relation: PET = AST - DELAI

where DELAI is a programmed meta-data parameter that determines the maximum amount of time given to the float for diving from PARKING to PROFILE depth.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.5 Deep Park Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the NC_CYCLE array, fill value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.6 Ascent Start Time

The hours and minutes of the AST are provided, in the technical message, by the technical parameter "profile ascent start time".

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.3.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "time at end of ascent".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been

estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 101009, 101006, 101008 and 101010 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	101009, 101006, 101008, 101010	Descent Start Time without clock offset applied (2.2.14.1.3.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.3.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	101009, 101006, 101008, 101010	First Stabilization Time without clock offset applied (2.2.14.1.3.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.3.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	101009, 101006, 101008, 101010	Park Start Time without clock offset applied (2.2.14.1.3.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.3.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	101009, 101006, 101008, 101010	Park End Time without clock offset applied (2.2.14.1.3.4)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.3.4)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	101009, 101006, 101008, 101010	Fill value (2.2.14.1.3.5)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.3.5)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	101009, 101006, 101008, 101010	Ascent Start Time without clock offset applied (2.2.14.1.3.6)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.3.6)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	101009, 101006, 101008, 101010	Ascent End Time without clock offset applied (2.2.14.1.3.7)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.3.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	101009, 101006, 101008, 101010	Transmission Start Time without clock offset applied (2.2.14.1.3.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.3.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	101009, 101006, 101008, 101010	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.4 Timed events for PROVOR 100006, 100005, 100004, 100008 and 100003 versions

2.2.14.1.4.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "descent start time".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "First Stabilization Time".

In the N_CYCLE array, the first stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "end of descent time".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.4 Park End Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the N_CYCLE array, fill value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.5 Deep Park Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

Fill value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.6 Ascent Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the N_CYCLE array, fill value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.4.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "end of resurfacing time".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 100006, 100005, 100004, 100008 and 100003 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	100006, 100005, 100004, 100008, 100003	Descent Start Time without clock offset applied (2.2.14.1.4.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.4.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	100006, 100005, 100004, 100008, 100003	First Stabilization Time without clock offset applied (2.2.14.1.4.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.4.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	100006, 100005, 100004, 100008, 100003	Park Start Time without clock offset applied (2.2.14.1.4.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.4.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	100006, 100005, 100004, 100008, 100003	Fill value (2.2.14.1.4.4)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.4.4)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	100006, 100005, 100004, 100008, 100003	Fill value (2.2.14.1.4.5)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.4.5)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	100006, 100005, 100004, 100008, 100003	Fill value (2.2.14.1.4.6)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.4.6)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	100006, 100005, 100004, 100008, 100003	Ascent End Time without clock offset applied (2.2.14.1.4.7)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.4.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	100006, 100005, 100004,	Transmission Start Time without clock offset applied (2.2.14.1.4.8)	2: value is transmitted by float	

	100008, 100003	**JULD_ADJUSTED**	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.4.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	100006, 100005, 100004, 100008, 100003	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.5 Timed events for PROVOR 101007 version

2.2.14.1.5.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "heure début de plongée".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "heure de première stabilisation".

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "pression de première stabilisation".

In the N_CYCLE array, the stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "heure de fin de descente".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.4 Park End Time

The PET is deduced from AST by the following relation: PET = AST - DELAI

where DELAI is a programmed meta-data parameter that determines the maximum amount of time given to the float for diving from PARKING to PROFILE depth.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.5 Deep Park Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the N_CYCLE array, fill value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.6 Ascent Start Time

The hours and minutes of the AST are provided, in the technical message, by the technical parameter "heure de début profil remontée".

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.5.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "heure de fin de remontée à la surface".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 101007 version				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	101007	Descent Start Time without clock offset applied (2.2.14.1.5.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.5.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	101007	First Stabilization Time without clock offset applied (2.2.14.1.5.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.5.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	101007	Park Start Time without clock offset applied (2.2.14.1.5.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.5.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	101007	Park End Time without clock offset applied (2.2.14.1.5.4)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.5.4)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	

450 DPST	101007	Fill value (2.2.14.1.5.5)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.5.5)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	101007	Ascent Start Time without clock offset applied (2.2.14.1.5.6)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.5.6)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	101007	Ascent End Time without clock offset applied (2.2.14.1.5.7)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.5.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	101007	Transmission Start Time without clock offset applied (2.2.14.1.5.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.5.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	101007	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.6 Timed events for PROVOR 101002, 101005 and 100002 versions

2.2.14.1.6.1 Descent Start Time

The hours and minutes of the DST are provided, in the technical message, by the technical parameter "heure début de plongée".

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "heure de première stabilisation".

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "pression de première stabilisation". This pressure should go in the PRES variable with an MC of 150.

In the N_CYCLE array, the stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "heure de fin de descente".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.4 Park End Time

The PET is deduced from AST by the following relation: PET = AST - *DELAI*

where *DELAI* is a programmed meta-data parameter that determines the maximum amount of time given to the float for diving from PARKING to PROFILE depth.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.5 Deep Park Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the N_CYCLE array, fill value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 9. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.6 Ascent Start Time

The AST is computed from TSD in float time (TSD_{FT}) (i.e. not already corrected from clock offset).

AST_{FT} = floor((TSD_{FT} - *MinProfDuration*)*24)/24

where:

- AST_{FT} is the AST in float time,
- *MinProfDuration* is the minimum profile duration, given by the latest CTD measurement time of the profile (these times are relative to AST_{FT}).

We thus assume that AST is programmed at a given hour (in float time), which is the case.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.6.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "heure de fin de remontée à la surface".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been

estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array PROVOR floats 101002, 101005 and 100002 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	101002, 101005, 100002	Descent Start Time without clock offset applied (2.2.14.1.6.1)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.6.1)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	101002, 101005, 100002	First Stabilization Time without clock offset applied (2.2.14.1.6.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.6.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	101002, 101005, 100002	Park Start Time without clock offset applied (2.2.14.1.6.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park Start Time with clock offset applied (2.2.14.1.6.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	101002, 101005, 100002	Park End Time without clock offset applied (2.2.14.1.6.4)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.6.4)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	101002, 101005, 100002	Fill value (2.2.14.1.6.5)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.6.5)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	101002, 101005, 100002	Ascent Start Time without clock offset applied (2.2.14.1.6.6)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.6.6)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	101002, 101005, 100002	Ascent End Time without clock offset applied (2.2.14.1.6.7)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.6.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	101002, 101005, 100002	Transmission Start Time without clock offset applied (2.2.14.1.6.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.6.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	101002, 101005, 100002	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.7 Timed events for PROVOR 101003 and 101004 versions

2.2.14.1.7.1 Descent Start Time

The hours and minutes of the Buoyancy Reduction Start Time (BRST) are provided, in the technical message, by the technical parameter "heure début de plongée".

The Number of Valve Actions at the Surface (NVAS) is provided, in the technical message, by the technical parameter "nombre d'actions EV en surface".

The DST is computed from BRST and NVAS:

DST = BRST + NVAS*130 seconds

In the N_CYCLE array, the value should be stored in the JULD_DESCENT_START variable and the JULD_DESCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.2 First Stabilization Time

The hours and minutes of the FST are provided, in the technical message, by the technical parameter "heure de première stabilisation".

The associated pressure (**in bars**) is also provided, in the technical message, by the technical parameter "pression de première stabilisation". This pressure should be stored in PRES with an MC code equal to 150.

In the N_CYCLE array, the stabilization value should be stored in the JULD_FIRST_STABILIZATION variable and the JULD_FIRST_STABILIZATION_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.3 Park Start Time

The hours and minutes of the PST are provided, in the technical message, by the technical parameter "heure de fin de descente".

In the N_CYCLE array, the value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.4 Park End Time

The PET is deduced from AST by the following relation: PET = AST - *DELAI*

where *DELAI* is a programmed meta-data parameter that determines the maximum amount of time given to the float for diving from PARKING to PROFILE depth.

In the N_CYCLE array, the value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.5 Deep Park Start Time

There is no easy way to get this time for this PROVOR float, so fill value should be used.

In the N_CYCLE array, fill value should be stored in the JULD_DEEP_PARK_START variable and the JULD_DEEP_PARK_START_STATUS set to 9. If the float clock offset has been estimated and

applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.6 Ascent Start Time

The AST is computed from TSD in float time (TSD_{FT}) (i.e. not already corrected from clock offset).

AST_{FT} = floor((TSD_{FT} - *MinProfDuration*)*24)/24

where:

- AST_{FT} is the AST in float time,
- *MinProfDuration* is the minimum profile duration, given by the latest CTD measurement time of the profile (these times are relative to ASTFT).

We thus assume that AST is programmed at a given hour (in float time), which is the case.

In the N_CYCLE array, the value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.7 Ascent End Time

The AET is deduced from TST by the following relation: AET = TST - 16 minutes

In the N_CYCLE array, the corresponding sum should be stored in the JULD_ASCENT_END variable and the JULD_ASCENT_END_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

2.2.14.1.7.8 Transmission Start Time

The hours and minutes of the TST are provided, in the technical message, by the technical parameter "heure de fin de remontée à la surface".

In the N_CYCLE array, the value should be stored in the JULD_TRANSMISSION_START variable and the JULD_TRANSMISSION_START_STATUS set to 2. If the float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

N_MEASUREMENT Array				
PROVOR floats 101003 and 101004 versions				
MC	**Float type**	**JULD**	**JULD_STATUS**	
100 DST	101003 101004	Descent Start Time without clock offset applied (2.2.14.1.7.1)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Descent Start Time with clock offset applied (2.2.14.1.7.1)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
150 FST	101003 101004	First Stabilization Time without clock offset applied (2.2.14.1.7.2)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		First Stabilization Time with clock offset applied (2.2.14.1.7.2)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
250 PST	101003 101004	Park Start Time without clock offset applied (2.2.14.1.7.3)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	

		Park Start Time with clock offset applied (2.2.14.1.7.3)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
300 PET	101003 101004	Park End Time without clock offset applied (2.2.14.1.7.4)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Park End Time with clock offset applied (2.2.14.1.7.4)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
450 DPST	101003 101004	Fill value (2.2.14.1.7.5)	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value (2.2.14.1.7.5)	9: not immediately know, but believe value can be estimated later	
MC	**Float type**	**JULD**	**JULD_STATUS**	
500 AST	101003 101004	Ascent Start Time without clock offset applied (2.2.14.1.7.6)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent Start Time with clock offset applied (2.2.14.1.7.6)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
600 AET	101003 101004	Ascent End Time without clock offset applied (2.2.14.1.7.7)	3: value is directly computed from relevant, transmitted float information	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Ascent End Time with clock offset applied (2.2.14.1.7.7)	3: value is directly computed from relevant, transmitted float information	
MC	**Float type**	**JULD**	**JULD_STATUS**	
700 TST	101003 101004	Transmission Start Time without clock offset applied (2.2.14.1.7.8)	2: value is transmitted by float	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Transmission Start Time with clock offset applied (2.2.14.1.7.8)	2: value is transmitted by float	
MC	**Float type**	**JULD**	**JULD_STATUS**	
800 TET	101003 101004	Fill value	9: not immediately know, but believe value can be estimated later	
		JULD_ADJUSTED	**JULD_ADJUSTED_STATUS**	
		Fill value	9: not immediately know, but believe value can be estimated later	

2.2.14.1.8 From day, hours and minutes to time

The hours and minutes of the event times are obtained from technical message information.

The associated day can be obtained by the following algorithms.

The day of TST is determined using FMT:

1. Convert FMT in Float Time (FMT_{FT} = FMT + *FloatClockDrift*),
2. Convert the hours and minutes of FMT_{FT} in Technical Message time (in tenths of and our after truncation) to obtain FMT_{FTTM},
3. Compare the resulting FMT_{FTTM} with TST to determine the day of TST (remembering that $FMT_{FTTM} \geq TST$).

The day of AET is determined using TST.

The day of AST is determined using AET and the assumption that: AET-AST < 24 h.

The day of DDET is determined using AST and the assumption that: AST- DDET < 24 h.

The day of PET is determined using DDET and the assumption that: DDET - PET < 24 h.

The day of DST is determined using a Reference Date (RD) which can be:

- For cycle #0: the day of the first descent (meta-data parameter needed for data decoding),
- For a given cycle #*N* (*N* > 0):
  - If cycle #*N-1* exists: RD is the LMT of the cycle #*N-1*,
  - Otherwise, RD is computed from the last received Argos **CTD** message (LMT_{CTD}): RD = LMT_{CTD} - *CycleDuration*.

The obtained RD is then used to determine the day of DST:

1. Convert RD in float time (RD_{FT} = RD + *FloatClockDrift*),
2. Convert the hours and minutes of RD_{FT} in Technical Message time (in tenths of and our after truncation) to obtain RD_{FTTM},
3. Compare the resulting RD_{FTTM} with DST to determine the day of DST (remembering that $RD_{FTTM} \leq DST$).

The day part of FST is determined using DST and the assumption that: FST-DST < 24 h.

The day part of DET is determined using FST and the assumption that: FST-DET < 24 h.

2.2.14.1.9 Technical time resolution

For PROVOR 102004 version, technical times are given in minutes.

For all other PROVOR float versions, technical times are given in tenths of an hour; moreover they are resulting from a truncation of raw measurements.

Consequently, for example, an event dated 13:36 by the float occurred in the time interval [13:36 - 13:42[.

To take this characteristic into account and to have a statistical mean estimate of the event time, in the decoding process we must add 3 minutes to DST, FST, DET, DDET, AET and TST.

Note however that PET and AST should not be modified because they are resulting from float programmed actions (at a specific hour).

2.2.15 PROVORCTS3 & Arvor Iridium

The following measurement codes are set by the Coriolis decoder for:

- Provor CTS3 Iridium floats (Firmware version 5900A04, Coriolis version 5.75, Decoder Id 214),

- Arvor Iridium floats (Firmware version 5900A04, Coriolis version 5.46, Decoder Id 217).

Argo trajectory file measurement codes (MC) for Provor CTS3 Iridium and Arvor Iridium floats.

Code (timing)	Name in float data output	Description and name of data file where this is found	Units and data profile number	JULD_STATUS
0 (launch position)	Provided by PI (from deployment team).	From Coriolis Excel deployment file provided at Coriolis DAC by	Time, latitude, longitude. Cycle #-1 (convention).	4 (value is determined by satellite)

		float PI.		
89 = DST-11 (buoyancy action)	Cycle start time (buoyancy reduction start time). From: 'Cycle start gregorian day' 'Cycle start gregorian month' 'Cycle start gregorian year' 'Cycle start time'.	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
100 (DST)	Descent to park Start Time. From: 'Descent start time' and Cycle start time (MC=89).	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
189 = DET-11 (buoyancy actions)	Buoyancy action (time and pressure) between DST and DET.	Hydraulic packets.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
150 = FST	First Stabilization Time during descent to park. From: 'Float 1st stabilisation time' and DST 'Float 1st stabilisation pressure'.	Tech packet #1.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
190 = DET-10 (relative series of measurements)	Dated levels of the descending profile. The first measurement of each packet is dated from: transmitted time + day of the first descent of the float.	Descending profile packets.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
203 (deepest measurement)	Deepest level of the descending profile.	Descending profile packets.	Time (if available), all available measurements. Cycle #N.	2 (when time is available) (value is transmitted by the float)
198 = DET-2	Max Pressure sampled during descent to park depth. From: 'Max pressure in descent to parking depth'.	Tech packet #1.	PRES. Cycle #N.	No time
Start of the drift phase				
250 (PST)	Park drift Start Time. From: 'End of descent time' and FST	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
289 = PET-11 (buoyancy actions)	Buoyancy action (time and pressure) between PST and PET.	Hydraulic packets.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
290 = PET-10 (relative series of measurements)	Measurements sampled during the drift at park depth. Times are computed: - For the first measurement of each packet: from transmitted measurement date + day of the first descent of the float - For following measurements: from drift sampling period (configuration parameter MC9).	Submerged drift packets. Parameter data packet.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float) for the time of the first measurement of each packet 1 (value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour) for the

				following measurements
300 (PET)	Park drift End Time. From: 'Descent to profile depth start time' **and DDET**.	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
297 = PET-3	Min Pressure sampled during park drift. From: 'Min pressure during drift at parking depth'.	Tech packet #1.	PRES. Cycle #N.	No time
298 = PET-2	Max Pressure sampled during park drift. From: 'Max pressure during drift at parking depth'.	Tech packet #1.	PRES. Cycle #N.	No time
301 (representative park measurement)	Averaged values of measurements sampled during the [Park start time;Park end time] time interval. REPRESENTATIVE_PARK_PRESSURE_STATUS = 1.	Submerged drift packets.	All available measurements. Cycle #N.	No time
End of drift measurements				
389 = DDET-11 (buoyancy actions)	Buoyancy action (time and pressure) between PET and DDET.	Hydraulic packets.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
398 = DDET-2	Max Pressure sampled during descent to profile depth. From: 'Max pressure during descent to profile depth'.	Tech packet #1.	PRES. Cycle #N.	No time
450 (DPST)	Deep Park Start Time. From: 'Descent to profile depth end time' **and AST.**	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
489 = AST-11 (buoyancy actions)	Buoyancy action (time and pressure) between DPST and AST.	Hydraulic packets.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)
497 = AST-3	Min Pressure sampled during deep park drift. From: 'Min Pressure during drift at profile depth'.	Tech packet #1.	PRES. Cycle #N.	No time
498 = AST-2	Max Pressure sampled during deep park drift. From: 'Max Pressure during drift at profile depth'.	Tech packet #1.	PRES. Cycle #N.	No time
Start of profile				
500 (AST)	Ascent Start Time. From: 'Ascent profile start time' **and AET.**	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
503 (deepest measurement)	Deepest level of the ascending profile.	Ascending profile packets.	Time (if available), all available measurements. Cycle #N.	2 (when time is available) (value is transmitted by the float)
589 = AET-11 (buoyancy actions)	Buoyancy action (time and pressure) between AST and AET.	Hydraulic packets.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)

590 = AET-10 (relative series of measurements)	Dated levels of the ascending profile. The first measurement of each packet is dated from: transmitted time + day of the first descent of the float.	Ascending profile packets.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
599 = AET-1 (relative single measurement)	Last pumped CTD measurement sampled during ascending profile. From: 'Sub-Surface pressure' 'Sub-Surface temperature' 'Sub-Surface salinity' 'Sub-Surface C1PHASE' etc…	Tech packet #2.	All available measurements. Cycle #N.	No time
600 (AET)	Ascent End Time. AET = **TST** – (10 minutes) – TC4 for cycles without 'Near Surface & In Air' sequence AET = **TST** – (10 minutes) – 2*MC31 – TC22 for cycles with 'Near Surface & In Air' sequence TC4, TC22 and MC31 are configuration parameters reported in parameter data packet.	Parameter data packet.	Time. Cycle #N.	2 (value is transmitted by the float)
Float is on the surface				
700 (TST)	Transmission Start Time. From: 'Ascent profile end time' **and time of first GPS fix.**	Tech packet #1.	Time. Cycle #N.	2 (value is transmitted by the float)
702 (FMT)	Earliest time of current cycle Iridium sessions.	Iridium e-mail.	Time. Cycle #N.	4 (value is determined by satellite)
703 (surface location)	All GPS fixes provided (one for each Iridium session).	Tech packet #1.	Time, latitude, longitude. Cycle #N.	4 (value is determined by satellite)
704 (LMT)	Latest time of current cycle Iridium sessions.	Iridium e-mail.	Time. Cycle #N.	4 (value is determined by satellite)
800 (TET)	Transmission End Time. Set as the cycle start time (MC=89) of the next cycle.	Tech packet #1.	Time. **Cycle #N-1**.	2 (value is transmitted by the float)
Miscellaneous				
710 (in-water samples, part of surface sequence)	For DO floats only. For cycles with 'Near Surface & In Air' sequence. Measurements sampled during the 'Near Surface' phase. Times are computed: - For the first measurement of each packet: from transmitted measurement date + day of the first descent of the float - For following measurements: from sampling period (configuration parameter MC30).	Near surface packets. Parameter data packet.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float) for the time of the first measurement of each packet 1 (value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour) for the following measurements

711 (in-air samples, part of surface sequence)	For DO floats only. For cycles with 'Near Surface & In Air' sequence. Measurements sampled during the 'In Air' phase. Times are computed: - For the first measurement of each packet: from transmitted measurement date + day of the first descent of the float - For following measurements: from sampling period (configuration parameter MC30).	In air packets. Parameter data packet.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float) for the time of the first measurement of each packet 1 (value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour) for the following measurements
901 (grounded cycle)	Grounding information. From: '1st grounding day relative to cycle start' and cycle start time (MC=89) '1st grounding hour' '1st grounding Pressure' '2nd grounding day relative to cycle start' '2nd grounding hour' '2nd grounding Pressure'.	Tech packet #2.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)

2.2.16 Arvor Argos

The following measurement codes are set by the Coriolis decoder for Arvor Argos floats (Firmware version 5605B05, Coriolis version 4.54, Decoder Id 32).

Argo trajectory file measurement codes (MC) for Arvor Argos floats.				
Code (timing)	**Name in float data output**	**Description and name of data file where this is found**	**Units and data profile number**	**JULD_STATUS**
0 (launch position)	Provided by PI (from deployment team).	From Coriolis Excel deployment file provided at Coriolis DAC by float PI.	Time, latitude, longitude. Cycle #-1 (convention).	4 (value is determined by satellite)
89 = DST-11 (buoyancy action)	Cycle start time (buoyancy reduction start time). From: 'Cycle start gregorian day' 'Cycle start gregorian month' 'Cycle start hour' + day of the first descent of the float.	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
100 (DST)	Descent to park Start Time. From: 'Descent Start Time' and Cycle start time (MC=89).	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
150 = FST	First Stabilization Time during descent to park. From: 'Float Stabilisation Time' and DST	Tech message #2.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)

	'1st Stabilisation Pressure'.			
190 = DET-10 (relative series of measurements)	Dated levels of the descending profile. The first measurement of each message is dated from: transmitted time + DST.	Descent profile data message.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
203 (deepest measurement)	Deepest level of the descending profile.	Descent profile data message.	Time (if available), all available measurements. Cycle #N.	2 (when time is available) (value is transmitted by the float)
198 = DET-2	Max Pressure sampled during descent to park depth. From: 'Max pressure in descent to Parking Depth'.	Tech message #2.	PRES. Cycle #N.	No time
Start of the drift phase				
250 (PST)	Park drift Start Time. From: 'End of descent time' and FST	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
290 = PET-10 (relative series of measurements)	Measurements sampled during the drift at park depth. Times are computed: - For the first measurement of each packet: from transmitted measurement date + transmitted measurement time + DST - For following measurements: from drift sampling period (configuration parameter MC8).	Submerged drift data message.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float) for the time of the first measurement of each packet 1 (value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour) for the following measurements
300 (PET)	Park drift End Time. From: 'Descent to Profile Depth Start time' **and DDET**.	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
297 = PET-3	Min Pressure sampled during park drift. From: 'Min Pressure in Drift'.	Tech message #2.	PRES. Cycle #N.	No time
298 = PET-2	Max Pressure sampled during park drift. From: 'Max Pressure in Drift'.	Tech message #2.	PRES. Cycle #N.	No time
301 (representative park measurement)	Averaged values of measurements sampled during the [Park start time;Park end time] time interval. REPRESENTATIVE_PARK_PRESSURE_STATUS = 1.	Submerged drift data message.	All available measurements. Cycle #N.	No time
End of drift measurements				

398 = DDET-2	Max Pressure sampled during descent to profile depth. From: 'Max Pressure in descent Profile Depth'.	Tech message #2.	PRES. Cycle #N.	No time
450 (DPST)	Deep Park Start Time. From: 'Descent to Profile Depth Stop time' **and AST.**	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
497 = AST-3	Min Pressure sampled during deep park drift. From: 'Min Pressure in Drift at Profile Depth'.	Tech message #2.	PRES. Cycle #N.	No time
498 = AST-2	Max Pressure sampled during deep park drift. From: 'Max Pressure in Drift at Profile Depth'.	Tech message #2.	PRES. Cycle #N.	No time
Start of profile				
500 (AST)	Ascent Start Time. From: 'Ascent Profile Start time' **and AET.**	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
503 (deepest measurement)	Deepest level of the ascending profile.	Ascent profile data message.	Time (if available), all available measurements. Cycle #N.	2 (when time is available) (value is transmitted by the float)
590 = AET-10 (relative series of measurements)	Dated levels of the ascending profile. The first measurement of each message is dated from: transmitted time + AST.	Ascent profile data message.	Time, all available measurements. Cycle #N.	2 (value is transmitted by the float)
600 (AET)	Ascent End Time. AET = **TST** – (14 minutes) for Arvor float AET = **TST** – (16 minutes) for Provor float.		Time. Cycle #N.	2 (value is transmitted by the float)
Float is on the surface				
700 (TST)	Transmission Start Time. From: 'Ascent Profile Stop time' **and time of first Argos message.**	Tech message #2.	Time. Cycle #N.	2 (value is transmitted by the float)
702 (FMT)	Earliest time of current cycle Argos messages.	CLS information (Argos message dates).	Time. Cycle #N.	4 (value is determined by satellite)
703 (surface location)	All Argos fixes provided.	CLS information (Argos fixes estimated by CLS).	Time, latitude, longitude, location class. Cycle #N.	4 (value is determined by satellite)
704 (LMT)	Latest time of current cycle Argos messages.	CLS information (Argos message dates).	Time. Cycle #N.	4 (value is determined by satellite)
800 (TET)	Transmission End Time. Set as the cycle start time (MC=89) of the next cycle.	Tech message #2.	Time. **Cycle #N-1**.	2 (value is transmitted by the float)
Miscellaneous				
901 (grounded cycle)	Grounding information. From: '1st Grounding day relative to cycle start' and cycle start time (MC=89) '1st Grounding Hour'	Tech packet #1.	Time, PRES. Cycle #N.	2 (value is transmitted by the float)

	'1st grounding Pressure'.			

2.2.17 SOLO floats

Real time:

All cycle times (DST, FST, DET, PST, PET, DDET, DPST, AST, AET, TST, TET) cannot be filled in real time for SOLO floats and should be filled with fill value. The corresponding status variables for these timing variables should all be a "9" for time unknown. No times should be filled from information provided in the meta files.

Delayed mode:

The SIO SOLO returns little timing information directly measured by the float. However, estimation of float surfacing and sinking from the surface, as well as most other important timing events can be successfully carried out in DMQC. If the standard amount of data is returned by the float, timing can be estimated within 10 minutes for most float events.

DMQC for timing is best done when all data has been received from the float (the float has stopped reporting). SIO has followed the following procedure for DMQC of SIO Argos float timing estimation.

1) The surface time for the SIO SOLO is FIXED. This is built into the firmware. Surface time CANNOT vary. Since the float resets it's clock each cycle, it is very unlikely clock drift will be an issue.
2) Assume that changes in cycle time are slowly evolving over the life of the float. Here 'slowly varying' is defined as slower than the float's cycle time.
3) Collect each surface interval's first and last Argos data times. These times will not have a width greater than the floats programmed surface time. If the measured surface interval does exceed the programmed surface time, the likely culpret is a 'ghost message', and the Argos dataset will need to be edited. It is best to use the times of all Argos data, and not only position times.
4) Fit a slowly varying envelope of 'surface time' width to the Argos first/last times.
5) Some 'jumps' may be necessary depending on float behavior.
6) To determine other float timing (off the surface), add forwards and backwards from the determined AET and DST times, using float parameters.

The exact method of determining the envelope is left to the PI.

The Dtraj netCDF files from SIO have been filled using the above procedure. The estimated time of AET has been transferred to the DMQC profile netCDF file variable JULD. When this is done JULD_QC has been changed to '8', meaning interpolated.

2.2.18 SOLO-II floats

Solo-II floats are all newer, Iridium floats and as such, their cycle timing variables follow a different chronological order than many of the other floats included in this document. J. Gilson has created a

table from which one can match all the desired measurement codes with what the SOLO-II float measures, referring back to the float's documentation. The documentation is all available on the SIO-Argo website (http://sio-argo.ucsd.edu/manuals.html).

With SOLO-II float version 2.0 and later, data is returned which closely labels the data sent by the float to the Measurement Code discussed here. In the below table the code retuned by the float is described as 'Variable Code'. The Variable Code value does not equate to the Measurement Code value.

Argo program measurement codes (MC)				
Code (timing)	**SOLO II Variable**	**Description**	**Units**	**JULD_STATUS**
0	Cy 0: GPS ID=0x00 (Variable Code=1)	GPS fix from surfacing after short ~100dbar test dive	Time,position	1
100 (DST)	Cy>0: Fall ID=0x40 (Variable Code=1)	Typically, thefirst T,P pair [taken as valve opened to leave surface]	Time,P(0.04db)	2
199	Cy=0: Eng ID=0xe0 (Variable Code=7)	P,T,S triplet taken when float realizes it is under the surface and pumps to return to the surface (Eng ID=0xe0 bytes 39-47)	P(0.04db),T(0.001°C), S(0.001psu)	2
139/140	Cy>0: Fall ID=0x40	All pre-FST T,P Fall pairs not assigned to other MC (139 used for buoyancy adjustments)	Time,P(0.04db)	2
150 (FST)	Cy>0: Fall ID=0x40 (Variable Code=2)	T,P Fall pair ~ 100dbar	Time,P(0.04db)	2
189/190	Cy>0:Fall ID=0x40	All other pre-DET T,P Fall pairs (189 used for buoyancy adjustments)	Time,P(0.04db)	2
200 (DET)	Cy=0:Rise ID=0x50	Typically, Deepest T,P pair	Time,P(0.04db)	2
	Cy>0: Fall ID=0x40	Choice of T,P pair that is first within 3% of pressure at beginning of drift (see Eng ID=0xe2 bytes 63-65)	Time,P(0.04db)	2
n=239/240	Cy>0: Fall ID=0x40	All post DET T,P pairs. MC239 are used for buoyancy adjustments. If n is the number of stabilizations (see Argo ID=0xf0), the T,P n+1 from end of Fall record is a stabilization. Each later T,P pair excluding the last will be an additional stabilization. Note: for some floats there are stabilizations during drift.	Time,P(0.04db)	2
if there is a drift phase (drift pressure defined) (common to cycles > 1)				
250 (PST)	Cy>0: Fall ID=0x40 (Variable Code=4)	Last T,P Fall pair	Time,P(0.04db)	2
296	Cy>0: Eng ID=0xe2	Drift broken into two averaged halves. Stored in Eng ID=0xe2 bytes 63-80; Time estimated from the last Fall ID=0x40 T,P pair [note: not DET] and first Rise ID=0x50 T,P pair	P(0.04db),T(0.001°C), S(0.001psu)	2
290	Cy>0 with park phase Drift ID 0x98	The SOLOII can return the raw drift measurements. Time is not returned, but can be estimated within a few	P(0.04db),T(0.001°C), S(0.001psu)	3

		seconds.		
300 (PET)	Cy>0: Rise ID=0x50	First T,P Rise pair [taken as valve opened]	Time,P(0.04db)	2
301		Best estimate of drift depth (average of two averaged halves)	Pressure	1
Endif if there is a deep dive (profile pressure > drift pressure and drift pressure defined)				
389/390	Cy>1: Rise ID=0x50	All pre-DDET T,P Rise pairs (389 indicates time of buoyancy adjustedment)	Time,P(0.04db)	2
400 (DDET)	Cy>1: Rise ID=0x50	DDET is determined by a) 2nd derivative of Rise pair series or b) within 3% of profile depth (see Eng ID=0xe2 bytes 39-41).	Time,P(0.04db)	2
489/490	Cy>1: Rise ID=0x50	All post-DDET/pre-AST T,P Rise pairs (489 indicates time of buoyancy adjustment)	Time,P(0.04db)	2
500 (AST)	Cy>1: Rise ID=0x50; Eng ID=0xe2 (Typically Variable Code =7)	AST is determined by 2nd derivative of Rise pair series.	Time,P(0.04db); P(0.04db),T(0.001°C), S(0.001psu)	2
Else				
500 (AST)	Cy=0: Rise ID=0x50; (Variable Code=7) Cy=1 Eng ID=0xe2 (Typically Variable Code =7)	First T,P Rise pair [taken as valve opened] AST is determined by 2nd derivative of Rise pair series	Time,P(0.04db); P(0.04db),T(0.001°C), S(0.001psu)	2 2
Endif				
589/590	Cy>=0: Rise ID=0x50	All T,P Rise pairs post AST excluding last or last two. 589 indicates buoyancy adjustment.	Time,P(0.04db)	2
599	Cy=0: Eng ID=0xe0 Cy>0: Eng ID=0xe2	last P,T,S triplet taken before turning off CTD (Eng ID=0xe0 bytes 48-56) last P,T,S triplet taken before turning off CTD (Eng ID=0xe2 bytes 45-50)	P(0.04db),T(0.001°C), S(0.001psu) P(0.04db),T(0.001°C), S(0.001psu)	2 2
600 (AET)	Cy>-1: Rise ID=0x50 (Variable Code=8)	Last or 2nd to last T,P Rise pair	Time,P(0.04db)	2
703	Cy=0: GPS ID=0x00 Cy>0: GPS ID=0x02	GPS Fix GPS Fix	Time, Position Time, Position	2 2
700 (TST) 702 (FMT)	 Time in SBD email	TST is not recorded by the float, but it is within a minute of the first message Time of first SBD message	Time Time	1 4
704 (LMT) 800 (TET)	Time in SBD email	Time of last SBD message TET is not recorded by the float, but it is within a few seconds of the last message	Time Time	4 3
703	Cy>0: GPS ID=0x01	GPS Fix: May be multiple GPS fixes, depending on float	Time, Position	2

		settings		

2.3 Guidelines for Argos message selection

2.3.1 Argos float message selection

Ideally, every DAC should use the same method for Argos message selection for each float type. Some floats are transmitting a CRC (Cyclic Redundancy Check) done on board the float and others are not. Additionally, not all CRC have the same reliability. Recommendations were issued at ADMT 10, but inconsistencies still exist between DACs.

Each float types message selection strategy will be listed below:

Argos message selection done at Coriolis for PROVOR/ARVOR

Technical message selection

1. If only one technical message is received with a good CRC, use it,
2. If more than one technical message is received, all with good CRCs, use the"first received one"
3. If no technical message is received with a good CRC, no technical message is used. In this case, times provided by the float are missing and, consequently, the order of the drift CTD measurements cannot be determined.

CTD data message selection

Received messages are processed by type (type 4: "descent profile CTD message", type 5: "submerged drift CTD message" and type 6: "ascent profile CTD message").

For each type, the Id of the received message is computed.

- For type 4 or type 6 messages, the Id is defined by the date and the pressure of the first CTD measurement of the message,
- For type 5 messages, the Id is defined by the date and the time of the first CTD measurement of the message.

The selection process must lead to (at most) one message for a given Id.

For a given type, all messages of a given Id are processed:

1. If only one message is received with a good CRC, use it,
2. If more than one message is received all with good CRCs, use the "first received one",
3. If no message is received with a good CRC:
    a. If 1 or 2 copies of the message has been received, no message is used for this Id,
    b. If more than 2 copies of the message have been received:
        i. If an even number of copies of the message have been received, reject the "first received one",
        ii. The possibly emitted message is computed from received copies (each bit of the message is defined by selecting the "most redundant" received one),
        iii. A CRC check is done on this "reconstructed" message:
            1. If it succeeds, use this "reconstructed" message,
            2. If it fails, no message is used for this Id.

2.4 Sensor measurements

2.4.1 Sensor measurements sampled during the drift phase at parking depth

2.4.1.1 APEX and Navis floats

2.4.1.1.1 CTD measurement sampled at the end of the drift phase at parking depth

APEX and Navis float versions provide a CTD measurement (P, T and S) sampled at the end of the drift phase at parking depth, generally called park (or bottom) measurement. The corresponding time of the measurement is provided by Iridium float versions only.

This measurement should be associated to PET (MC 300) or AST (MC 500) if theoretical PARKING and PROFILE depths are equal for the corresponding cycle (see §2.2.4.5).

2.4.1.1.2 CTD measurements regularly sampled during the drift phase at parking depth

Some APEX and Navis float versions provide CTD measurements (only P and T generally) regularly sampled during the drift phase at parking depth. The corresponding times of the measurements are provided by Iridium float versions only.

For regularly sampled CTD measurements, use MC minus 10. Usually this will be 290 because the float is transitioning towards PET which is 300. If the float is transitioning towards a different MC, subtract four from that MC.

For averaged sampled CTD measurements, use MC minus 4. Usually this will be 296 because the float is transitioning towards PET which is 300. If the float is transitioning towards a different MC, subtract four from that MC.

For Argos float versions these measurements need to be dated in a post-processing procedure. For that we need to know additional meta-data parameters.

The following six sampled strategies can be encountered.

2.4.1.1.2.1 Normal float behavior

For many of the APEX and Navis floats, the first drift measurement is sampled sometime between 4-8 hours after DST and the following ones with a programmed theoretical period.

Thus, for these floats, to compute the CTD measurement times we must first know:

- The DST (see §2.2.4.2),
- The theoretical period of the drift measurements.

2.4.1.1.2.2 Floats with daily CTD measurements

Some APEX floats provide a daily measurement, the first one sampled 24 hours after DST.

2.4.1.1.2.3 Floats providing only averaged values

Some floats provide *N* averages of hourly sampled CTD measurements. *N* is a programmed meta-data parameter.

The times of the averaged value should be computed to be regularly set between DET and PET.

The time of the average #*i* should be: time(*i*) = DET + (2**i* - 1)*(PET - DET)/(2**N*)

2.4.1.1.2.4 Isopycnal floats behavior

Isopycnal APEX floats generally provide two sets of CTD measurements.

The first one corresponds to the stabilization at the target sigma-theta value.

The first CTD measurement is sampled 6 hours after DST and the following ones with a 1.5 hour period.

See §2.4.3.3 for the storage of these data.

The second set of CTD measurements corresponds to the drift at the target sigma-theta value.

The first CTD measurement is sampled 6 hours after the last measurement of the first set and the following ones with a 6 hour period.

These are series of measurements, so the MC should be MC-10.

2.4.1.1.2.5 Old versions of isopycnal floats

Some (old) versions of isopycnal floats provide CTD measurement sampled at isopycnal depth but we do not know how to compute their corresponding times.

2.4.1.1.2.6 RAFOS floats behavior

RAFOS floats generally provide a daily CTD measurement sampled at the end of the last listening window of the day.

Thus, for these floats, to compute the CTD measurement times we must first know:

- The hour of the last listening window,
- The length of the listening windows.

The decoded and dated (if possible) CTD measurements should be stored in the N_MEASUREMENT arrays with:

- JULD set to the computed times,
- JULD_QC: set to 0,
- JULD_STATUS set to 3 (computed directly from information sent by the float)
- CYCLE_NUMBER set to corresponding cycle number,
- MEASUREMENT_CODE set to 290,
- <PARAM> set to decoded CTD values,
- <PARAM>_QCs set to 0,
- All other variables set to _FillValue.

For Iridium floats, the decoded times should be corrected for clock offset and stored in the JULD_ADJUSTED variable. CLOCK_OFFSET should also be filled in the N_CYCLE array.

For Argos floats, the times are computed from DST, thus clock offset is already taken into account. The times should still be stored in JULD_ADJUSTED since clock offset has been taken into account.

For Argos floats, we must consider missing Argos messages to correctly set CTD measurement dates (i.e. we must take into account the missing CTD measurements, due to not received data, to correctly apply the periodicity of the dates).

2.4.1.1.3 Minimum and maximum values of pressure during drift

Some APEX floats, provide the minimum and maximum values of the pressure regularly sampled during the drift at PARKING depth.

These values should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 297 or 298.

2.4.1.1.4 PARAM at min/max of another PARAM

Some APEX float versions record the pressure at max temperature during drift and pressure at minimum temperature during drift. In order to not assign too many float specific MCs, new relative specific MCs were added to address this general situation. There are:

MC minus 12	Any supporting measurements for the maximum value (MC minus 2)
MC minus 13	Any supporting measurements for the minimum value (MC minus 3)
MC minus 14	Any supporting measurements for the averaged value (MC minus 4)
MC minus 15	Any supporting measurements for the median value (MC minus 5)

So, for the pressure at the maximum temperature during drift, use MC 288. For pressure at the minimum temperature during drift, use MC 287.

2.4.1.1.5 BGC measurements regularly sampled during the drift phase at parking depth

A growing subset of APEX and Navis BGC-Argo floats are now sampling various biogeocemical parameters during the drift phase along with PRES and TEMP. For APEX Apf11 floats, this can include bio-optical measurements taken from the FLBB sensor (ie CHLA and BBP700). For Navis BGC-Argo floats, those equipped with oxygen and pH sensors can sample DOXY and PH_IN_SITU_TOTAL during drift (as well as PSAL). Additionally, a smaller number of older BGC Navis have drift sampling turned on for CHLA, BBP700 and CDOM from the MCOMS sensor. Considering the potential science applications for bio-optical data sampled during the drift phase (ie particle export studies), the sampling of bio-optical measurements at the park depth may become more prominent in the future.

Similar to CTD measurements taken during drift, the appropriate measurement code for these types of examples is usually 290 (MC minus 10) because the float is transitioning towards PET which is 300. Sample frequency can vary during the drift phase, but most APEX sample at an hourly rate during drift (for both core and BGC measurements), while many Navis sample every six hours during drift.

2.4.1.2 PROVOR floats

2.4.1.2.1 CTD measurements regularly sampled during the drift phase at parking depth

All PROVOR floats have the capability to achieve and provide CTD measurements (P, T and S) regularly sampled during the drift phase at parking depth but this capability must be enabled by the operator in the programmed float mission. Regularly sampled CTD measurements are represented by the MC towards which the float is transitioning - 10. Usually, the float is transitioning towards PET or 300, making the MC code 290.

For PROVOR 100001 version, we have no information, in the transmitted data, about the drift measurement times.

For PROVOR 100006, 100005, 100004, 100008 and 100003 versions, the time of the first drift measurement is provided in the technical message as well as the drift data sampling period.

For all other float versions, the time of the first drift measurement of each Argos message (or Iridium packet) is transmitted in the data message. The day number of this time is relative to the day of the first descent (cycle #0).

Moreover, for some float versions, to minimize the impact of the loss of a drift CTD message, drift measurements are transmitted using an interleaving scheme:

- Measurements #1, #3, #5, #7, … are transmitted in a first message,
- Measurements #2, #4, #6, #8, … are transmitted in a second message.

In this case, it is very important to determine the **theoretical** time of the first drift measurement (particularly when this measurement is not received from the float).

This time depends on float version.

2.4.1.2.1.1 Drift measurement times determination for PROVOR 101011, 102002, 101012, 101014, 101015, 102003, 101013 and 100001 versions

For these float versions, measurements are done at round hours.

The theoretical First Drift Measurement Time (FDMT) is relative to the hour of the DET (which must be rounded down).

If DET is given as a Julian day.

If DSP is the Drift Sampling Period (in hours).

Then FDMT = floor(DET*24)/24 + DSP/24.

For PROVOR 101011, 101012, 101014, 101015 and 101013 versions, drift measurements are transmitted using an interleaving scheme.

This is not the case for PROVOR 102002, 102003 and 100001 versions.

2.4.1.2.1.2 Drift measurement times determination for PROVOR 101009, 101006, 101008, 101007, 101010, 101002, 101005, 101003, 101004 and 100002 versions

For these float versions, the theoretical First Drift Measurement Time (FDMT) is relative to the day of DET.

If DET is given in Gregorian time as "MM/DD/YYYY hh:mm:ss".

If DSP is the Drift Sampling Period (in hours).

Then FDMT = "MM/DD/YYYY 00:00:00" + *N**DSP/24

where *N* is the minimum integer value for which

"MM/DD/YYYY 00:00:00" + *N**DSP/24 > "MM/DD/YYYY hh:mm:ss"

For PROVOR 101009, 101006, 101008, 101007, 101010, 101002, 101005 and 100002 versions, drift measurements are transmitted using an interleaving scheme.

This is not the case for PROVOR 101003 and 101004 versions.

2.4.1.2.2 Minimum and maximum values of pressure during drift

Some PROVOR float versions provide the minimum and maximum values of the pressure regularly sampled during the drift at PARKING depth.

These values should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 297 or 298.

For PROVOR floats these values are given **in bars**, the pressure resolution should be set accordingly (see §1.2.2).

2.4.1.3 NINJA floats

2.4.1.3.1 CTD measurements for NINJA 300001, 300002 and 300003 versions

2.4.1.3.1.1 CTD measurement sampled at the beginning and end of the drift phase at parking depth

These NINJA versions provide a pressure measurement sampled at the beginning and at the end of the parking phase.

The first pressure should be stored in the N_MEASUREMENT array in association with PST (thus with a MEASUREMENT_CODE set to 250).

The second pressure should be stored in the N_MEASUREMENT with a MEASUREMENT_CODE set to 300 for PET.

The time associated with these pressure measurements cannot be estimated, so JULD should be fill value and JULD_STATUS should be ‘9’.

2.4.1.3.1.2 CTD measurements regularly sampled during the drift phase at parking depth

These NINJA versions provide pressure measurements daily sampled during the drift phase at parking depth.

These pressure measurements should be stored in the N_MEASUREMENT with a MEASUREMENT_CODE set to 290 if the float is transitioning towards PET. Even though we don't know how to compute the time for these measurements, they can be included in the N_MEASUREMENT array with the appropriate MC code.

The time associated with these pressure measurements cannot be estimated, so JULD should be fill value and JULD_STATUS should be ‘9’.

2.4.1.3.1.3. CTD measurements for NINJA 300004 version

These NINJA versions provide CTD measurements (P, T and S) sampled during the drift phase at parking depth.

These CTD measurements should be stored in the N_MEASUREMENT with a MEASUREMENT_CODE set to 290 if transitioning towards PET.

The time associated with these pressure measurements cannot be estimated, so JULD should be fill value and JULD_STATUS should be ‘9’.

2.4.1.4 SOLO-II and SOLO floats

Measurement code	SOLO II Variable	Description	Units
296	Cy>0: Eng ID=0xe2	The drift is broken into two averaged halves. Stored in Eng ID=0xe2 bytes 67-78; Time estimated from the last Fall ID=0x40 T,P pair [note: not DET] and first Rise ID=0x50 T,P pair	P(0.04db),T(0.001°C), S(0.001psu)

SOLO floats perform the same drift measurements as SOLO-II floats. Use MC=296 for the two drift measurements reported by the SOLO.

2.4.1.5 NOVA floats

Measurement code	NOVA Variable	Description	Units
290	Series of pressure Series of temperature Series of salinity	A series of CTD measurements taken during drift at user specified times	Pressure Temp Salinity
297	Minimum pressure	Minimum pressure recorded during drift phase	Pressure
298	Maximum pressure	Maximum pressure recorded during drift phase	Pressure

NOVA has PARAMETER 5 PARKING SAMPLING PERIOD which ranges from 0 – 24 hours and is user configurable. It comes set to 24, which means a CTD measurement is recorded and transmitted per time period per cycle (normally 9 measurements).

2.4.2 REPRESENTATIVE_PARK_PRESSURE

The REPRESENTATIVE_PARK_PRESSURE and REPRESENTATIVE_PARK_PRESSURE_STATUS variables in the N_CYCLE array are to include one pressure value for the drift period of the current cycle. These values can be filled in real time, but should be confirmed/updated in delayed mode. The STATUS flags are clear as to how the value is calculated and should be done for each float. Flag '1' involves finding a weighted average of regularly sampled pressures during drift (MC = 290). Flag '2' is the mean value directly provided by the floats of pressure measurements regularly sampled during drift (MC = 296). Flag '3' is the median value, directly provided by the float of pressure measurement regularly sampled during drift. Flag '4' is the

pressure measured at PET. Flag '5' is the average of the min and max pressure measurements sampled during drift (MCs = 297 and 298). Flag '6' and '7' is the PARKING_PRESSURE meta-data value for floats that for some reason either missed the pressure measurement ('6') or do not make pressure measurements ('7') during drift. Flag '8' is the value estimated in Delayed Mode from float behavior/data. This may include profile limits, data from other cycles, temperature at drift, etc.

As an example here is the algorithm used to compute the RPP for ANDRO files:

The RPP is computed for each cycle and depends on the measurements sampled during the drift phase at parking depth. We start from the most reliable RPP (STEP #1) and, if the needed data are not present, we try the next step and so on until the last step (STEP #).

STEP #1:

If we have isopycnal pre-stabilization CTD measurements and CTD measurement regularly sampled during the drift phase at parking depth: the RPP is the average value weighted by the time (thus with a weight of 1.5 for the pre-stabilization CTD measurements and 6 for the other ones).

STEP #2:

If we have CTD measurement regularly sampled during the drift phase at parking depth: the RPP is the average value of these measurements (note that the measurement done at PET which is generally also present is not used in this case).

STEP #3:

If we directly have the mean value of (generally hourly) regularly sampled CTD measurements: the RPP is this mean value (note that the measurement done at PET which is generally also present is not used in this case).

STEP #4:

If we directly have the median value of regularly sampled CTD measurements: the RPP is this median value.

STEP #5:

If we have a CTD measurement done at PET: the RPP is this measurement.

STEP #6:

If we have the Min and Max pressure values of the measurements done during the drift phase at parking depth: the RPP is the mean of this two values.

STEP #7:

If we have multiple profiles during the cycle: the RPP is the average of mean and max profile values, weighted by the time spent in profile and in drift between profiles (this case is specific to APEX BOUNCE cycles).

STEP #8:

If we have the PARKING_PRESSURE meta-data value: the RPP is this value.

2.4.3 Ascending and descending measurements

2.4.3.1 Stabilization CTD measurements

2.4.3.1.1 PROVOR floats

All PROVOR versions provide the First Stabilization Time but only some of them provide the associated pressure (**in bars**).

Detailed information can be found in paragraphs 2.2.14.1.1.2, 2.2.14.1.2.2, 2.2.14.1.3.2, 2.2.14.1.4.2, 2.2.14.1.5.2, 2.2.14.1.6.2 and 2.2.14.1.7.2.

2.4.3.1.2 NINJA floats

Some NINJA versions provide three First Stabilization Times with the associated pressures (see §2.2.11.1.1.2).

The **three** First Stabilization Times and pressures should be stored in the N_MEASUREMENT arrays with the MEASUREMENT_CODE set to 150 for the first stabilization and 189 for the second and third stabilizations.

2.4.3.2 APEX descending pressure marks

Some APEX float versions (see ANNEX H: Cookbook entry point) provide pressure marks hourly sampled during descent from the surface to the PARKING depth. These can be assigned MC = 190.

2.4.3.2.1 APEX Argos floats

For APEX Argos floats, these descending pressure marks are provided in the Auxiliary Engineering Data.

The first measurement is done at the end of the piston retraction, thus at DST, the following are done with a 1 hour period.

Note also that the descending pressure marks are **in bars**.

For APF9 APEX floats, it appears to be DST plus one hour. Here is a real example:

(Jun 12 2012 22:19:36, 15 sec) DescentInit() Surface pressure: -0.1dbar. IER: 0x00

(Jun 12 2012 22:19:41, 20 sec) PistonMoveAbsWTO() 226->066 225 224 223 222 221 220 219 218 217 216 215 214 213 212 211 210 209 208 207 206 205 204 203 202 201 200 199 198 197 196 195 194 193 192 191 190 189 188 187 186 185 184 183 182 181 180 179 178 177 176 175 174 173 172 171 170 169 168 167 166 165 164 163 162 161 160 159 158 157 156 155 154 153 152 151 150 149 148 147 146 145 144 143 142 141 140 139 138 137 136 135 134 133 132 131 130 129 128 127 126 125 124 123 122 121 120 119 118 117 116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 099 098 097 096 095 094 093 092 091 090 089 088 087 086 085 084 083 082 081 080 079 078 077 076 075 074 073 072 071 070 069 068 067 066 [873sec, 13.4Volts, 0.314Amps, CPT:873sec]

(Jun 12 2012 22:34:16, 895 sec) Descent() Pressure: 3.5

(Jun 12 2012 22:34:17, 896 sec) SetAlarm() Success: itimer=896 sec, ialarm=3600 sec

(Jun 12 2012 23:19:23, 3602 sec) Apf9Init() Wake-up initiated by interval-timer alarm signal.

(Jun 12 2012 23:19:26, 3604 sec) Descent() Pressure: 282.1

The pressure mark times are computed from DST (already corrected for clock offset); thus they do not need to be corrected for clock offset but the information should be set.

The descending pressure marks are not always transmitted (depending on the remaining space in the last Argos message) but if received, the decoded and dated pressures should be stored in the N_MEASUREMENT arrays with:

- JULD or JULD_ADJUSTED, if clock offset has been applied, set to the computed times,
- JULD_QC: set to 0,
- JULD_STATUS set to 2
- CYCLE_NUMBER set to corresponding cycle number,
- MEASUREMENT_CODE set to 190,
- PRES set to decoded pressure mark values,
- PRES:resolution should be deleted (see §1.2.2)
- PRES_QCs set to 0,
- All other variables set to _FillValue.

2.4.3.2.2 APEX Iridium floats

For APEX Iridium floats, the descending pressure marks and the associated times can be retrieved from log file as the pressure values and the times of the events:

Descent() Pressure: XX.X

Descent() Pressure: YYY.Y

…

Descent() Pressure: ZZZ.Z

associated with the concerned cycle.

The pressure mark times should be first corrected from clock offset, then stored in the N_MEASUREMENT arrays with:

- JULD set to the retrieved dates or JULD_ADJUSTED if clock offset has been applied,
- JULD_QC/JULD_ADJUSTED_QC: set to 0,
- JULD_STATUS/JULD_ADJUSTED_STATUS set to 2,
- CYCLE_NUMBER set to corresponding cycle number,
- MEASUREMENT_CODE set to 190,
- PRES set to retrieved pressure mark values,
- PRES:resolution should be deleted (see §1.2.2)
- PRES_QCs set to 0,
- All other variables set to _FillValue.

2.4.3.3 APEX isopycnal pre-stabilization measurements

Isopycnal APEX floats generally provide a set of CTD measurements sampled just before the stabilization at the target sigma-theta value.

The first CTD measurement is sampled 6 hours after DST and the following ones with a 1.5 hour period.

The decoded and dated CTD measurements should be stored in the N_MEASUREMENT arrays with:

- JULD set to the computed times of JULD_ADJUSTED if clock offset has been applied,
- JULD_QC/JULD_ADJUSTED_QC: set to 0,
- JULD_STATUS/JULD_ADJUSTED_STATUS set to 2,
- CYCLE_NUMBER set to corresponding cycle number,
- MEASUREMENT_CODE set to 189,
- <PARAM> set to decoded CTD values,
- <PARAM>_QCs set to 0,
- All other variables set to _FillValue.

For Iridium floats, the decoded times should be corrected for clock offset.

For Argos floats, the dates are computed from DST, thus clock offset is already taken into account.

For Argos floats, we must consider missing Argos messages to correctly set CTD measurement times (i.e. we must take into account the missing CTD measurements, due to not received data, to correctly apply the periodicity of the times).

2.4.3.4 Dated bins of descending/ascending profiles

2.4.3.4.1 PROVOR floats

PROVOR float transmits profile data through specific Argos messages (or Iridium packets). Only the first CTD measurement of each Argos message is transmitted with its associated time. Thus, after decoding, because of the interleaving scheme used to pack the profile bin measurements, some profiles bins (around one over four or five, depending of the PROVOR version) are dated.

The dated bins of the descending (or ascending) profiles should be stored in the N_MEASUREMENT arrays (CTD measurements and associated times) with a MEASUREMENT_CODE set to 190 or 590.

2.4.3.4.2 NINJA floats

Some NINJA versions provide time information recorded during the ascent phase.

The transmitted data consist of the elapsed time for each vertical slice of ascent (from the max pressure to 2000 dbar for the first slice; and for each 100 dbar thick other slices until the surface).

These times can be used to (roughly) compute the time of one bin each 100 dbar.

The obtained dated bins of the ascending profiles should be stored in the N_MEASUREMENT arrays (CTD measurements and associated time) with a MEASUREMENT_CODE set to 590.

2.4.3.4.3 SOLO-II floats

SOLO-II floats provide timed pressure measurements on descent and ascent. The dated pressures of both the descending and ascending profiles should be stored in the N_MEASUREMENT arrays (CTD measurements and associated times) with a measurement code set to 190 for descending and 590 for ascending.

2.4.3.5 Deepest descending/ascending CTD measurements

The profile CTD measurements are stored in the PROF file. However, a copy of the deepest bin CTD measurements should be stored in the TRAJ file.

The deepest bin CTD measurements of a descending profile should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 203.

The deepest bin CTD measurements of an ascending profile should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 503.

2.4.3.6 Max pressure during descent to PARKING depth

Some PROVOR float versions provide the maximum pressure experienced by the float during the descent to PARKING depth. This value is given **in bars**, the pressure resolution should be set accordingly (see §1.2.2).

This pressure should be stored in the N_MEASUREMENT arrays with the MEASUREMENT_CODE set to 198.

2.4.3.7 Min/max pressure during drift at PROFILE depth

Some PROVOR float versions provide the minimum and maximum values of the pressure regularly sampled during the drift at PROFILE depth.

These values should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 497 and 498.

These values are given **in bars**, the pressure resolution should be set accordingly (see §1.2.2).

2.4.3.8 Max pressure during descent to PROFILE depth

Some PROVOR float versions provide the maximum pressure experienced by the float during the descent to PROFILE depth. This value is given **in bars**, the pressure resolution should be set accordingly (see §1.2.2).

This pressure should be stored in the N_MEASUREMENT arrays with the MEASUREMENT_CODE set to 398.

2.4.3.9 Max pressure of the cycle

Some NINJA float versions provide the maximum pressure experienced by the float during the cycle.

This pressure should be stored in the N_MEASUREMENT arrays with the MEASUREMENT_CODE set to 498.

2.4.3.10 PROVOR Iridium spy data

Some PROVOR Iridium versions provide pressure values (**in bars**) versus time sampled during the three vertical phases of the cycle (from surface to PARKING depth, from PARKING depth to PROFILE depth and from PROFILE depth to surface).

These dated pressure measurements should be stored in the N_MEASUREMENT arrays with a MEASUREMENT_CODE set to 189 or 389 or 589 depending on the phase.

2.4.4 Surface measurements

Some floats carry sensors that can take samples during the surface interval. These surface measurements should be recorded in the trajectory file. In general, there are two types of surface measurements.

(a). Discrete surface samples. These are denoted by MC = X – 1, where X is the primary measurement code that the float cycle is transitioning towards. For example, if the discrete surface samples are taken during the transmission period before TET (MC800), then these measurements are stored with MC = 800 – 1 = 799. Note that there can be multiple discrete samples during a transmission period.

(b). Oxygen AirCal Sequence. For dissolved oxygen sensors, the SCOR Working Group 142 recommends that a sequence of measurements be taken at the surface just before and just after bladder inflation. The samples collected just before bladder inflation are intended to collect samples in-water very near the surface. The samples collected just after bladder inflation are intended to represent samples collected in-air. Together, they should provide sufficient quality control data for calibration of the dissolved oxygen sensors. For this purpose, measurements from the AirCal Sequence are stored together in the trajectory file with two distinct MCs. Measurements from the in-water portion of the AirCal Sequence are stored with MC = X + 10, while measurements from the in-air portion are stored with MC = X + 11, where X is the primary measurement code that the float cycle is transitioning towards. For example, if the AirCal Sequence occurs after AET(MC600) and before TST(MC700), then the measurements from the AirCal Sequence are stored with MC = 700 + 10 = 710 for the in-water portion, and 700 + 11 = 711 for the in-air portion.

Note:

- X is always the primary measurement code that the float cycle is transitioning towards. For surface measurements, X can be 600 (AET), 700 (TST), or 800 (TET).
- The surface measurements in (a) and (b) are stored in the trajectory file with different MCs because they are two different events, and can represent different information or quality level.

The following are some examples of floats managed by the AOML and Coriolis DACs.

2.4.4.1 NKE floats

2.4.4.1.1 NKE Oxygen only floats without the “Near Surface & In Air” feature

For these floats no additional information is stored in the TRAJ file.

2.4.4.1.2 NKE Oxygen only floats with the “Near Surface & In Air” feature

The “Near Surface & In Air” feature is activated on some cycles (which repetition rate is set by the *CONFIG_InAirMeasurementPeriodicity_NUMBER* configuration parameter).

When this feature is not activated, no additional information is stored in the TRAJ file.

When this feature is activated, additional information is stored in the TRAJ file:

- Concerning “Near surface” data set:
  - PPOX_DOXY parameter is computed and added to the data (with oxygen intermediated parameters);
  - As these measurements are sampled after Ascent End Time (10 dbar) (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700+10=710.
- Concerning “In air” data set:
  - PPOX_DOXY parameter is computed and added to the data (with oxygen intermediated parameters);

- As these measurements are sampled after Ascent End Time (10 dbar) (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700+11=711.

2.4.4.1.3 NKE BGC floats

2.4.4.1.3.1 For Provor CTS4 floats

Provor CTS4 floats do not have any specific “Near surface” or “In air” measurement phase. However, as they have the ability to sample “raw data” (i.e. data transmitted without any post-processing decimation or averaging), we can select the “Near surface” and “In air” measurements from pressure values.

The algorithm has been specified by Henry BITTIG (henry.bittig@io-warnemuende.de) and it is the following:

1. The “In air” DO measurements are selected as

PRES(_ADJUSTED) + *CONFIG_OptodeVerticalPressureOffset_dbar* ≤ -0.1 dbar

2. If “In air” DO measurements have been sampled, a “Near surface” DO measurement of the descending (resp. ascending) profile is selected as the “last DO measurement” (resp. ”first DO measurement”) for witch

PRES(_ADJUSTED) + *CONFIG_OptodeVerticalPressureOffset_dbar* ≥ 0.3 dbar

CONFIG_OptodeVerticalPressureOffset_dbar is the configuration parameter that stores the vertical offset between the CTD and the Optode.

When “In air” measurements have been sampled, additional information is stored in the TRAJ file:

- Concerning “Near surface” measurement:
  - PPOX_DOXY parameter is computed and added to the data (with oxygen intermediated parameters);
  - It is stored in the TRAJ file with the MC = 100+10 = 110 for descending profile and with the MC = 700+10 = 710 for ascending profile.
- Concerning “In air” data set:
  - PPOX_DOXY parameter is computed and added to the data (with oxygen intermediated parameters);
  - They are stored in the TRAJ file with the MC = 100+11 = 111 for descending profile and with the MC = 700+11 = 711 for ascending profile.

2.4.4.1.3.2 For Provor CTS5 floats

Provor CTS5 floats provide “In air” measurements.

For these data, additional information is stored in the TRAJ file:

- PPOX_DOXY parameter is computed and added to these measurements (with oxygen intermediated parameters);
- As these measurements are sampled after Ascent End Time (10 dbar) (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700+11=711.

Provor CTS5 floats also provide temperature measurements sampled during the transmission phase. They are stored in the TRAJ file with the MC=800-1=799.

2.4.4.2 Apex floats

2.4.4.2.1 For Apex Argos floats

Some Apex APF9 Argos float versions provide surface measurements sampled by DO, FLNTU or FLBB sensors.

In some of the concerned float manuals we can read that "A new optode/FLNTU surface measurement is made each time a new message block is transmitted." (extract from float version 082807 manual).

Thus in that case the reference MC to use is Transmission End Time (MC=800).

For these floats, additional information is stored in the TRAJ file:

- PPOX_DOXY parameter is computed and added to these measurements (with oxygen intermediated parameters);
- The reference MC used is MC=800 and they are stored in the TRAJ file with the MC=800-1=799.

2.4.4.2.2 For Apex Iridium floats

2.4.4.2.2.1 Apex APF9 floats

Some Apex Iridium float versions provide surface measurements sampled by DO or FLBB sensors.

In these float version manuals, one can find the following information:

"Usually, only one telemetry cycle is required to upload the data to the remote host computer. However, sometimes the iridium connection is broken or the quality of the connection is so poor that the float will abort the telemetry attempt, wait a few minutes, and then try again. Data blocks 4 and 5 will be repeated for each telemetry cycle of a given profile."

Please note that surface measurements are in Data block 4.

In the .log files we also learn that the surface measurement is done just before GPS location determination and could be repeated when something failed during the transmission phase.

Thus, in that case the reference MC to use is Transmission Start Time (MC=700).

For these floats, the following information is stored in the TRAJ file:

- PPOX_DOXY parameter is computed and added to these measurements (with oxygen intermediated parameters);
- The reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700-1=699.

2.4.4.2.2.2 Apex APF11 floats

Apex APF11 Iridium float versions, equipped with an oxygen sensor provide a series of DO surface measurements in addition to the single sample measurements taken just before GPS determination as described above for Apex APF9 floats.

Implementation of the surface series of oxygen measurements was based on the SCOR Working Group 142 recommendation. As described by Hugh FARGHER (from TWRC), float operation during this measurement phase is typically as follows:

"… on surfacing, 10 oxygen measurements are taken at 15-second intervals before & after inflating the air bladder. The resulting data (recorded in the science_log file) should provide enough quality control data to ensure high long-term accuracy for oxygen readings."

Thus, for these floats, this additional information is stored in the TRAJ file:

- Concerning measurement series sampled before bladder inflation:
  - PPOX_DOXY parameter is computed and added to these measurements;
  - Some floats make measurements that are sampled after Ascent End Time (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with MC=700+10=710.
  - Some floats make measurements that are sampled after Transmission Start Time (associated MC=700) and before Transmission End Time (associated MC=800), the reference MC used is MC=800 and they are stored in the TRAJ file with MC=800+10=810.
- Concerning measurement series sampled after bladder inflation:
  - PPOX_DOXY parameter is computed and added to these measurements;
  - Some floats make measurements that are sampled after Ascent End Time (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with MC=700+11=711.
  - Some floats make measurements that are sampled after Transmission Start Time (associated MC=700) and before Transmission End Time (associated MC=800), the reference MC used is MC=800 and they are stored in the TRAJ file with MC=800+11=811.
-
- Concerning single measurements sampled after bladder inflation:
  - PPOX_DOXY parameter is computed and added to these measurements;
  - As these measurements are sampled after the in-air series and before the end of transmission, the reference MC is 800 and they are stored in the TRAJ file with MC=800-1=799.

2.4.4.3 Navis floats

Some older Navis floats provide 3 distinct sets of "Near surface" or "In air" measurements:

- One set of "Near surface" samples;
- Two sets of "Surface" samples: one sampled with the bladder deflated, the second sampled with the bladder inflated.

For these data, additional information is stored in the TRAJ file:

- Concerning "Near Surface" samples (provided by the Aanderaa 4330 optode):
  - DOXY and PPOX_DOXY parameters are computed and added to these measurements and stored in the TRAJ file;
  - DOXY parameter is computed and added to these measurements and stored in the PROF file, in the same profile as the remaining DOXY data (i.e. not in a dedicated "Near-surface sampling: []" profile).
  - As these measurements are sampled after Ascent End Time (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700-10=690.
- Concerning "Surface Bladder deflated" samples:
  - PPOX_DOXY parameter is computed and added to these measurements;
  - As these measurements are sampled after Ascent End Time (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700+10=710.

- Concerning “Surface Bladder inflated” samples:
    - PPOX_DOXY parameter is computed and added to these measurements;
    - As these measurements are sampled after Ascent End Time (associated MC=600) and before Transmission Start Time (associated MC=700), the reference MC used is MC=700 and they are stored in the TRAJ file with the MC=700+11=711.

Note that many Navis floats available are outfitted with pumped SBE63 oxygen sensors that are not capable of sampling in air.

2.5 GROUNDED Flags

The updated GROUNDED flags can be found in Reference table 20 in the Users Manual and on the NVS (nvs-vocabs/R20: Argo GROUNDED flags (github.com)). The table has been updated in an effort to make the grounded flags more useful when making velocity estimates during the drift period. There are two different ways to determine grounding: 1) based on float performance and technical data or 2) based on checks with bathymetry. Each of these methods now allows for DACs and/or DM operators to indicate if the float grounded during drift which would affect velocity estimates or if it grounded elsewhere in the mission which likely would not affect a velocity estimate. Please choose your flag conservatively; if unsure if grounded affected the drift, use 'Y' or 'B'.

Grounded flag	Meaning
Y	Yes, the float touched the ground and the programmed free-drift period was affected. If it is uncertain what mission phase the grounding took place, use 'Y'.
P	Yes, the float touched the ground during a part of the mission that did not affect the free-drift period.
B	Yes, the float touched the ground as determined by an external bathymetry database and the programmed free-drift period was affected. If it is uncertain what mission phase the grounding took place, use 'B'.
C	Yes, the float touched the ground as determined by an external bathymetry database during a part of the mission that did not affect the free-drift period.
N	No, the float did not touch the ground
S	Float is known to be drifting at a shallower depth than originally programmed. *Warning: this is not a "grounded" situation. The "S" flag should probably be declared obsolete.*
U	Unknown *Warning: U and FillValue " " have a close meaning. The "U" flag should probably be declared obsolete*

ANNEX A: Some definitions

Here are some definitions about elements mentioned in this document, if some of them remain unclear, please ask for a new or updated definition (argo@ucsd.edu, support@argo.net).

2.6 Definitions of Argos raw data contents

The following definitions can be found in the Argos User's manual (http://www.argos-system.org/manual/).

Let us consider Argos raw data provided in a PRV/DS command output format.

Argos float messages are collected by a given satellite during a satellite pass. A header of the satellite pass (in underlined bold in the two following examples) is added to the data by CLS.

In this header, one can find:

- The number of lines of data relative to the satellite pass header (including the header line),
- The name of the satellite,
- If a location has been computed from the data collected during the satellite pass (example 2):
  - The location class of the location,
  - The date of the location,
  - The latitude and longitude of the location.

The Argos float messages, collected during the satellite pass, follow the header.

For each we find:

- The Argos message date (time of reception of the message by the satellite),
- The Argos message redundancy,
- The Argos message content.

Example 1: A satellite pass without Argos location.

```
02412 63706 17 31 L
2007-04-24 02:40:16 2 64 A2 56 BA
B2 3C 8D 7C
AF 9F 85 AD
72 ED D5 4E
65 09 F7 5D
5C 1E B9 52
D0 CE AA 61
9A 30 00
2007-04-24 02:40:58 1 67 09 D5 CB
5F 31 75 7C
23 8D 3D 82
73 AA 30 8E
5C 46 A1 C7
68 D0 F9 91
D9 60 B9 7E
EB 38 00
```

Example 2: A satellite pass with Argos location.

```
02412 63706 33 31 D 2 2007-04-24 05:30:15 -32.189 11.405 0.000 401651871
2007-04-24 05:27:35 1 51 C9 1B A6
F4 0B 5B 5F
2E 83 F4 7F
DF E0 E4 06
1F 99 80 1E
94 6A 80 FD
```

```
FE 10 39 1E
A7 F4 00
2007-04-24 05:30:15 1 05 08 16 92
0F 83 AE 18
40 20 90 0A
20 00 19 9E
04 15 A6 00
0C 39 05 85
89 01 8E 04
C9 68 20
2007-04-24 05:30:52 1 58 A3 7D 66
F4 8B 16 DF
33 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00
2007-04-24 05:32:55 1 69 F5 58 B9
28 72 27 88
72 A5 59 91
54 B5 B4 2A
95 02 43 52
A8 49 2A 5C
E9 DD 4C F7
62 00 00
```

2.7 Cyclic Redundancy Check

All Argos floats (except SIO SOLO) have an error detection code embedded in their Argos messages. Checking this code, called Cyclic Redundancy Check (CRC), can theoretically enhance the reliability of the data by rejecting messages possibly corrupted by transmission error.

2.8 Float clock drift and clock offset

Some Argo float versions provide times for dated events or dated measurements. Over time, the float's clock may drift. Clock drift can be defined as the drift of the clock in hours/ minutes/ seconds per year. To correct for this, we must apply a clock offset where clock offset is defined as a measurement, done at a given time, of the offset of the clock due to clock drift. Thus a clock offset should be estimated for each of these float times.

Note that, in this document, float clock offset can also embrace a clock that has not been correctly set or a clock that has been set in local time. Of course, in these cases clock offset is not only revealing a drift of the float clock...

Float clock offset is defined as: Float clock offset = Float time - UTC time.

A good estimate of the clock offset can be obtained when the float transmits its Real Time Clock (RTC) time in the technical data. It can then be compared to the time from Argos of the corresponding message to compute a clock offset for all the float times of the concerned cycle.

Unfortunately this is not always the case, some floats do not transmit their RTC time and even if they do, this RTC time is not always received.

2.9 APEX Argos test/data messages

APEX Argos test messages are transmitted by an APEX float during the six hours period spent at the surface prior to its first dive. The test message contains programmed mission parameters and technical data.

After each deep cycle, APEX floats transmit the collected data in the ARGOS data messages.

2.10 APEX Deep Profile First floats

Some APEX floats are programmed to achieve their first profile shortly after deployment (for comparison to conventional CTD cast from the ship).

In the Deep Profile First (DPF) cycle the firmware is set to complete the first profile within 24 hours of deployment. The float descends to park, parks then descends to profiling depth. Then the profile commences. The duration of park is 5 minutes for Iridium floats and 60 minutes for ARGOS floats. The DPF also ignores any time of day setting. It is unclear how long the two descents are. Experience has shown that both the descent time out configuration settings are activated when the float fails to reach target depths.

2.11 APEX Time Of Day feature

Some APEX floats have the capability to schedule profiles so that the float surfaces at a particular Time Of Day (TOD).

When the TOD feature is enabled, the float RTC is used to dynamically set the end of the DOWN TIME period to a (user programmed) number of minutes after midnight.

The time of day feature is ignored by Deep Profile First floats.

2.12 APEX Auxiliary Engineering Data

For some APEX floats, the remaining space of the last Argos data message is filled with Auxiliary Engineering Data (AED).

The AED are considered to be of lower priority and will never cause an additional Argos data message to be generated.

3 ANNEX B: Transmission End Time estimation for an APEX Argos float

The methods used to estimate TET for an Apex Argos float are all based on the float's theoretical functioning.

3.1 Apex float theoretical functioning

The core cycle of an APEX float is based on four main parameters as illustrated in Figure 2:

1. DOWN TIME: The time period including the descent and the drift at depth,
2. UP TIME: The time period including the ascent and the drift at surface,
3. PARKING PRESSURE: The aimed depth for the drift phase,
4. DEEPEST PRESSURE: The aimed depth for the start of the profile (also called PROFILE PRESSURE in the document).

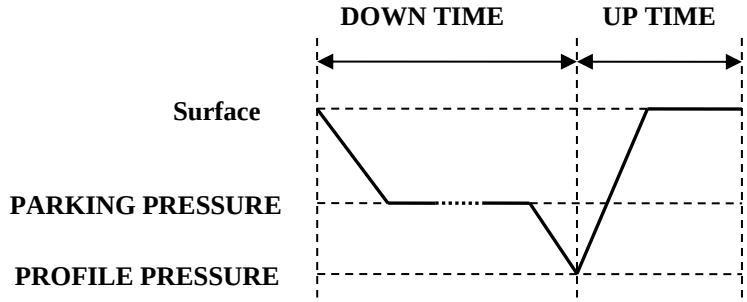


Figure 2: Main parameters of an APEX cycle

The float can also be programmed to achieve a deep profile once over *N* cycles (in this case, *N* is called the Park and Profile parameter).

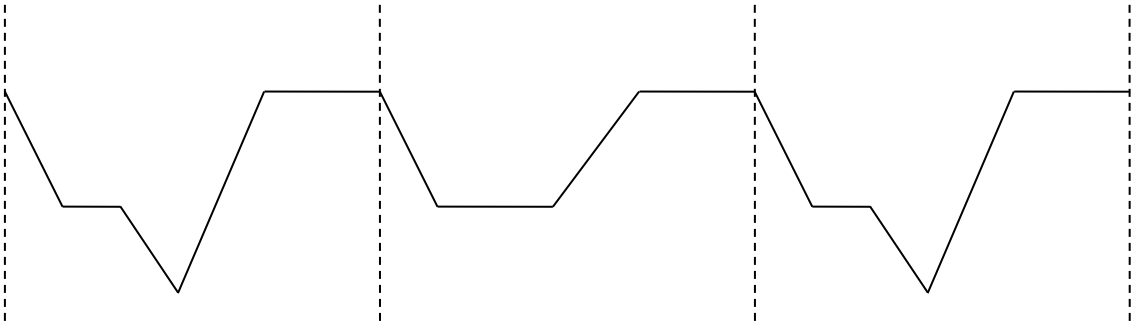


Figure 3: Example of programmed mission with a PnP parameter of 2 (deep profile every second cycle)

Finally, the float can also be programmed to achieve its first profile shortly after deployment (for comparison to conventional CTD cast from the ship).

During its first cycle, this Deep Profile First (DPF) float directly descends to the PROFILE PRESSURE and immediately achieves its first profile.

3.2 The Park et al. method

The Park et al. method is based on the assumption that the Apex Argos float always starts to profile at the end of the DOWN TIME. Unfortunately, this is only the case for floats when PARKING and PROFILE pressures are equal.

The float starts its descent to PROFILE PRESSURE hours before the end of the DOWN TIME. The length of time to descend from the drift pressure to the profile pressure varies. It is user specified for APF9 floats. Often users choose somewhere between 4 and 6 hours. Six hours is shown in the diagram below.

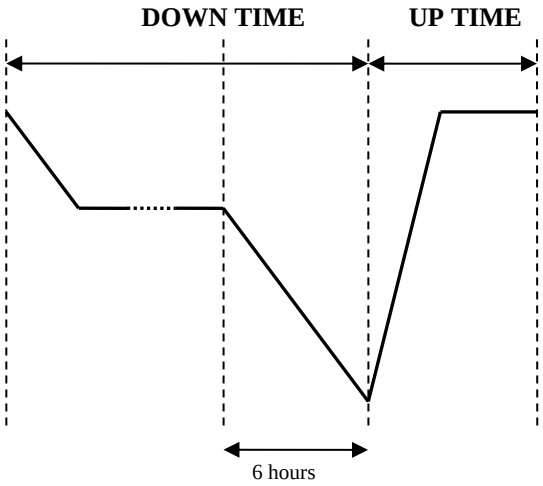


Figure 4: Schematics of a cycle where PARKING and PROFILE pressures differ

The float then starts to profile when one of the following conditions is met:

- If the PROFILE PRESSURE depth is reached,
- If the end of the DOWN TIME period is reached.

Note that the second assumption is only true for floats without Time Of Day (TOD) capability or with the TOD feature disabled.

In the first case, the PROFILE depth can be reached before the end of the DOWN TIME period implying an arrival at the surface earlier than for the theoretical case (under the assumption of a constant ascent rate), see next figure.

After such a cycle, if it reached the surface before the end of the DOWN TIME period, the float increases the value of its profile piston position count to decrease its descent rate from PARKING depth to PROFILE depth for the next cycle.

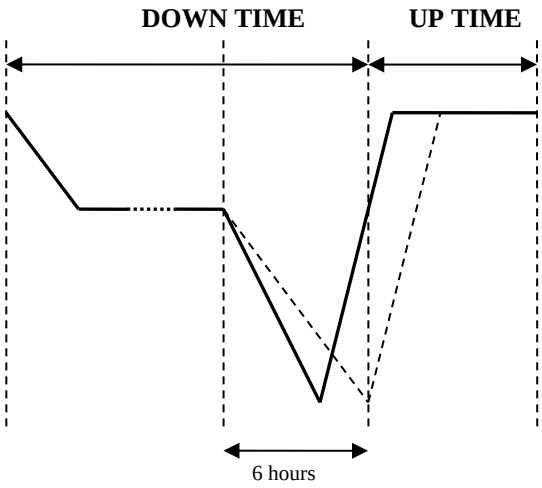


Figure 5: First case, PROFILE depth reached before end of the DOWN TIME period

In the second case, at the end of the DOWN TIME, the float has not necessarily reached the PROFILE depth implying an arrival at the surface earlier than for the theoretical case but later than for the first case (under the assumption of a constant ascent rate), see next Figure.

After such a cycle, the float decreases the value of its profile piston position count to increase its descent rate from PARKING depth to PROFILE depth for the next cycle.

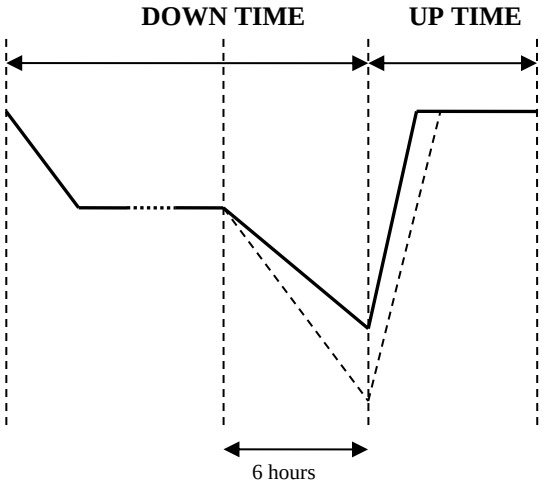


Figure 6: Second case, PROFILE depth not reached before end of DOWN TIME period

This behavior produces continual variation of AETs; 2 main values can roughly be distinguished as illustrated in the next figure.

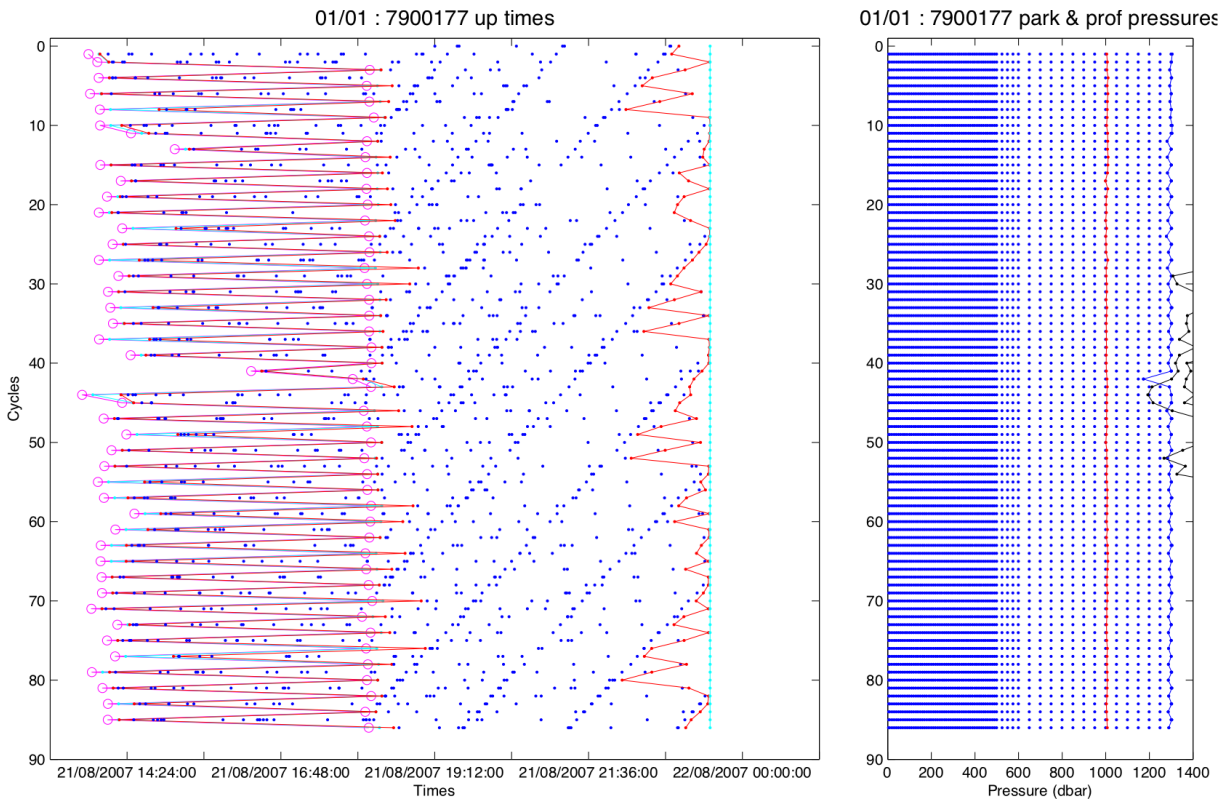


Figure 7: Example of behaviour of a float which drift and profile at different depths

In this figure, we can see on the left part **the times** and on the right part **the pressures of the CTD measurements**.

The times are drawn in reference to cycle #0. For a given cycle, we can find, in chronological order: the AET (pink circle), the TST (light blue dot), the FMT (red dot), the location times (blue dots), the LMT (red dot) and the TET (light blue dot).

We clearly see 2 sets of AETs: the first one around 21/08/2007 14:16:00 and the second one around 21/08/2007 18:14:00 (in reference to cycle #0).

The pressures of the CTD measurements are drawn in blue dots for the profile measured pressures and in red dots for the Representative Parking Pressure (RPP) pressures. The black dots represent the local bathymetry provided by the ETOPO2 atlas.

We see that this float (WMO #7900177) has drift around 1000 dbar and profiled from 1300 dbar.

More detailed information can be found in the Kobayashi and Nakajima paper.

In the DEP data set, only 22.5% (i.e. 806 floats) of the 3622 APEX Argos floats drift and profile at the same depth and are then eligible for the Park et al. method.

Consequently, we decided to forget this method and to specify another one applicable to all APEX Argos floats.

3.3 The proposed method

The proposed method is based on two algorithms that can be alternatively used, depending of the number of cycles available for a given float.

The first algorithm uses half of the Park et al. method to determine the maximum envelope of the LMTs.

The second algorithm also estimates TETs but takes into account the drift of the float clock.

These algorithms are based on the duration of the APEX Argos cycles which are always known CYCLE TIME = DOWN TIME + UP TIME.

We cannot say that these durations are constant but we can say that their theoretical values are known (i.e. predictable).

In the ANDRO data set, these cycle durations vary only for two float versions:

- For APEX bounce floats: bounce cycle (even numbered cycles) duration is smaller than usual cycle (odd numbered cycles) duration (but both theoretical values are known),
- For APEX "seasonal" floats: the cycle duration is two (known) values depending of the day in the year (concerned versions are 001046 and 001055).

3.3.1 First algorithm: Transmission End Times estimated from the maximum envelope of the Last Message Times

The maximum envelope of the LMTs is a lower bound of the TET as illustrated in the following figure.

The first algorithm is the following:

1. Identify the reference cycle (number *N*).
   The reference cycle is the first received cycle.
   For DPF floats however the reference cycle can't be cycle #0 it is then the first received cycle with a number > 0.
2. Compute the reference value of the LMT of each cycle (LMT_{RV}).
   For cycle #*i*:
   $LMT_{RV}(i) = LMT(i) - duration(N, i)$
   Where duration(*N*, *i*) is the theoretical duration between cycle #*N* and cycle #*i*.
   If the cycle theoretical duration is constant and equal to CYCLE TIME (all floats except bounce and "seasonal" ones) duration(*N*, *i*) = (*i* - *N*)*CYCLE TIME and then
   $LMT_{RV}(i) = LMT(i) - (i - N)$*CYCLE TIME
3. Find the maximum value of the obtained $LMT_{RV}(i)$ values.
4. Compute the TET of each cycle.
   $TET(i) = \max(LMT_{RV}) + duration(N, i)$
   Here again if the cycle theoretical duration is constant we obtain
   $TET(i) = \max(LMT_{RV}) + (i - N)$*CYCLE TIME

Note that for DPF floats we can choose to set TET equal to LMT for cycle #0 (i.e. for DPF floats: TET(0) = LMT(0)).

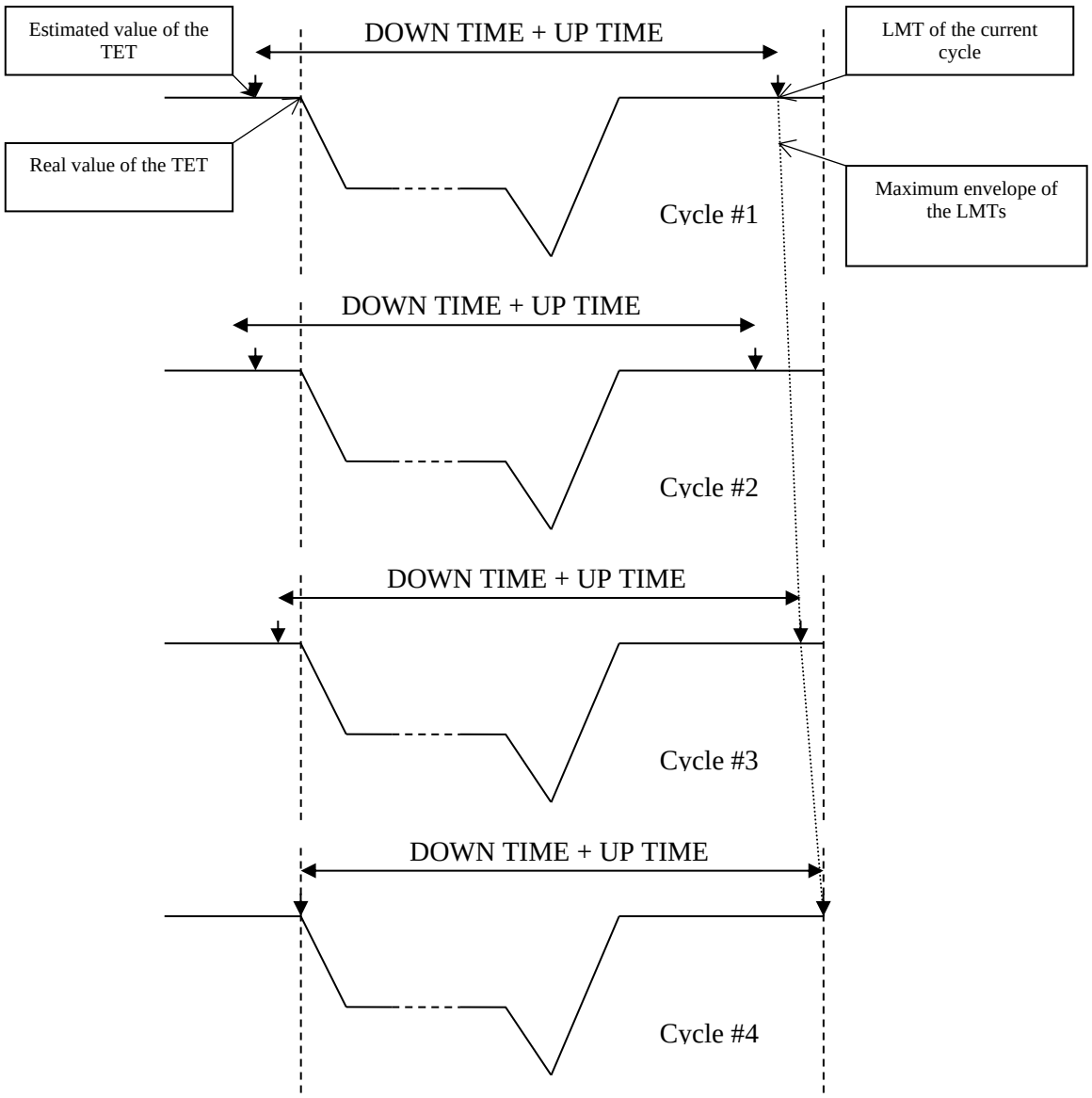


Figure 8: Estimation of the TETs from the maximum envelope of the LMTs

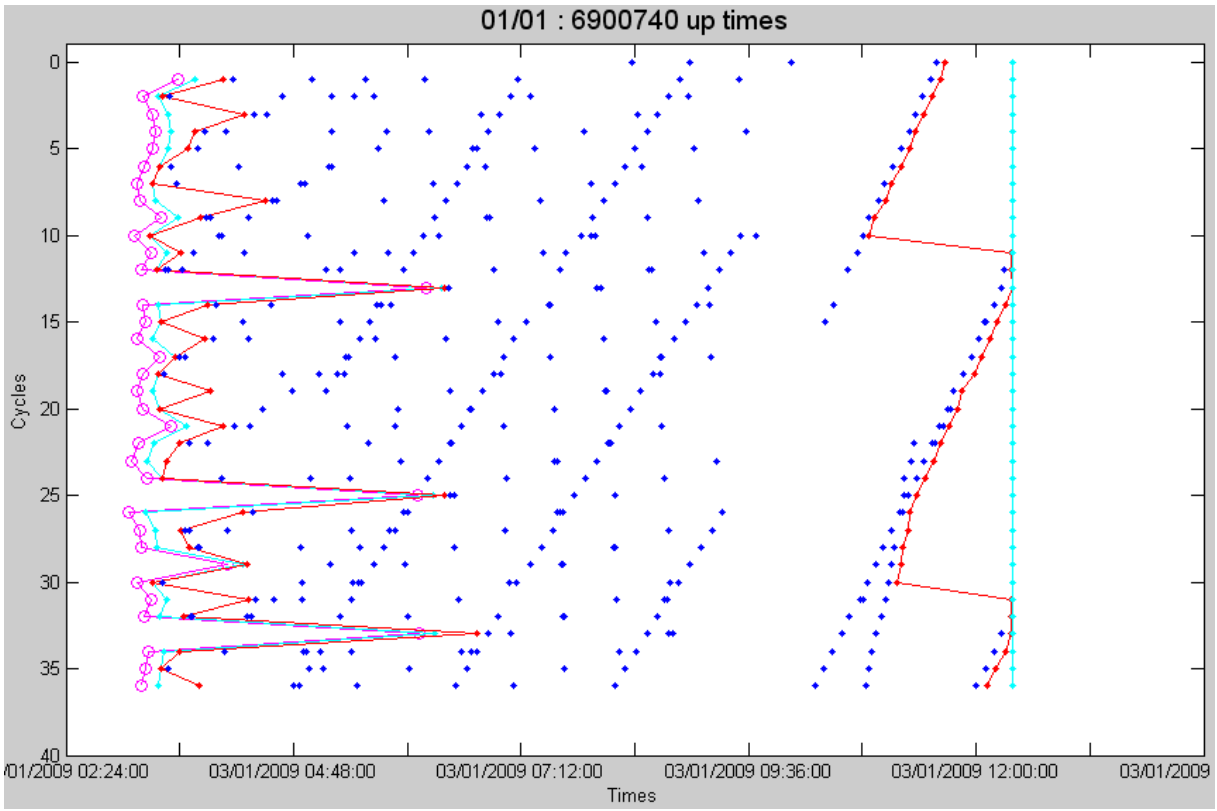


Figure 9: Example of estimation of TETs with the maximum envelope of the LMTs

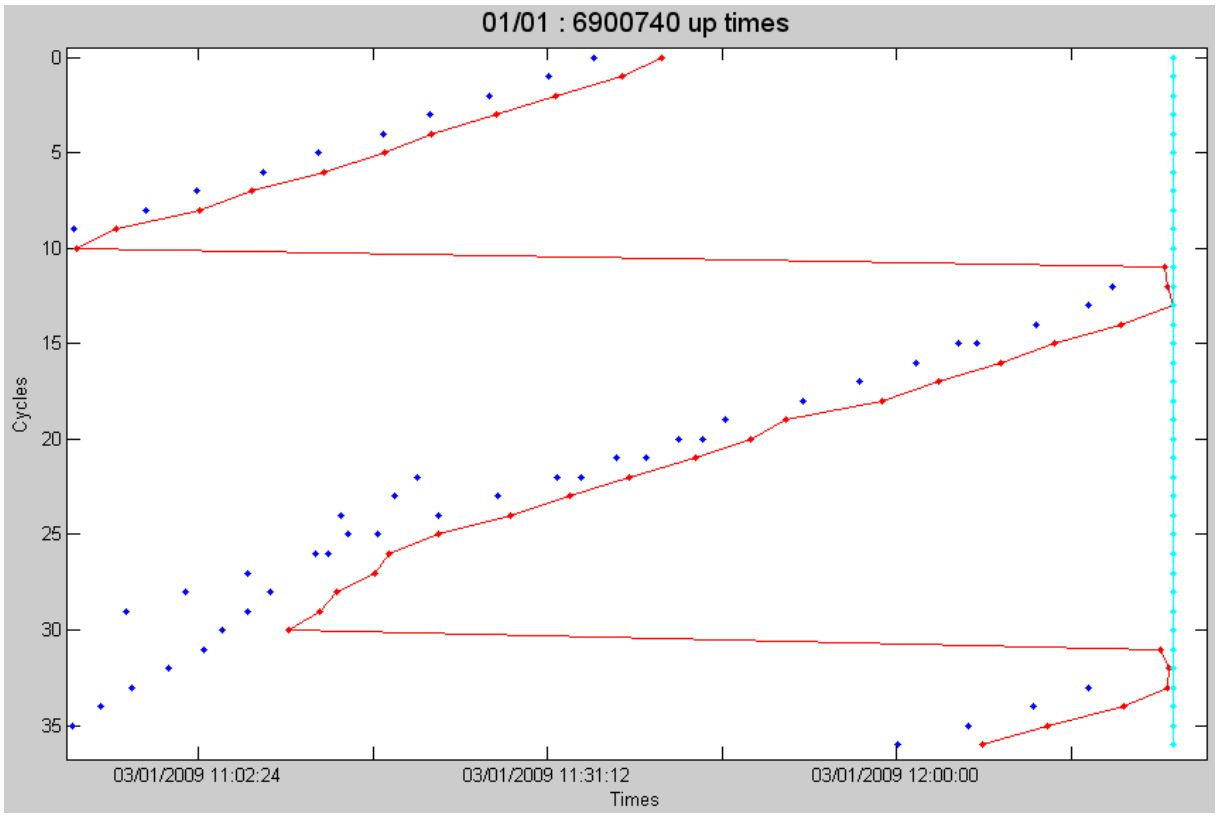


Figure 10: A zoom of the previous figure shows that estimated TETs are defined by the LMT of cycle #13

3.3.2 Second algorithm: Transmission End Times estimated by a method that takes the float clock offset into account

The TETs obtained so far have been estimated under the assumption that the theoretical CYCLE TIME value is the real duration of the cycles.

However, the on board float clock can drift during float life implying variations of the cycle durations.

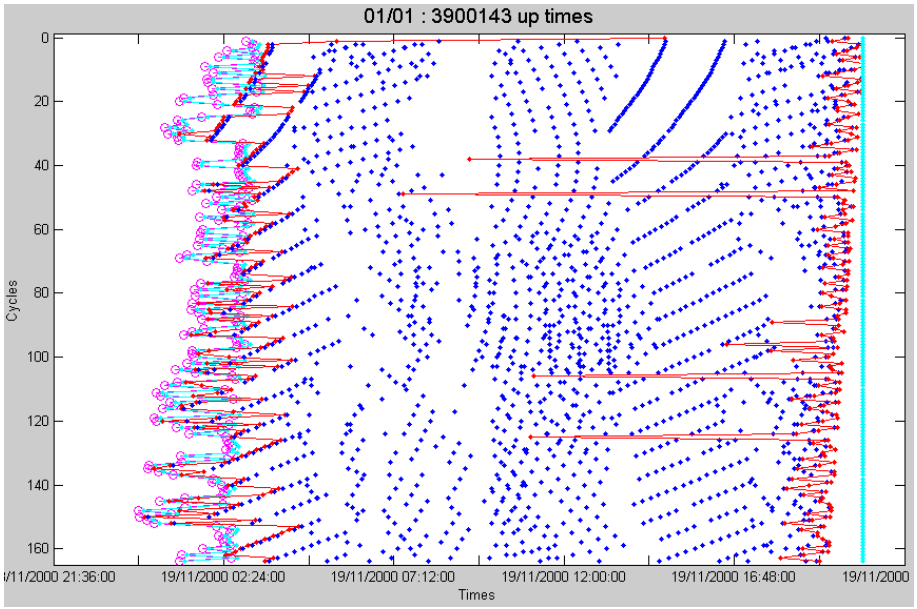


Figure 11: Example of a rather important negative clock drift (estimated to - 00:13:59 per year which implies a maximum difference of 00:58:34 between TETs estimated with or without taking the clock drift into account). We clearly see a regular decrease of cycle duration.

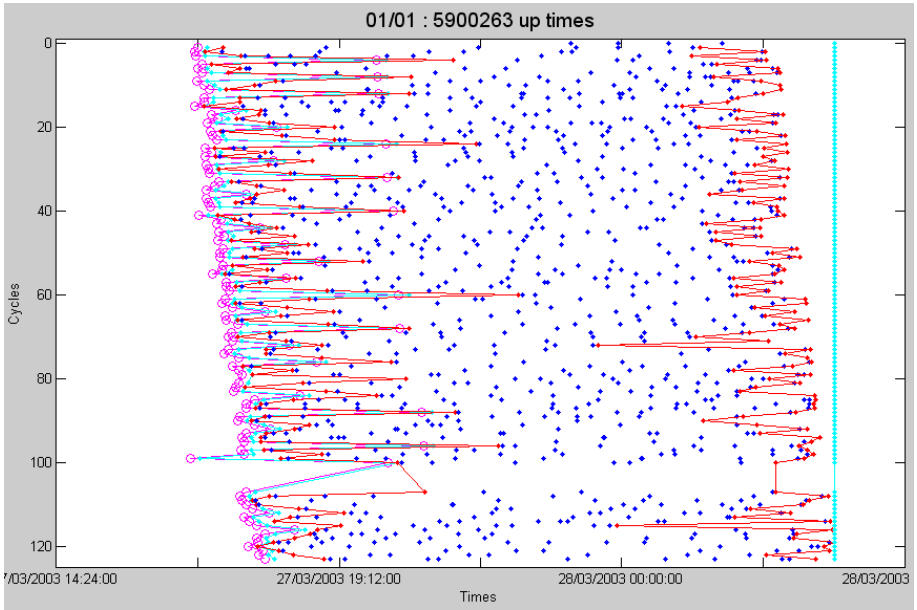


Figure 12: Example of a rather important positive clock drift (estimated to + 00:17:15 per year which implies a maximum difference of 00:58:56 between TETs estimated with or without taking the clock drift into account). We clearly see a regular increase of cycle duration.

The proposed algorithm can also take into account erroneous theoretical CYCLE TIME values.

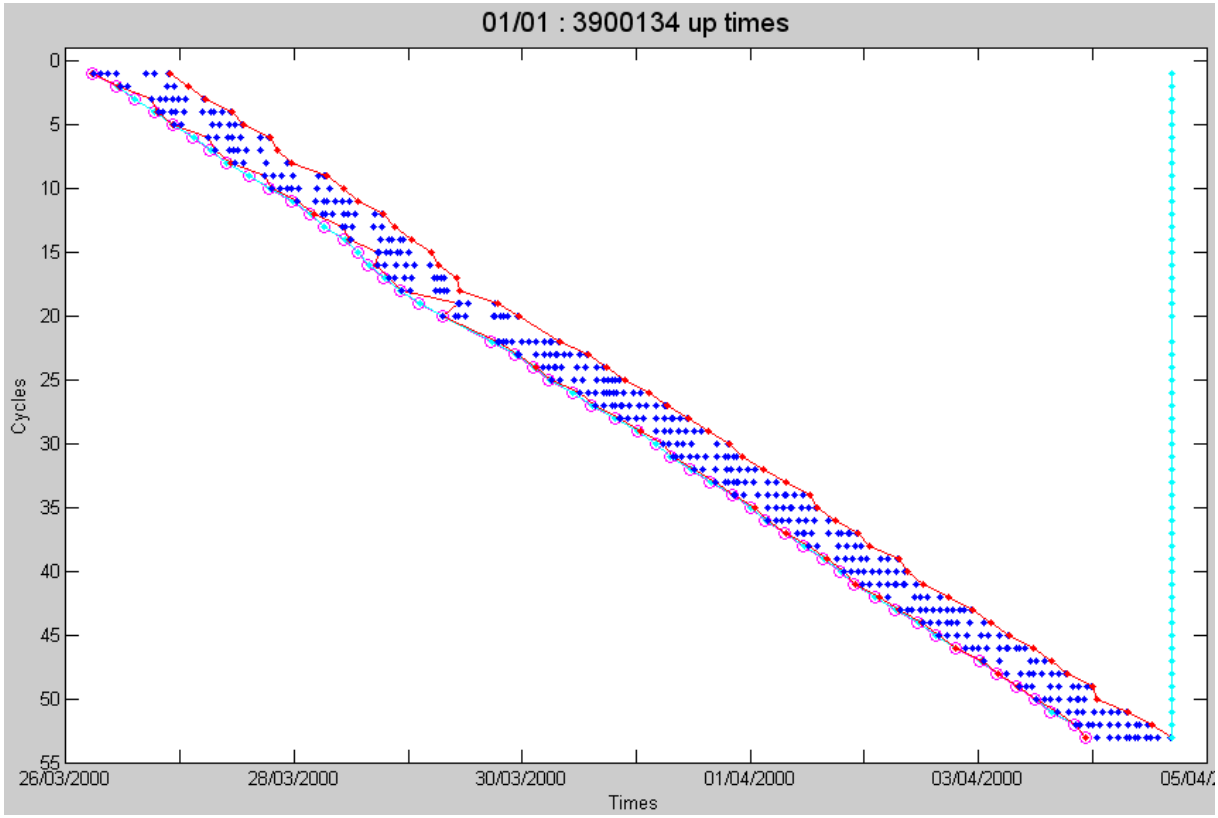


Figure 13: Example of an erroneous theoretical cycle time (DOWN TIME + UP TIME = 228 + 22 = 240 hours whereas the cycle time is around 244 hours) seen as a very important positive clock drift (estimated to + 147:44:36 per year which implies a maximum difference of 209:26:54 between TETs estimated with or without taking the clock drift into account).

In the second algorithm we **linearly estimate the float clock offset** to take it into account in the TETs estimation.

The algorithm stands in six steps illustrated in the following paragraphs.

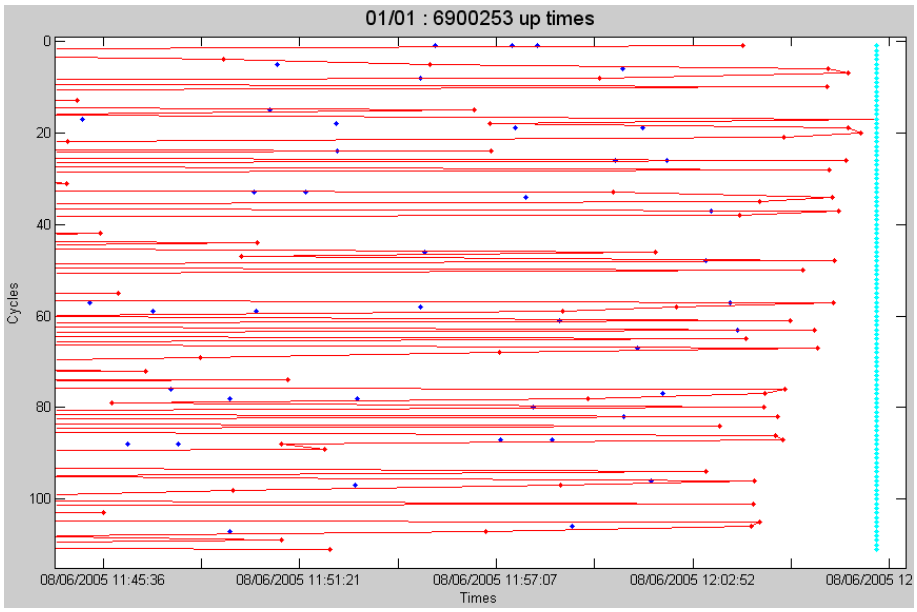


Figure 14: Example of TETs estimated with the maximum envelope of the LMTs (first algorithm), we want to estimate the TETs taking into account the float clock offset.

3.3.2.1 Step #1

First determine the convex envelope of the LMTs.

This can be done by various algorithms; the one we used is in section 5.3.2.7.

Next, obtain a subset of the LMTs (the base points) that define the convex envelope of all the LMTs as illustrated in the next figure.

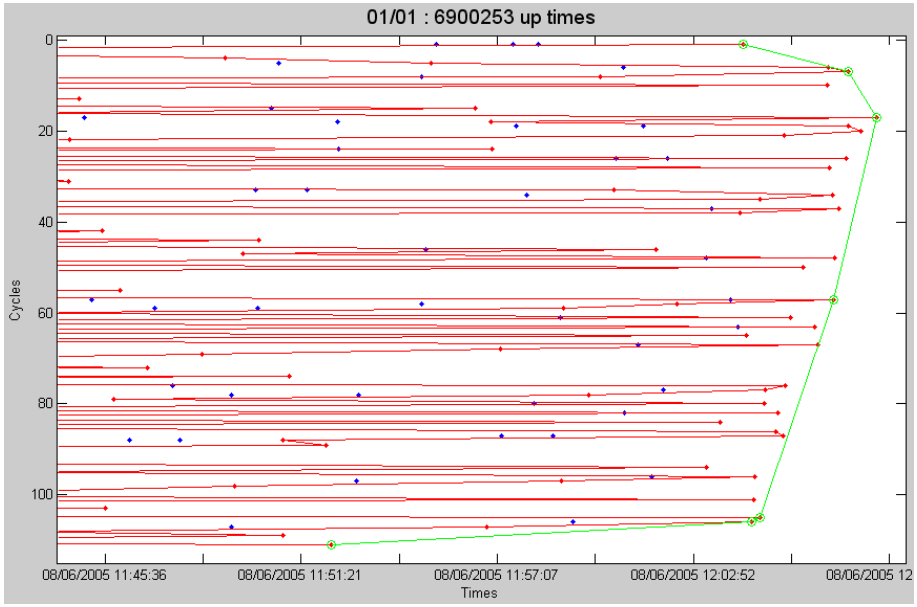


Figure 15: The convex envelope of the LMTs (green line) defined by the base points (green circles)

3.3.2.2 Step #2

The second step consists in setting a point on the convex envelope for all received cycles.

We thus obtain a set of points on the convex envelope.

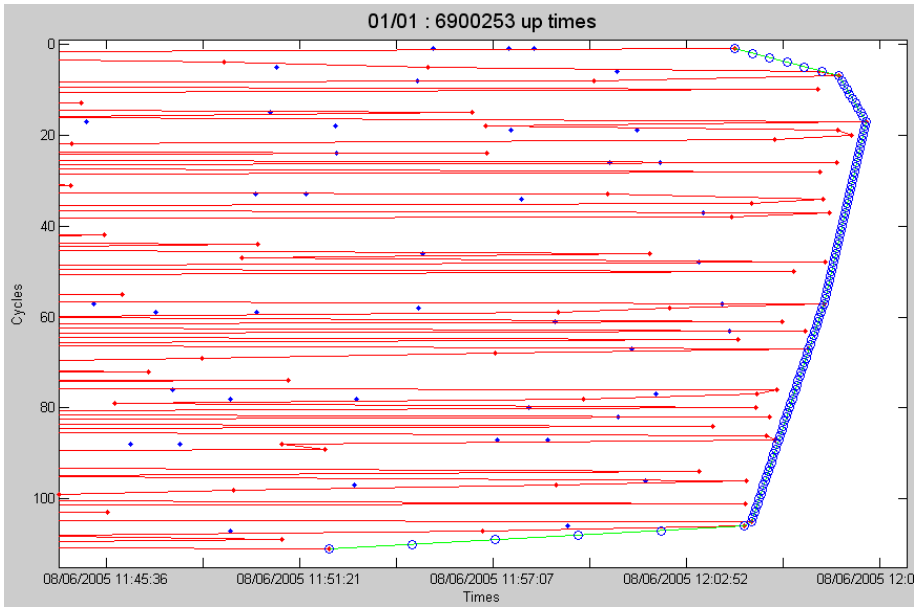


Figure 16: The 111 received cycles are set on the convex envelope (blue circles)

3.3.2.3 Step #3

The third step consists in deleting some first and last points on the convex envelope.

2/5 of the points on the convex envelope are deleted: the first 1/5 of the point and the last 1/5 of the points.

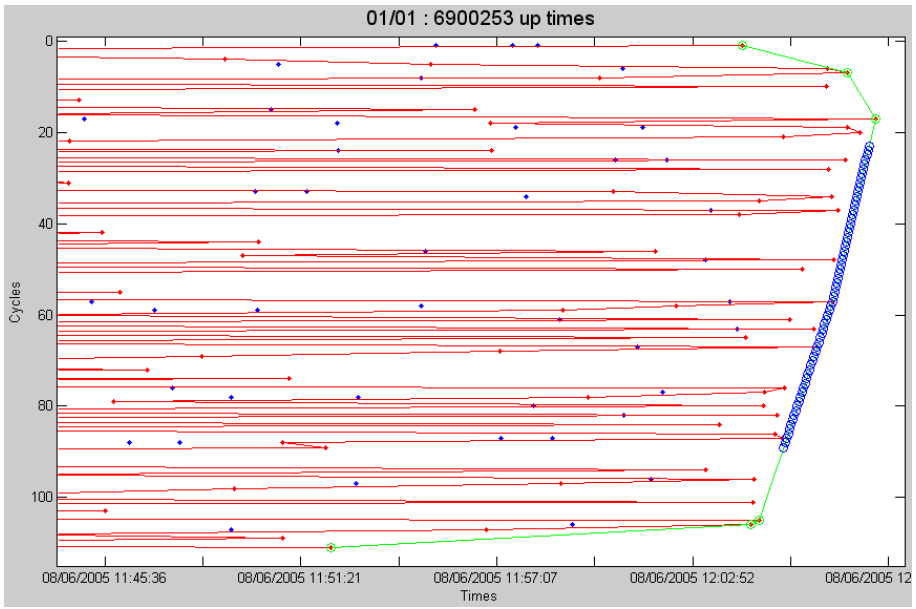


Figure 17: The first and last 22 points are ignored (only the 67 central points are preserved)

3.3.2.4 Step #4

The fourth step consists in linearly fitting the points on the convex envelope (in a least squares sense).

We used the "polyfit" Matlab function to do that.

The slope of the resulting line is our estimated clock offset.

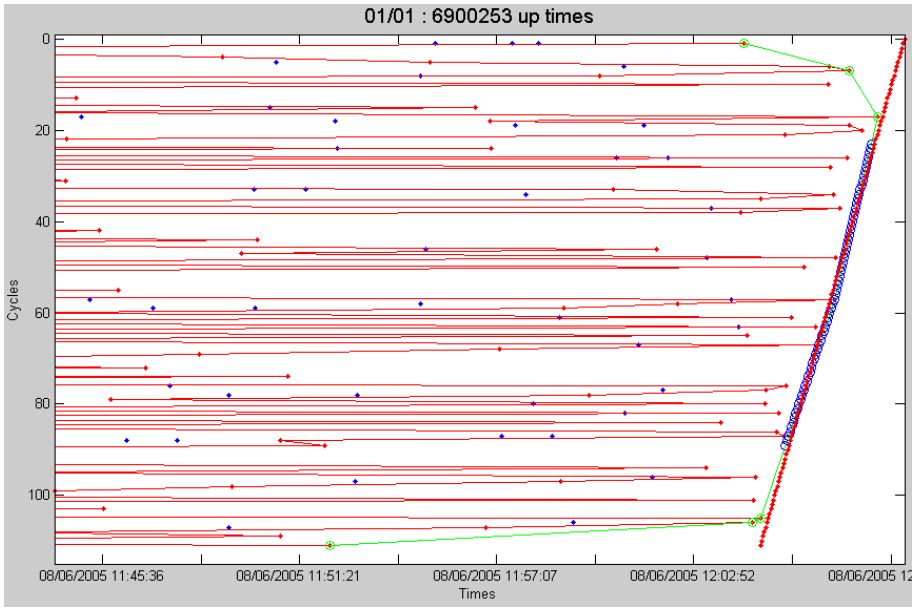


Figure 18: The blue circles are linearly fitted (red line)

3.3.2.5 Step #5

The fifth step consists of adjusting the estimated clock offset on the convex envelope.

Obviously, the contact point(s) is(are) base point(s), thus we try each base point of the convex envelope.

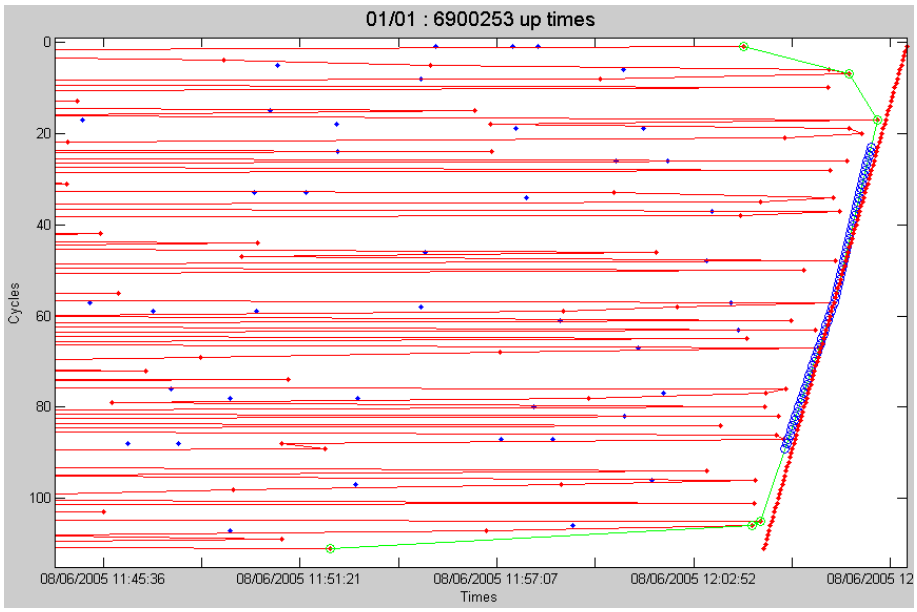


Figure 19: The red line is adjusted on the convex envelope

3.3.2.6 Step #6

For each received cycle, we compute the estimated TETs situated on the adjusted line of estimated clock offset.

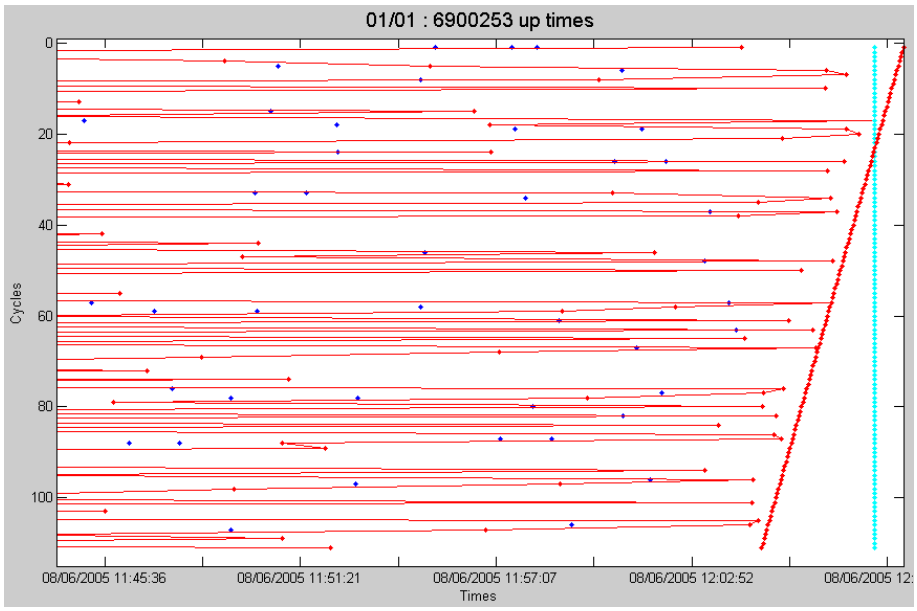


Figure 20: Estimated TETs with (red points on the red line) or without (light blue points on the light blue line) taking float clock drift into account. In this example, clock drift is estimated to -00:00:55 per year which implies a maximum difference of 00:03:19 between the red and the light blue lines

3.3.2.7 Example of implementation of the second algorithm

```
% Extract of the "dep_add_apx_descent_start_with_clock_drift.m" program used in
% the ANDRO toolbox.
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)

% Inputs:
% cycles: cycle numbers
% cycleTime : cycle durations
% tabLastPosDate: date of the last Argos location for each cycle
% tabLastMsgDate: date of the last received Argos message for each cycle

% Due to possible anomalies in cycle duration (DPF floats, clock jumps, etc),
% the cycles have to be processed by slices.
% A slice is defined by a number of consecutive cycles.
% For each slice we can define the following parameters:
%    - tabFirstId/tabLastId: first/last Id of the cycles that define the slice
%    - tabNbMinPts: minimum number of cycles needed to estimate the clock drift
%      for a given slice (set to 33 by default)

% Outputs:
% tabArgosStopFinal: the max envelope of the LMTs
% tabSavEstClockDrift: the estimated clock drift for each slice


% max envelope of the LMTs
tabArgosStopFinal = ones(max(cycles)+1, 1)*g_dateDef;

% estimated clock drifts
tabSavEstClockDrift = ones(length(tabFirstId), 1)*g_dateDef;

% processing of each slice
tabIdSlice = [];
tabCoef = [];
tabNbPts = [];
tabCycleTime = [];
for idSlice = 1:length(tabFirstId)

  firstId = tabFirstId(idSlice);
  lastId = tabLastId(idSlice);
  fprintf('   Slice #%d: cycles %d to %d\n', ...
    idSlice, firstId-1, lastId-1);
  cycleSlice = cycles(firstId:lastId);
  nbMinPts = tabNbMinPts(idSlice);

  % last Argos message date (or last location date) of each cycle
  convexEnv = tabLastMsgDate(firstId:lastId);
  argosLastLoc = tabLastPosDate(firstId:lastId);
  idNotDated = find(convexEnv == g_dateDef);
  convexEnv(idNotDated) = argosLastLoc(idNotDated);
  oriDates = convexEnv;

  % maximum envelope without clock drift estimation
  maxEnv = convexEnv;
  idDated = find(maxEnv ~= g_dateDef);
```

```
  maxEnvDates = maxEnv(idDated);
  maxEnvCycles = cycleSlice(idDated);
  if (~isempty(maxEnvCycles))
    maxEnvDates = maxEnvDates - compute_duration(maxEnvCycles, maxEnvCycles(1),
cycleTime);
    maxDate = max(maxEnvDates);
    maxEnv(idDated) = maxDate + compute_duration(maxEnvCycles, maxEnvCycles(1), cycleTime);

    tabArgosStopFinal(firstId:lastId) = maxEnv;
  end

  % compute the convex envelope of the LMTs
  oneMore = 1;
  while (oneMore)
    oneMore = 0;
    idDated = find(convexEnv ~= g_dateDef);
    if (~isempty(idDated))
      idStart = idDated(1);
      stop = 0;
      while (~stop)
        stop = 1;
        idIdDated = find(idDated > idStart);
        for id = 1:length(idIdDated)-1
          idFirst = idStart;
          xFirst = compute_duration(cycleSlice(idStart), cycleSlice(1), cycleTime);
          yFirst = convexEnv(idFirst);
          idCheck = idDated(idIdDated(id));
          xCheck = compute_duration(cycleSlice(idCheck), cycleSlice(1), cycleTime);
          yCheck = convexEnv(idCheck);
          idLast = idDated(idIdDated(id+1));
          xLast = compute_duration(cycleSlice(idLast), cycleSlice(1), cycleTime);
          yLast = convexEnv(idLast);

          coefA = (yFirst - yLast)/(xFirst - xLast);
          coefB = yFirst - coefA*xFirst;

          expValue = coefA*xCheck + coefB;
          if (yCheck <= expValue)
            % this point is not part of the convex envelope
            convexEnv(idCheck) = g_dateDef;
            % one point has been excluded, a new iteration is needed
            oneMore = 1;
          else
            % we must start from this base point (possibly in the convex
            % envelope)
            idStart = idCheck;
            stop = 0;
            break;
          end
        end
      end
    end
  end

  % the dates of the excluded points are set on the convex envelope
```

```
  convexEnvAll = convexEnv;
  idDated = find(convexEnv ~= g_dateDef);
  for id = 1:length(idDated)-1
    idFirst = idDated(id);
    xFirst = compute_duration(cycleSlice(idFirst), cycleSlice(1), cycleTime);
    yFirst = convexEnv(idFirst);
    idLast = idDated(id+1);
    xLast = compute_duration(cycleSlice(idLast), cycleSlice(1), cycleTime);
    yLast = convexEnv(idLast);

    polyCoef = polyfit([xFirst xLast], [yFirst yLast], 1);

    idNew = [idFirst+1:idLast-1];
    idNew(find(oriDates(idNew) == g_dateDef)) = [];

    convexEnvAll(idNew) = polyval(polyCoef, compute_duration(cycleSlice(idNew), cycleSlice(1),
cycleTime));
  end

  % clock drift estimation
  idDated = find(convexEnvAll ~= g_dateDef);
  if (length(idDated) < nbMinPts)
    fprintf('      Not enough points (%d) => no clock drift estimation\n', ...
      length(idDated));
  else

    % delete the first and last 1/5 of the points
    nbTotal = length(idDated);
    nbToDel = fix(length(idDated)/5);
    fprintf('      Number of points total:del/used/del %d:%d/%d/%d\n', ...
      length(idDated), nbToDel, length(idDated)-2*nbToDel, nbToDel);
    convexEnvAll(idDated(1:nbToDel)) = g_dateDef;
    convexEnvAll(idDated(end-(nbToDel-1):end)) = g_dateDef;

    % linear estimation of the clock drift with the remaining points
    idDated = find(convexEnvAll ~= g_dateDef);
    xVal = compute_duration(cycleSlice(idDated), cycleSlice(1), cycleTime);
    yVal = convexEnvAll(idDated);

    polyCoef = polyfit(xVal, yVal, 1);

    if (length(unique(cycleTime(cycleSlice+1))) == 1)
      tabIdSlice = [tabIdSlice; idSlice];
      tabCoef = [tabCoef; polyCoef(1)];
      tabNbPts = [tabNbPts; length(idDated)];
      tabCycleTime = [tabCycleTime; unique(cycleTime(cycleSlice+1))];
    end

    % set the estimate on the convex envelope (it is a base point of the
    % envelope)
    basePoint = [];
    idDated = find(convexEnv ~= g_dateDef);
    for id = 1:length(idDated)
      idPoint = idDated(id);
      xPoint = compute_duration(cycleSlice(idPoint), cycleSlice(1), cycleTime);
```

```
      yPoint = convexEnv(idPoint);

      coefB = yPoint - polyCoef(1)*xPoint;
      curPolyCoefCur = [polyCoef(1) coefB];

      yAll = polyval(curPolyCoefCur, compute_duration(cycleSlice(idDated), cycleSlice(1),
cycleTime));

      idKo = find(convexEnv(idDated) > yAll);
      if (isempty(idKo) || ((length(idKo) == 1) && (idKo == id)))
        basePoint = [basePoint; idPoint];
      end
    end

    if (length(basePoint) > 1)
      comment = sprintf(' %d', cycleSlice(basePoint));
      fprintf('      WARNING : %d base points (cycles %s)\n', ...
        length(basePoint), comment);
      basePoint = basePoint(1);
    end

    if (isempty(basePoint))
      fprintf('      ERROR : no base point => nothing done\n');
    else
      % compute the clock drift

      idPoint = basePoint;
      xPoint = compute_duration(cycleSlice(idPoint), cycleSlice(1), cycleTime);
      yPoint = convexEnv(idPoint);

      coefB = yPoint - polyCoef(1)*xPoint;
      polyCoefFinal = [polyCoef(1) coefB];

      clockDrift = polyval(polyCoefFinal, compute_duration(cycleSlice, cycleSlice(1), cycleTime));
      clockDrift(find(oriDates == g_dateDef)) = g_dateDef;

      idDated = find(clockDrift ~= g_dateDef);
      totalClockDrift = clockDrift(idDated(end)) - ...
        (clockDrift(idDated(1)) + compute_duration(cycleSlice(idDated(end)), cycleSlice(idDated(1)),
cycleTime));
      yearClockDrift = totalClockDrift*365/(compute_duration(cycleSlice(idDated(end)),
cycleSlice(idDated(1)), cycleTime));
      fprintf('      Clock drift (per year): %s\n', ...
        format_time(yearClockDrift*24));

      if (~isnan(yearClockDrift))
        tabClockDrift = [tabClockDrift; yearClockDrift];
        tabSavEstClockDrift(idSlice) = yearClockDrift;
      end

      idDated = find((maxEnv ~= g_dateDef) & (clockDrift ~= g_dateDef));
      fprintf('      Max diff: %s\n', ...
        format_time(max(abs(maxEnv(idDated) - clockDrift(idDated)))*24));

      tabArgosStopFinal(firstId:lastId) = clockDrift;
```

```
    end
  end
end

% same processing for slices where clock drift estimation has not been done (we
% use a weighted average of the already computed estimates)
if ((length(tabIdSlice) ~= length(tabFirstId)) && ~isempty(tabIdSlice))

  for idSlice = 1:length(tabFirstId)
    if (isempty(find(tabIdSlice == idSlice, 1)))

      firstId = tabFirstId(idSlice);
      lastId = tabLastId(idSlice);
      fprintf('   Phase2, slice #%d: cycles %d à %d\n', ...
        idSlice, firstId-1, lastId-1);
      cycleSlice = cycles(firstId:lastId);

      if (length(cycleSlice) == 1)
        continue;
      end

      % compute the mean coefficient (weighted average)
      % be careful, we must use only the coefficients available for this cycle
      % time (case of the seasonal APEXs)
      if (length(unique(cycleTime(cycleSlice+1))) == 1)
        cycTime = unique(cycleTime(cycleSlice+1));
        idMead = find(tabCycleTime == cycTime);
        if (~isempty(idMead))
          meanCoef = sum(tabCoef(idMead).*tabNbPts(idMead))/sum(tabNbPts(idMead));
        else
          continue;
        end
      end

      % last Argos message date (or last location date) of each cycle
      convexEnv = tabLastMsgDate(firstId:lastId);
      argosLastLoc = tabLastPosDate(firstId:lastId);
      idNotDated = find(convexEnv == g_dateDef);
      convexEnv(idNotDated) = argosLastLoc(idNotDated);
      oriDates = convexEnv;

      % maximum envelope without clock drift estimation
      maxEnv = convexEnv;
      idDated = find(maxEnv ~= g_dateDef);
      maxEnvDates = maxEnv(idDated);
      maxEnvCycles = cycleSlice(idDated);
      if (~isempty(maxEnvCycles))
        maxEnvDates = maxEnvDates - compute_duration(maxEnvCycles, maxEnvCycles(1),
cycleTime);
        maxDate = max(maxEnvDates);
        maxEnv(idDated) = maxDate + compute_duration(maxEnvCycles, maxEnvCycles(1),
cycleTime);
      end

      % compute the convex envelope of the LMTs
```

```
oneMore = 1;
while (oneMore)
  oneMore = 0;
  idDated = find(convexEnv ~= g_dateDef);
  if (~isempty(idDated))
    idStart = idDated(1);
    stop = 0;
    while (~stop)
      stop = 1;
      idIdDated = find(idDated > idStart);
      for id = 1:length(idIdDated)-1
        idFirst = idStart;
        xFirst = compute_duration(cycleSlice(idStart), cycleSlice(1), cycleTime);
        yFirst = convexEnv(idFirst);
        idCheck = idDated(idIdDated(id));
        xCheck = compute_duration(cycleSlice(idCheck), cycleSlice(1), cycleTime);
        yCheck = convexEnv(idCheck);
        idLast = idDated(idIdDated(id+1));
        xLast = compute_duration(cycleSlice(idLast), cycleSlice(1), cycleTime);
        yLast = convexEnv(idLast);

        coefA = (yFirst - yLast)/(xFirst - xLast);
        coefB = yFirst - coefA*xFirst;

        expValue = coefA*xCheck + coefB;
        if (yCheck <= expValue)
          % this point is not part of the convex envelope
          convexEnv(idCheck) = g_dateDef;
          % one point has been excluded, a new iteration is needed
          oneMore = 1;
        else
          % we must start from this base point (possibly in the
          % convex envelope)
          idStart = idCheck;
          stop = 0;
          break;
        end
      end
    end
  end
end

% the dates of the excluded points are set on the convex envelope
convexEnvAll = convexEnv;
idDated = find(convexEnv ~= g_dateDef);
for id = 1:length(idDated)-1
  idFirst = idDated(id);
  xFirst = compute_duration(cycleSlice(idFirst), cycleSlice(1), cycleTime);
  yFirst = convexEnv(idFirst);
  idLast = idDated(id+1);
  xLast = compute_duration(cycleSlice(idLast), cycleSlice(1), cycleTime);
  yLast = convexEnv(idLast);

  polyCoef = polyfit([xFirst xLast], [yFirst yLast], 1);
```

```
      idNew = [idFirst+1:idLast-1];
      idNew(find(oriDates(idNew) == g_dateDef)) = [];

      convexEnvAll(idNew) = polyval(polyCoef, compute_duration(cycleSlice(idNew),
cycleSlice(1), cycleTime));
    end

    % use the mean coefficient
    basePoint = [];
    idDated = find(convexEnv ~= g_dateDef);
    for id = 1:length(idDated)
      idPoint = idDated(id);
      xPoint = compute_duration(cycleSlice(idPoint), cycleSlice(1), cycleTime);
      yPoint = convexEnv(idPoint);

      coefB = yPoint - meanCoef*xPoint;
      curPolyCoefCur = [meanCoef coefB];

      yAll = polyval(curPolyCoefCur, compute_duration(cycleSlice(idDated), cycleSlice(1),
cycleTime));

      idKo = find(convexEnv(idDated) > yAll);
      if (isempty(idKo) || ((length(idKo) == 1) && (idKo == id)))
        basePoint = [basePoint; idPoint];
      end
    end

    if (length(basePoint) > 1)
      comment = sprintf(' %d', cycleSlice(basePoint));
      fprintf('      WARNING : %d base points (cycles %s)\n', ...
        length(basePoint), comment);
      basePoint = basePoint(1);
    end

    if (isempty(basePoint))
      fprintf('      ERROR : no base point => nothing done\n');
    else
      % compute the clock drift

      idPoint = basePoint;
      xPoint = compute_duration(cycleSlice(idPoint), cycleSlice(1), cycleTime);
      yPoint = convexEnv(idPoint);

      coefB = yPoint - meanCoef*xPoint;
      polyCoefFinal = [meanCoef coefB];

      clockDrift = polyval(polyCoefFinal, compute_duration(cycleSlice, cycleSlice(1), cycleTime));
      clockDrift(find(oriDates == g_dateDef)) = g_dateDef;

      idDated = find(clockDrift ~= g_dateDef);
      totalClockDrift = clockDrift(idDated(end)) - ...
        (clockDrift(idDated(1)) + compute_duration(cycleSlice(idDated(end)),
cycleSlice(idDated(1)), cycleTime));
      yearClockDrift = totalClockDrift*365/(compute_duration(cycleSlice(idDated(end)),
cycleSlice(idDated(1)), cycleTime));
```

```
          fprintf('      Clock drift (per year): %s\n', ...
            format_time(yearClockDrift*24));

          idDated = find((maxEnv ~= g_dateDef) & (clockDrift ~= g_dateDef));
          fprintf('      Max diff: %s\n', ...
            format_time(max(abs(maxEnv(idDated) - clockDrift(idDated)))*24));

          tabArgosStopFinal(firstId:lastId) = clockDrift;
        end
      end
    end
end

% ------------------------------------------------------------------------------
% Compute the duration between 2 cycles.
%
% SYNTAX :
%  [o_duration] = compute_duration(a_tabEndCyNum, a_startCyNum, a_cycleTime)
%
% INPUT PARAMETERS :
%   a_tabEndCyNum : final cycle numbers
%   a_startCyNum  : first cycle number
%   a_cycleTime   : cycle durations
%
% OUTPUT PARAMETERS :
%   o_duration : computed durations
%
% EXAMPLES :
%
% SEE ALSO :
% AUTHORS  : Jean-Philippe Rannou (Altran)(jean-philippe.rannou@altran.com)
% ------------------------------------------------------------------------------
% RELEASES :
%   27/10/2009 - RNU - creation
% ------------------------------------------------------------------------------
function [o_duration] = compute_duration(a_tabEndCyNum, a_startCyNum, a_cycleTime)

global g_durationDef;

% default values initialization
init_valdef;

o_duration = ones(length(a_tabEndCyNum), 1)*g_durationDef;

for id = 1:length(a_tabEndCyNum)
  % number of the cycles for which the duration is wanted
  cyNum = [a_startCyNum+1:a_tabEndCyNum(id)];
  if (~isempty(cyNum))
    % duration computation
    o_duration(id) = sum(a_cycleTime(cyNum+1));
  else
    o_duration(id) = 0;
  end
end
```

o_duration = o_duration/24;

return;

3.3.3 Final improvement: taking the cycle duration anomalies into account

Some Apex floats have experienced anomalies in their cycle duration, an example is provided in next figure.

These floats have been processed by slices.

A new slice is defined for each set of cycles with a constant cycle duration. Each slice is then processed with the proposed method.

This is also the way we have processed DPF floats. We created a first slice with cycle #0 and a second one with all the other cycles (see Figure 22).

Within the 3622 APEX Argos floats of the ANDRO data set, 777 floats have been processed by slices: 717 floats only because they are DPF floats and 60 floats because they have cycle duration anomalies.

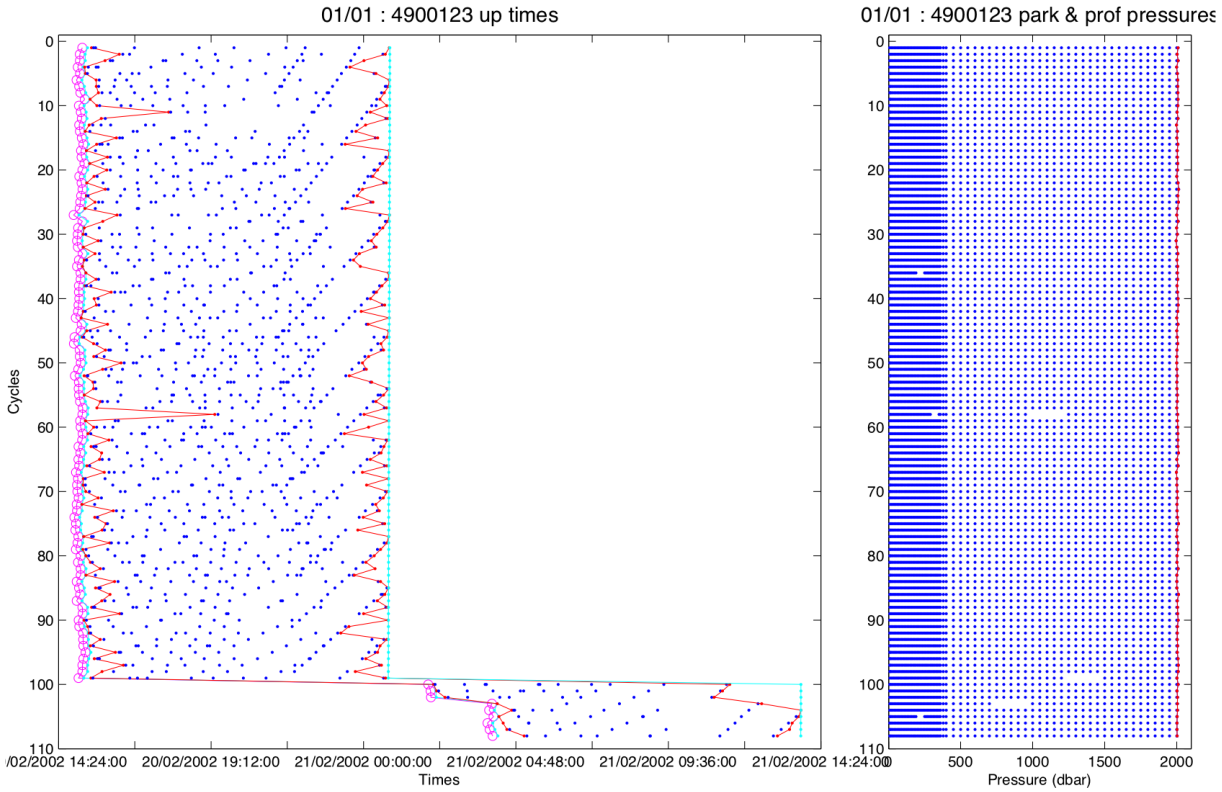


Figure 21: Example of a cycle duration anomaly (cycle #100 duration is 250 hours whereas others cycles have the expected duration i.e. 240 hours)

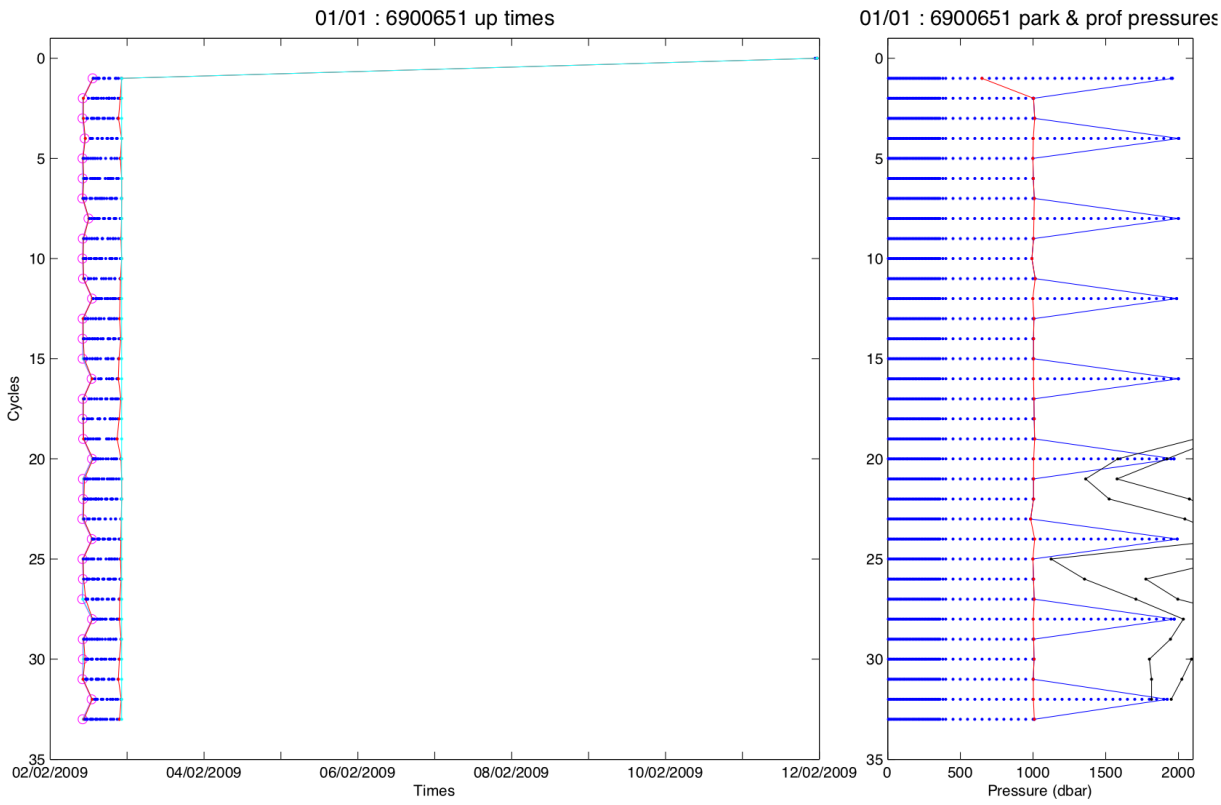


Figure 22: Example of DPF float, the first profile is a deep profile (whereas with a PnP parameter of 4, it should otherwise be a shallow profile) and the first cycle is shorter than the other ones

3.3.4 Results obtained in the ANDRO data set

For the processing of the ANDRO data set we have chosen to estimate the float clock offset only if we have more than 20 points to fit in step #4; thus only if we have received at least 33 cycles for the float.

The estimated TETs of the 3622 floats have then been visually checked and the following parameter modifications done:

- The estimation using the second algorithm has been canceled for 53 floats (i.e. we use the first algorithm results even if more than 33 cycles has been received). In most of these cases, bad results are due to too few cycles used in the estimate of the clock offset (between 21 and ~40),
- For 191 floats, the process needed additional customization (modification of the number of deleted points in step #3).

Thus we are convinced that the proposed method is robust enough to be implemented in real time (except for cycle duration anomalies which can only be detected by visual inspection).

Some Apex float versions provide the time of the end of the DOWN TIME period. We have estimated the float clock offset from these times and successfully compared it with the one obtained by the second algorithm.

Thus we are convinced that the second algorithm method is reliable for TETs estimation.

Most of the estimated clock offsets are in the [-00:10:00; +00:10:00] interval (the [-00:2:30; +00:00:30] interval for new floats) and they imply corrections of less than 80 minutes.

3.3.5 Recommended method for real time processing

To estimate the TETs of an APEX Argos float you need to know:

- Its theoretical CYCLE TIME duration(s),
- If the float is a DPF float.

If we think that clock offset can be neglected for real time processing, we recommend using only the first algorithm.

If not, we recommend using both algorithms:

- First algorithm when we have received less than 33 cycles from the float,
- Second algorithm when we have received at least 33 cycles from the float.

When using the second algorithm, if the absolute value of the estimated clock offset is greater than 00:20:00 per year, we must be sure that the float is not a DPF one; otherwise the float has probably experienced a cycle duration anomaly and the TETs should not be estimated in real time (neither by the first nor by the second algorithm).

The value 33 should be discussed.

4 ANNEX C: Computing Transmission Start Time for and APEX Argos float

The number of Argos messages needed to transmit the data collected at depth can vary between cycles (it mainly depends on profile length).

Starting when the float arrives at the surface, these *M* Argos messages are transmitted sequentially (from #1 to #*M*) and repeatedly until the end of the UP TIME period.

If a complete set of the *M* message is called a **block** of data, thus *B* blocks of *M* messages are transmitted.

Note however that the last block is not necessarily complete (because the transmission stops at the end of the UP TIME, not at the end of a block).

All Argos messages are numbered. Moreover, message #1 gives the block number.

Since all messages received by the ARGOS satellite are dated, we get the times of transmission of the messages received and their numbers.

From messages #1, we get also the block numbers to which they belong.

4.1 Teledyne Webb Research proposed method

This method is explained in the APEX user's manual and illustrated in the following figure.

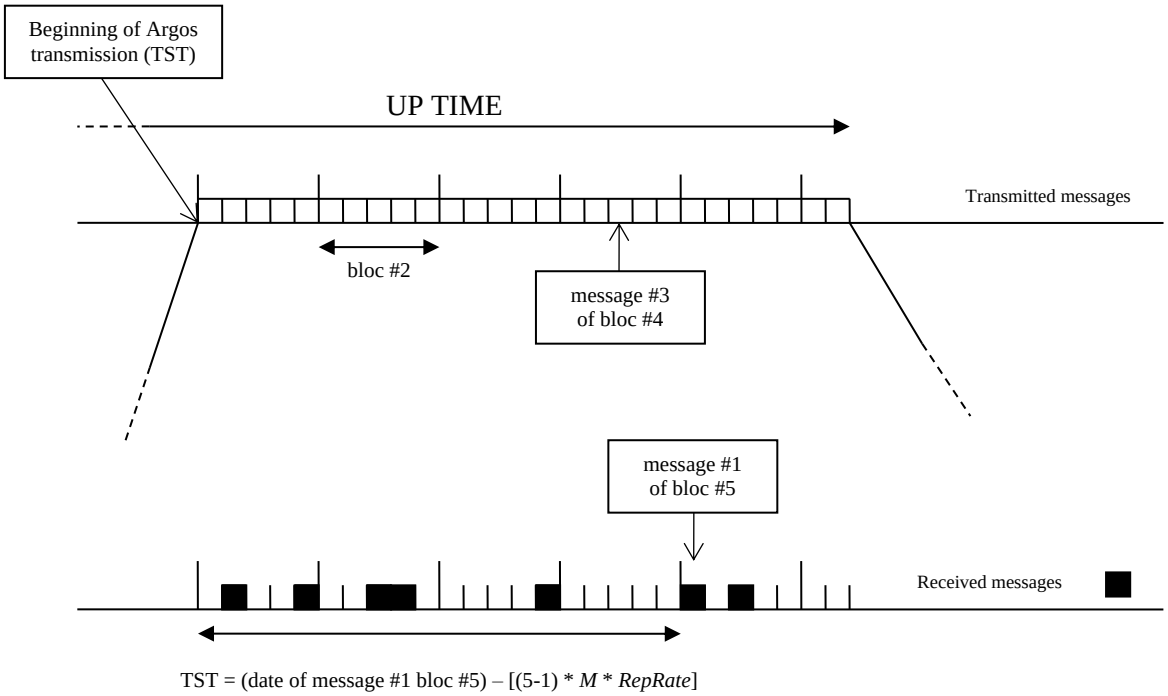


Figure 23: Teledyne Webb Research method to compute TST for an APEX Argos float

If **at least one message #1** is received, its corresponding block number (B_N) can be determined.

The number of transmitted messages since TST is then: (B_N-1)*M and

TST = (Argos time of received message #1) - [(B_N-1)*M*$RepRate$]

where *RepRate* is the period of the float Argos PTT.

This method thus implies:

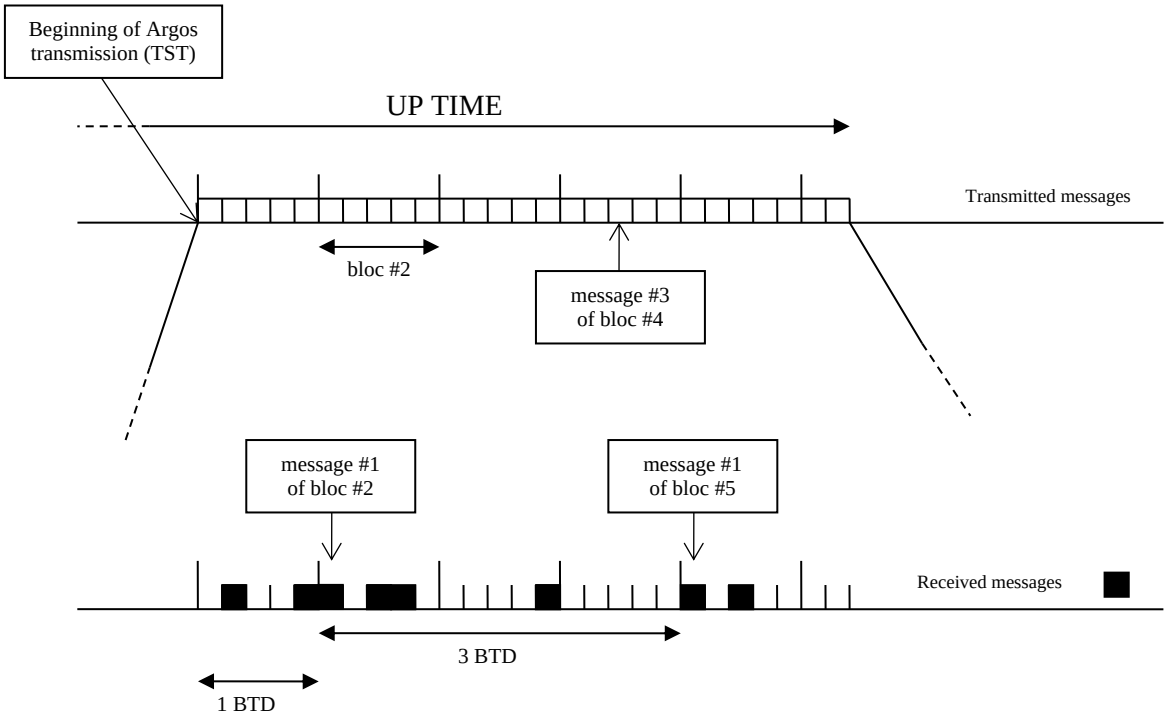
- To receive at least one message #1,
- To know the period of the Argos PTT (*RepRate*),
- To know the total number of transmitted Argos messages (*M*).

The *RepRate* parameter is a meta-data variable, thus possibly erroneous (it can however be checked from received message times).

The *M* value can sometimes be difficult to compute. Each float format must be carefully studied; *M* can be computed from the variable parts of the Argos messages (i.e. profile length but also to the number of PTS measurement sampled during the drift phase).

4.2 An improved proposed method

To get rid of the *M* value determination, a second method can be used, as illustrated in the following figure.



BTD = Block Transmission Duration = [(date of message #1 bloc #5) – (date of message #1 bloc #2)]/(5-2)

Figure 24: An improved method to compute TST for an APEX Argos float

If **at least two messages #1** (dated *T1* and *T2* respectively) **belonging to two different blocks** (B_{N1} and B_{N2} respectively) are received, the transmission duration of a block can be determined:

Transmission Duration of one Block =BTD = ($T2$ - $T1$)/(B_{N2} - B_{N1})

But ***BTD* = *M**RepRate***, then

TST = *T1* - [(B_{N1}-1)BTD*] or TST = *T2* - [(B_{N2}-1)**BTD*]**

The following strategy is then suggested:

1. If at least two messages belonging to two different blocks are received, use the improved method.
   In this case all (in fact at most 100) values of TST are computed (from all combinations of *Ti* and *Tj* messages #1 received) and the most redundant result is chosen for TST.
   At the beginning of our work, we started to compute all the values of TST but in case of shallow profiles (or no profile at all when the float stays at the surface) i.e. when the M value is only 1 or 2 and then B value is very important, the method doesn't work (the histogram of TST computed values has many redundant values and the most redundant one can be inconsistent). We have not understood why (it can be an interesting question for TWR) but we think this is due to inconsistencies in the transmitted data, we thus decided to compute "only" 100 values of the TST and to choose the most redundant value.
2. If only one message #1 is received, the TWR method is used with the assumption (sometimes erroneous) that *M* is equal to the maximal number of the received Argos messages (i.e. that the last Argos message has been received).

5 ANNEX D: Apex float vertical velocities

5.1 APEX float descending velocity

To estimate the descending APEX Argos velocity, 463 APEX floats which provide pressure marks hourly sampled during the descent to PARKING depth were analyzed.

At most 7 pressure marks are transmitted by the floats. Thus, at most 6 averaged hourly descent rates were computed (remember that the first pressure mark is sampled at the end of the piston retraction, thus it is not dated and useless for this estimation), and a global averaged descent rate for the 6 first hours of the descent.

The mean descent rate depends on the float PARKING pressure; the averages for 5 parking depths (250 dbar, 500 dbar, 1000 dbar, 1500 dbar and 2000 dbar) were computed. The RPP has been used to associate a descent rate to the corresponding depth.

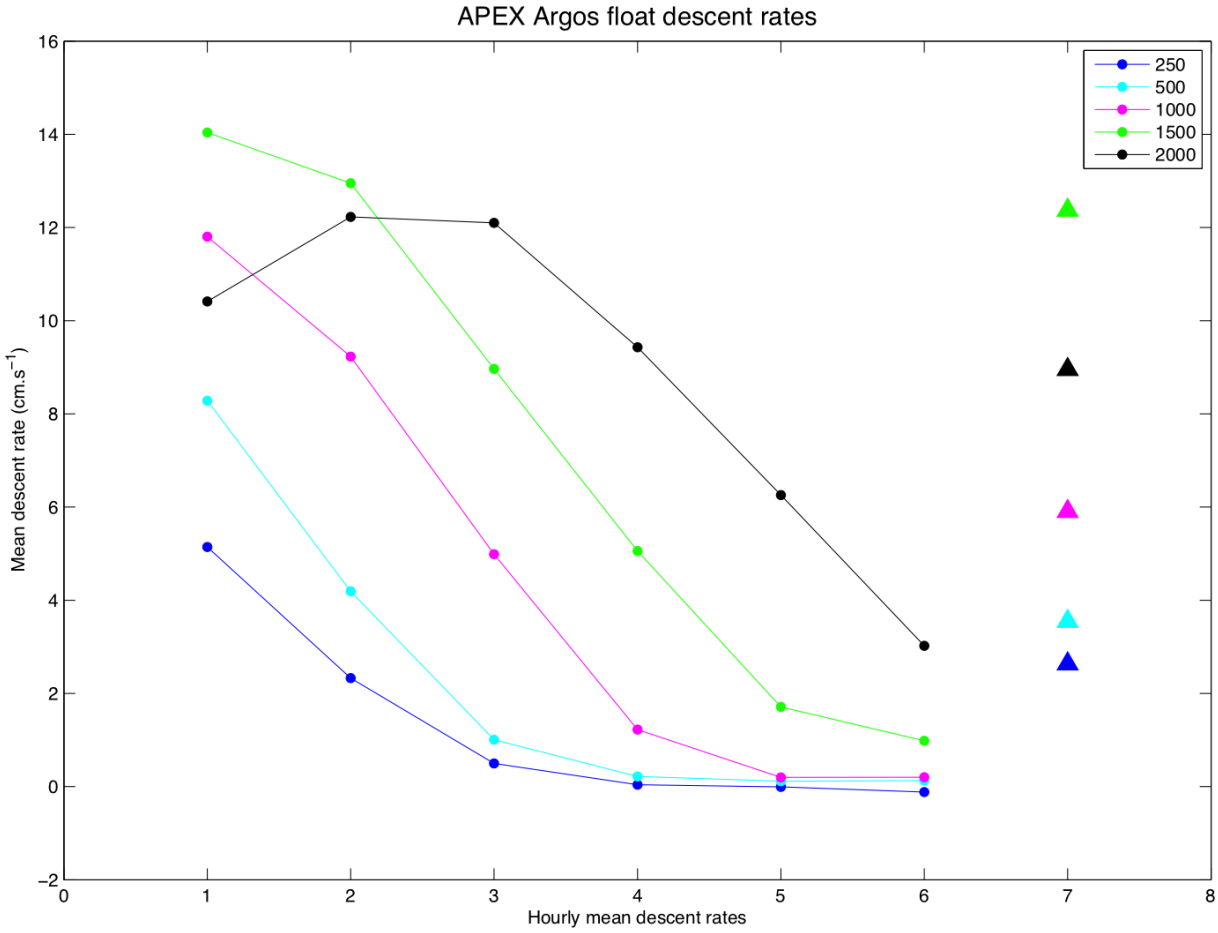


Figure 25: Mean hourly descent rates of APEX Argos floats. For each PARKING pressure the 6 mean hourly descent rates (dots) and the global mean descent rate of the 6 first hours of the descent (triangle) are shown.

The following table gives the number of samples used to compute the averages.

	1st hour	2nd hour	3rd hour	4th hour	5th hour	6th hour	Global
250 dbar	373	333	250	208	155	86	1 405
500 dbar	580	549	526	471	345	197	2 668
1000 dbar	15 010	14 852	14 762	14 646	8 762	3 827	71 859
1500 dbar	389	144	141	119	91	52	936
2000 dbar	2 114	2 105	2 091	2 059	2 040	1 981	12 390

Table 2: Number of samples used to compute mean descent rates

The global mean descent rate values are provided in the following table.

PARKING depth	250 dbar	500 dbar	1000 dbar	1500 dbar	2000 dbar
Mean descent rate ($cm*s^{-1}$)	2.64	3.55	5.91	12.37	8.95
Associated standard deviation	1.87	2.54	1.23	3.02	1.25

Table 3: Global mean descent rate values

5.2 APEX float ascending velocity

To estimate the ascending APEX Argos velocity, the two following data samples were used:

- The 298 APEX floats which provide the AST_{float} and TST_{float} (thus AET_{float}) (see §2.2.4.9 and 2.2.4.13),
- The 1497 APEX floats for which the following are available:
  - The AST, estimated as TET - UP TIME for cycles where PARKING and PROFILE pressure are equal (see §2.2.4.8)
  - The AET, computed from TST obtained with the method explained in Annex C (see §4.2)

The data of the first set are more reliable because they are measured and transmitted by the floats (whereas the data of the second set come from estimations).

To compute a mean ascent rate, a reliable deepest profile pressure value is needed. For that, it is important that the Argos message of the first (deepest) profile bin PTS measurement has been received. This information can be provided by the APEX Argos decoders.

As this information was not stored in the DEP data set, only profiles for which the following is true could be used:

| deepest bin pressure - PROFILE pressure | < 150 dbar.

(these data are in green in the following figures).

If | deepest bin pressure - PROFILE pressure | > 150 dbar, the cycle can be flagged "grounded" (blue stars) or not (red stars).

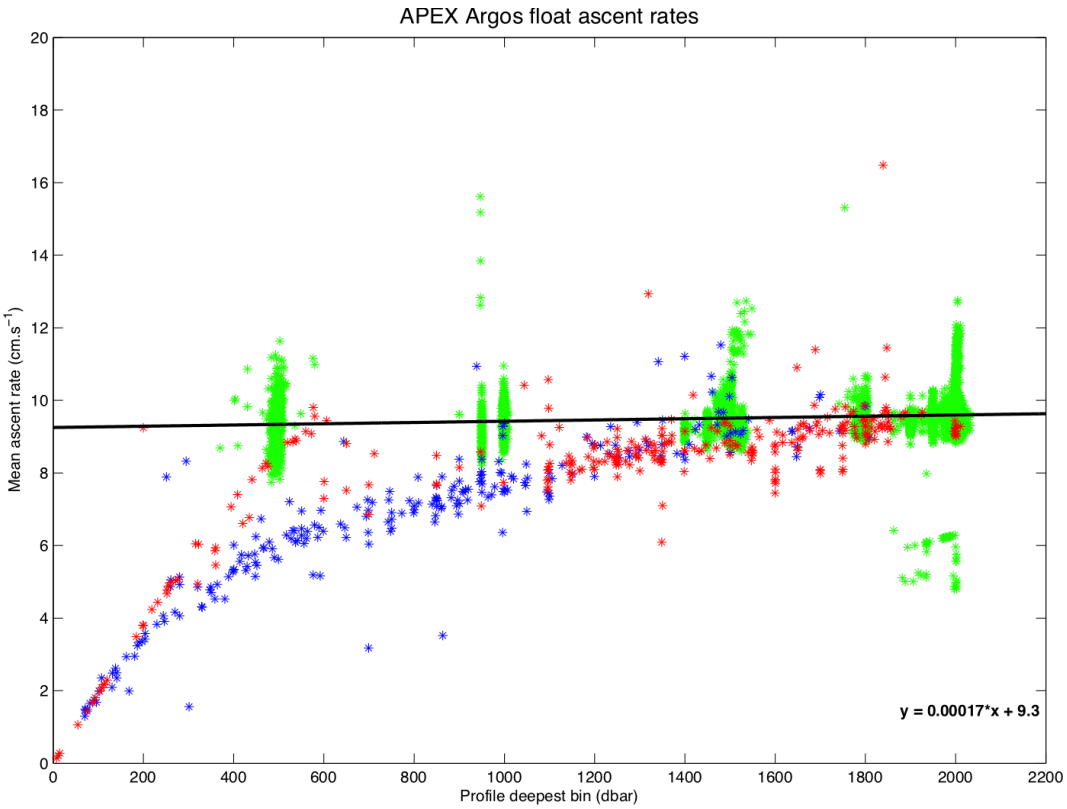


Figure 26: Ascent rates computed from measured data

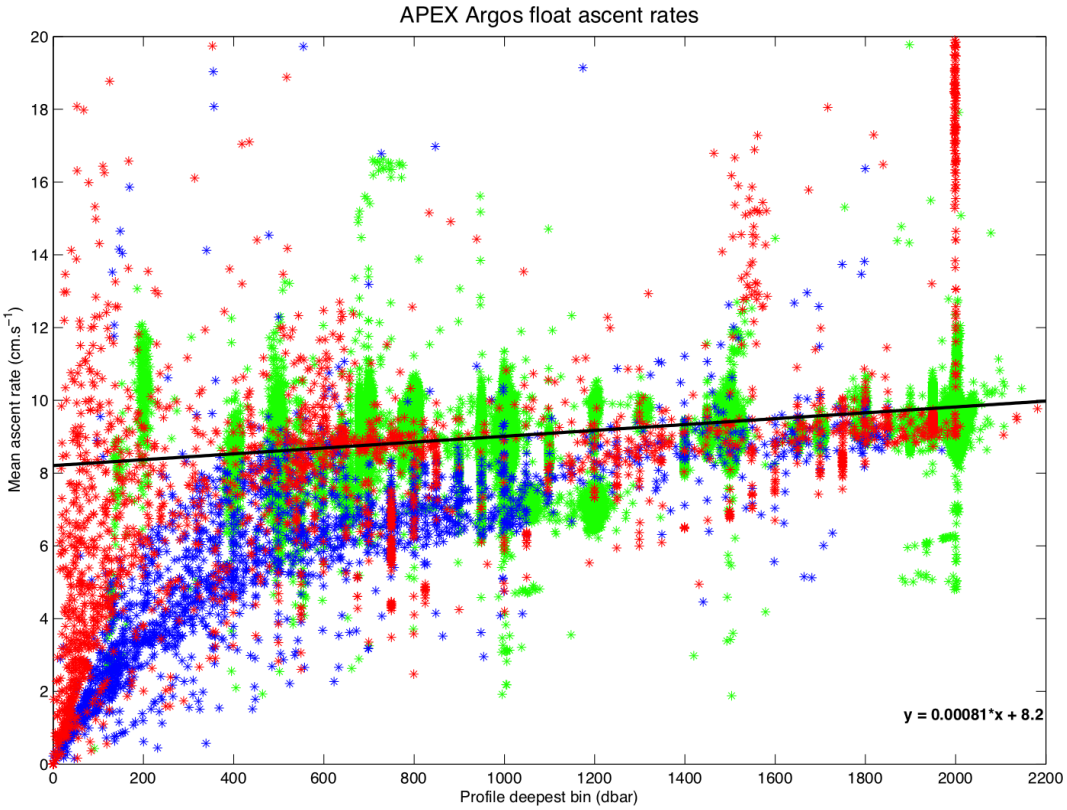


Figure 27: Ascent rates computed from estimated data

In the first figure, the green stars can be linearly fitted by the Y = 0.00017*X + 9.3 function.

In the second figure, the green stars can be linearly fitted by the Y = 0.00081*X + 8.3 function.

Thus the mean ascent rate is around 9.5 cm/s in the first case and between 9.0 and 10 cm/s in the second case.

Therefore, it is recommended to use **a mean ascent rate of 9.5 cm/s for APEX Argos floats**.

6 ANNEX E: Input parameters

Most of the specifications given in this document need input parameters. These parameters are part of the values used to program the float mission.

Unfortunately these values can be difficult to collect; some of them are transmitted by the instrument (in APEX float test message or PROVOR/APEX Iridium technical message) but the others can only be found in float operator notes.

It is thus important to ask each float PI to collect the programmed float parameters and to send them to the concerned DAC.

Some input parameters, gathered in the framework of the ANDRO project, are joined in electronic form to this document.

In the Excel file *CorrectedMetadata.xlsx* you can find, for the 5967 ANDRO floats, the corrected meta-data values for:

- Float PTT,
- All existing float missions:
  A mission is defined by:
  - Its repetition rate,
  - Its duration, given by:
    - The UP_TIME and DOWN_TIME period for APEX floats,
    - The CYLE_TIME for other floats.
  - Its parking pressure,
  - Its profile pressure,
- The float launch time and position,
- The startup time of the float.

In the Excel file *_provor_floats_information.xls*, you can find the corrected meta-data values used to decode the PROVOR Argos floats.

You can find in particular:

- The DELAI parameter values (maximum amount of time given to the float for diving from PARKING to PROFILE depth),
- The reference day ("day of the first descent") used to decode the transmitted times.

In the Excel file *DPDP_values.xlsx*, you can find the decoded values of the "Deep profile descent period" transmitted in the test message by some version of APEX floats.

7 ANNEX F: Measurement code table

7.1 General Measurement Code Table Key

Measurement code type	Definition
Any code evenly divisible by 100 (e.g. 100, 200, 300, etc.)	**Primary Measurement Codes (MC)**. Each marks a mandatory-to-fill cycle timing variable. These are very important for determining trajectory estimates. All are found in both the N_MEASUREMENT and N_CYCLE data arrays.
Any code evenly divisible by 50 but not evenly divisible by 100 (e.g. 150, 250, 450, et)	**Secondary Measurement Codes (MC)**. Each marks a suggested-to-fill cycle timing variable. Secondary MC are not always applicable to all floats, but are very useful in determining trajectory estimates.
Any code that falls in between any Primary or Secondary Measurement Code (span of 50 values). These codes describe data that are important cycle timing information but are not as important as the primary or secondary timing variables. The value span is subdivided into two halves. Measurement codes in this section will be described relative to the values of the Primary and Secondary codes.	**Relative Generic Codes**. Values spanning from MC minus 24 to MC minus 1: Measurement codes that have lower value and within 24 of a Primary or Secondary Measurement Code. These code definitions are phrased generally, so can be attached to data from many different floats. These code values (MC minus 24 to MC minus 1) are assigned when a float records a measurement while transitioning TOWARDS the MC. The definitions of the MC from MC minus 24 to MC minus 1 are repeated for all Primary and Secondary MC. An example, most floats record pressure/temperature/salinity during drift. The float is transitioning towards PET (MC=300) during this period. Thus the pressure/temperature/salinity measurements will have an MC between MC minus 24 and MC minus 1 where MC=300 (thus between MC=276 and MC=299). Which value is chosen is determined by the measurement itself (See table below). **Relative Specific Codes**. Values spanning from MC plus 1 to MC plus 25: These are specific measurements that are generally NOT recorded by multiple float types. They are believed to be valuable enough in trajectory estimation that they are defined here, and not within the generically defined MC minus 24 to MC minus 1 span. MC codes in this span will be specific to the MC code, and will NOT be repeated for other Primary and Secondary MCs. An example, APEX floats report the "Down-time end date", which is important in determining the start of ascent (MC=500). The MC for "Down-time end date" is recorded with MC plus 1 (MC=501).

7.2 Relative Generic Code Table Key (from MC minus 24 to MC minus 1)

This table pertains to any measurement code that has lower value and within 24 of a Primary or Secondary Measurement Code (see below). These definitions apply relative to every Primary and Secondary code. For example, AST (time of ascent start, MC=500) and AET (time of ascent end, MC=600) are both Primary MCs. There exists a measurement code MC minus 4 for both AST and AET which is assigned to any averaged measurement that is taken while transitioning towards the MC. If an averaged measurement is recorded while transitioning towards AST, the correct MC=496. If an averaged measurement is recorded while transitioning towards AET, the correct MC=596.

Relative Measurement code	Meaning
MC minus 1	Any single measurement transitioning towards MC (see MC-10 for a 'series' of measurements)
MC minus 2	Maximum value while float is transitioning towards an MC (e.g. pressure)
MC minus 3	Minimum value while float is transitioning towards an MC (e.g. pressure)
MC minus 4	Any averaged measurements made during transition to MC
MC minus 5	Median value while float is transitioning towards an MC
MC minus 6	Standard deviation of measurements taken during transition towards an MC
MC minus 7 to MC minus 9	currently unassigned
MC minus 10	Any "series" of measurements recorded while transitioning towards MC. (e.g. Provor 'spy' measurements, SOLOII pressure-time pairs, etc).
MC minus 11	Active adjustment to buoyancy made at this time
MC minus 12	Any supporting measurements for the maximum value (MC minus 2)

MC minus 13	Any supporting measurements for the minimum value (MC minus 3)
MC minus 14	Any supporting measurements for the averaged value (MC minus 4)
MC minus 15	Any supporting measurements for the median value (MC minus 5)
MC minus 16 to MC minus 24	currently unassigned
MC 600, 700 or 800 plus 10	Any in-water samples taken as part of a surface sequence while transitioning towards an MC. (e.g. O2 samples taken prior to air-bladder inflation or max buoyancy as part of in-air measurement sequence)
MC 600, 700, or 800 plus 11	Any in-air samples taken as part of a surface sequence while transitioning towards an MC. (e.g. O2 samples taken after in air-bladder inflation or max buoyancy as part of in-air measurement sequence)

7.3 Measurement Code Table

Measure-ment code	Variable	Meaning	Transmitted by listed float type. Value can be estimated in other floats
0		Launch time and location of the float	All float types
76-99	see above table	Any measurement recorded during transition towards DST	
100	**DST**	**All measurements made when float leaves the surface, beginning descent.** Time (JULD_DESCENT_START)	Time: PROVOR, ARVOR, SOLO-II,WHOI SOLOIR, NEMO, NEMOIR,APEX APF9, APEXIR APF9, Deep NINJA, NAVIS
101	DM Traj file only	This MC is used in the DM Traj file when new cycles have been recovered during DM operations (the TECH file, where surface pressure measurements usually belong, is not updated during TRAJ DM). The PRES variable should contain the measurement provided by the float (after 5dbar subtraction when needed). For APEX floats, this measurement is used to compute (see procedure 3.2.1 of Argo QC manual) a pressure offset applied to all pressure measurements. This offset should be stored in the PRES_ADJUSTED variable. No information should be stored with this MC in Real Time.	
102-125	unassigned	Reserved for specific timing events around DST.	
126-149	see above table	Any measurement recorded during transition towards FST	
150	**FST**	**All measurements made at time when a float first becomes water-neutral.** Time (JULD_FIRST_STABILIZATION)	PROVOR, ARVOR
151-175	unassigned	Reserved for specific timing events around FST.	
176-199	see above table	Any measurement recorded during transition towards DET	
200	**DET**	**All measurements made at time when float first approaches within 3% of the eventual drift pressure.** Float may be transitioning from the surface or from a deep profile. This variable is based on measured or estimated pressure only In the case of a float that overshoots the drift pressure on descent, DET is the time of the overshoot. Time (JULD_DESCENT_END)	Time: PROVOR, ARVOR, SOLO-II, NEMO, NEMOIR, DeepNINJA
201-202 & 204-225	unassigned	Reserved for specific timing events around DET.	
203		Deepest bin reached during descending profile	
226-249	see above table	Any measurement recorded during transition towards PST	

250	**PST**	**All measurements made at time when float transitions to its Park or Drift mission**. This variable is based on float logic based on a descent timer (i.e. SOLO), or be based on measurements of pressure (i.e. Provor). Time(JULD_PARK_START)	APEX non APF9, APEX APF9, APEX APF9i, SIO SOLO, SOLO-II, NEMO, NEMOIR, NAVIS CTD: WHOI SOLO NINJA
251-274	unassigned	Reserved for specific timing events around PST.	
275	RAFOS	RAFOS positions and times determined during drift	All float types with RAFOS capabilities
276-299	see above table	Any measurement recorded during transition towards PET	
300	**PET**	**All measurements made at time when float exits from its Park or Drift mission.** It may next rise to the surface (AST) or sink to profile depth Time (JULD_PARK_END)	Time: PROVOR (excluding PROVOR MT), ARVOR, SOLO-II, NEMO, NEMOIR, POPS, NAVIS CTD: WHOI SOLO
301		Representative Park <PARAM> found either from measurements taken during drift or from metafile information	
302-325	unassigned	Reserved for specific timing events around PET.	
376-399	see above table	Any measurement recorded during transition towards DDET	
400	**DDET**	**All measurements made at time when float first approaches within 3% of the eventual deep drift/profile pressure**. This variable is based on pressure only and can be measured or estimated. Time (JULD_DEEP_DESCENT_END)	Time: APEX APF9a or APF9t, APF9i, PROVOR CTS3, ARVOR, SOLO-II, POPSm , DeepNINJA, NAVIS
401-425	unassigned	Reserved for specific timing events around DDET.	
426-449	see above table	Any measurement recorded during transition towards DPST	
450	**DPST**	**All measurements made at time when float transitions to a deep park drift mission.** This variable is only defined if the float enters a deep drift phase (i.e. DPST not defined in cases of constant deep pressure due to bottom hits, or buoyancy issues).	
451-475	unassigned	Reserved for specific timing events around DPST.	
476-499	see above table	Any measurement recorded during transition towards AST	
500	**AST**	**All measurements made at the start of the float's ascent to the surface** Time (JULD_ASCENT_START)	Time: APEX APF9, PROVOR, ARVOR, SOLO-II, NEMO, NEMOIR, POPS, DeepNINJA, NAVIS
501		Down-time end time: end date of the down-time parameter reported by APEX floats	APEX
502		Ascent start time directly provided by old APEX Argos floats. See 2.2.4.9	APEX
503		Deepest bin reached during ascending profile	
504-525	unassigned	Reserved for specific timing events around AST.	
526-549	see above table	Any measurement recorded during transition towards DAST	
550	**DAST**	**All measurements made at the start of the float's ascent from profile pressure to drift pressure**. Used for floats that profile on descent and then move back up to drift pressure. Time (JULD_DEEP_ASCENT_START)	Time: Deep SOLO-II
551-575	unassigned	Reserved for specific timing events around DAST.	
576-599	see above	Any measurement recorded during transition	

	table	towards AET	
600	**AET**	**All measurements made at the end of ascent.** Time (JULD_ASCENT_END)	PROVOR, ARVOR, SOLO-II, NEMO, NEMOIR, POPS, DeepNINJA, NAVIS
601-625	unassigned	Reserved for specific timing events around AET.	
676-699	see above table	Any measurement recorded during transition towards TST	
700	**TST**	**Time of the start of transmission for the float.** Time (JULD_TRANSMISSION_START)	APEX APF9, APEXIR APF9, PROVOR, ARVOR, SOLO-II, NEMO, NEMOIR, POPS, DeepNINJA, NAVIS
701		Transmission start time directly transmitted by APEX float	APEX
702	**FMT**	**Earliest time of all messages received by telecommunications system** – may or may not have a location fix. Time (JULD_FIRST_MESSAGE)	All Argos floats. Iridium floats that send timing of messages. This includes SBD Iridium floats.
703	**FLT**	First location available	All floats
703		Surface times and locations (if available) during surface drift. Should be listed in chronological order.	All floats
703	**LLT**	Last location available	All floats
704	**LMT**	**Latest time of all messages received by telecommunications system** – may or may not have a location fix. Time (JULD_LAST_MESSAGE)	All Argos floats. Iridium floats that send timing of messages. This includes SBD Iridium floats.
705-725	unassigned	Reserved for specific timing events around TST	
776-799	see above table	Any measurement recorded during transition towards TET	
800	**TET**	**Time and location of the end of transmission for the float.** Time (JULD_TRANSMISSION_END)	PROVOR, ARVOR, SOLO-II,, DeepNINJA Time only: APF9i and APF11
801-825	unassigned	Reserved for specific timing events around TET	
901		Grounded flag	
902		Last time before float recovery. For floats that have been recovered, it is important to know when this occurred. This time in the JULD array will be the last time before the float was recovered. Determined by inspection of data	
903		Pressure offset used to correct APEX pressure measurements	APEX

8 ANNEX G: Implementation of the JAMSTEC trajectory quality control method

The JAMSTEC trajectory quality control method is described in Nakamura et al (2008), "Quality control method of Argo float position data", JAMSTEC Report of Research and Development, Vol. 7, 11-18 (http://www.godac.jamstec.go.jp/catalog/data/doc_catalog/media/JAM_RandD07_02.pdf).

This method checks the surface trajectory of an Argos float by considering the speeds induced by the successive Argos fixes. The test can flag the surface position as '3' or '4'.

In the following, we propose a detailed description of the algorithm to implement.

8.1 Inputs

The inputs of the algorithm are:

- The surface trajectory to be checked (N Argos location dates, latitudes, longitudes and classes),
- The last good (flagged as '1') surface location of the (already checked) previous (received) cycle.

8.2 Algorithm

Assuming that the location dates have not been flagged as bad by the test #2 "Impossible date test", we first chronologically sort the surface positions.

The whole surface trajectory is used to initialize the (checked) current trajectory.

The current trajectory is processed in an infinite loop in which the following steps are performed:

8.2.1 Step 1

The subsurface drift speed is computed between the last good surface position of the previous cycle and the first position of the current trajectory.

If this speed is greater than 3 m/s, the first position of the current trajectory is flagged as '4', this position is then excluded from the current trajectory and a new iteration of the infinite loop starts.

8.2.2 Step 2

Speeds are computed for the second position to the last position of the current trajectory. Each speed is computed between position #i-1 and position #i and affected to position #i.

In case of duplicated positions (i.e. if position #i-1 and position #i have the same latitude, longitude and date): the position #i is flagged as '4', it is then excluded from the current trajectory and a new iteration of the infinite loop starts.

In case of an erroneous cycle number of the position #i (i.e. if the times difference between position #i and position #i-1 is greater than one day): the position #i is flagged as '4', it is then excluded from the current trajectory and a new iteration of the infinite loop starts.

8.2.3 Step 3

The position #iMax is found as the position with the maximum speed.

If this maximum speed is greater than 3 m/s, the position #iMax is 'questionable' and the **speed test** (see §8.3) is performed on it over the current trajectory.

The **speed test** should lead to define position #iMax or/and position #iMax-1 as 'abnormal'.

8.2.4 Step 4

If the **distance test** (see §8.4) between position #iMax and position #iMax-1 is verified, the 'abnormal' position(s) is (are) flagged as '3'.

The 'abnormal' position(s) is(are) then deleted from the current trajectory (even when the distance test is not verified) and a new iteration of the infinite loop starts.

The infinite loop ends when no 'abnormal' position has been detected or when the current trajectory has less than 2 positions.

8.3 Speed test

The speed test is performed on a 'questionable' position over a given trajectory.

The 'questionable' position (called B in the following) can be all but the first position of the trajectory. The position which precedes B on the trajectory is called A in the following.

8.3.1 Case of different Argos classes

If positions A and B have different Argos classes, the position with the less accurate Argos class is defined as 'abnormal' by the speed test.

Remember that the accuracy of the Argos location classes is the following:

more accurate <= 3, 2, 1, 0, A, B, Z => less accurate

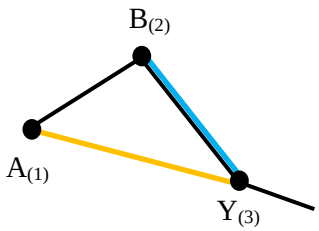
8.3.2 Case of identical Argos classes

If positions A and B have the same Argos classes:

- If the trajectory only comprises the two positions A and B, both positions are defined as 'abnormal' by the speed test,
- Otherwise the speed test depends on the position of the location B on the trajectory, 3 cases are possible.

Case 1: If B is the second position of the trajectory

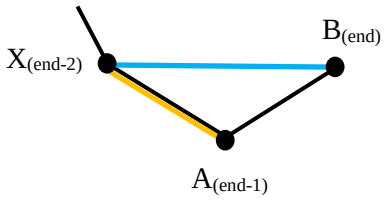
In this case: A is the first position, B the second one and there is a position Y following the position B on the trajectory.



Speeds on the segments A-Y (orange) and B-Y (blue) are computed: if $speed_{A-Y}$ is greater than $speed_{B-Y}$, the position A is defined as 'abnormal' by the speed test otherwise B is defined as 'abnormal' by the speed test.

Case 2: If B is the last position of the trajectory

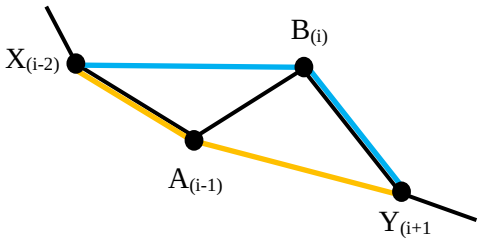
In this case: A is the last but one position, B is the last position and there is a position X preceding the position A on the trajectory.



Speeds on the segments X-A (orange) and X-B (blue) are computed: if $speed_{X-A}$ is greater than $speed_{X-B}$, the position A is defined as 'abnormal' by the speed test otherwise B is defined as 'abnormal' by the speed test.

Case 3: we are not in case 1 or 2

In this case: there is a position X preceding the position A on the trajectory and a position Y following the position B on the trajectory.



Speeds on the segments X-A-Y (orange trajectory) and X-B-Y (blue trajectory) are computed. If $speed_{X\text{-}A\text{-}Y}$ is greater than $speed_{X\text{-}B\text{-}Y}$, the position A is defined as 'abnormal' by the speed test otherwise B is defined as 'abnormal' by the speed test.

8.4 Distance test

The distance test is performed on two Argos locations A and B.

The distance test is verified if the distance between locations A and B is greater or equal to

$1.0 \times \sqrt{{Er_A}^2 + {Er_B}^2}$ where Er_A and Er_B are the radii of position error for locations A and B respectively.

These position errors, deduced from the position classes, are 150 m, 350 m and 1000 m for Argos class 3, 2 and 1 respectively. Moreover we have associated a position error of 1500 m, 1501 m, 1502 m and 1503 m for Argos classes 0, A, B and Z respectively.

8.5 Distance computation

As far as distance and speed are concerned in this trajectory QC method, we must specify an algorithm to compute distance between positions of the surface trajectory. This algorithm must be common to all the DACs so that the trajectory QC results will not depend on DAC's distance computation method.

We propose to use the distance algorithm from the *Laboratoire de Physiques des Océans* (*LPO*) at IFREMER.

This algorithm computes distance between points on the earth using the WGS 1984 ellipsoid, its Matlab implementation and some test points are provided below.

8.5.1 Matlab implementation of the LPO distance algorithm

```
function [range, A12, A21] = distance_lpo(lat, long)
%
% Computes distance and bearing between points on the earth using WGS 1984
% ellipsoid
%
% [range, A12, A21] = distance_lpo(lat, long) computes the ranges RANGE between
% points specified in the LAT and LONG vectors (decimal degrees with positive
% indicating north/east). Forward and reverse bearings (degrees) are returned
% in AF, AR.
%
% Ellipsoid formulas are recommended for distance d<2000 km,
% but can be used for longer distances.
%
% GIVEN THE LATITUDES AND LONGITUDES (IN DEG.) IT ASSUMES THE IAU SPHERO
% DEFINED IN THE NOTES ON PAGE 523 OF THE EXPLANATORY SUPPLEMENT TO THE
% AMERICAN EPHEMERIS.
%
% THIS PROGRAM COMPUTES THE DISTANCE ALONG THE NORMAL
% SECTION (IN M.) OF A SPECIFIED REFERENCE SPHEROID GIVEN
```

```
% THE GEODETIC LATITUDES AND LONGITUDES OF THE END POINTS
% *** IN DECIMAL DEGREES ***
%
% IT USES ROBBIN'S FORMULA, AS GIVEN BY BOMFORD, GEODESY,
% FOURTH EDITION, P. 122.  CORRECT TO ONE PART IN 10**8
% AT 1600 KM.  ERRORS OF 20 M AT 5000 KM.
%
% CHECK:  SMITHSONIAN METEOROLOGICAL TABLES, PP. 483 AND 484,
% GIVES LENGTHS OF ONE DEGREE OF LATITUDE AND LONGITUDE
% AS A FUNCTION OF LATITUDE. (SO DOES THE EPHEMERIS ABOVE)
%
% PETER WORCESTER, AS TOLD TO BRUCE CORNUELLE...1983 MAY 27
%

% On 09/11/1988, Peter Worcester gave me the constants for the
% WGS84 spheroid, and he gave A (semi-major axis), F = (A-B)/A
% (flattening) (where B is the semi-minor axis), and E is the
% eccentricity, E = ( (A**2 - B**2)**.5 )/ A
% the numbers from peter are: A=6378137.; 1/F = 298.257223563
% E = 0.081819191
A = 6378137.;
E = 0.081819191;
B = sqrt(A.^2 - (A*E).^2);
EPS = E*E/(1.-E*E);

NN = max(size(lat));
if (NN ~= max(size(long))),
   error('dist: Lat, Long vectors of different sizes!');
end

if (NN == size(lat))
   rowvec = 0;  % it is easier if things are column vectors,
else
   rowvec = 1; % but we have to fix things before returning!
end;

% convert to radians
lat = lat(:)*pi/180;
long = long(:)*pi/180;

% fixes some nasty 0/0 cases in the geodesics stuff
lat(lat == 0) = eps*ones(sum(lat == 0), 1);

% endpoints of each segment
PHI1 = lat(1:NN-1);
XLAM1 = long(1:NN-1);
PHI2 = lat(2:NN);
XLAM2 = long(2:NN);

% wiggle lines of constant lat to prevent numerical problems.
if (any(PHI1 == PHI2))
   for ii = 1:NN-1
      if (PHI1(ii) == PHI2(ii))
         PHI2(ii) = PHI2(ii) + 1e-14;
      end
   end
end
% wiggle lines of constant long to prevent numerical problems.
if (any(XLAM1 == XLAM2))
   for ii = 1:NN-1
      if (XLAM1(ii) == XLAM2(ii))
         XLAM2(ii) = XLAM2(ii) + 1e-14;
      end
   end
end

% COMPUTE THE RADIUS OF CURVATURE IN THE PRIME VERTICAL FOR EACH POINT
xnu = A./sqrt(1.0-(E*sin(lat)).^2);
xnu1 = xnu(1:NN-1);
xnu2 = xnu(2:NN);

% COMPUTE THE AZIMUTHS.
% A12 (A21) IS THE AZIMUTH AT POINT 1 (2) OF THE NORMAL SECTION CONTAINING THE POINT 2 (1)
TPSI2 = (1.-E*E)*tan(PHI2) + E*E*xnu1.*sin(PHI1)./(xnu2.*cos(PHI2));
PSI2 = atan(TPSI2);

% SOME FORM OF ANGLE DIFFERENCE COMPUTED HERE??
```

```
DPHI2 = PHI2-PSI2;
DLAM = XLAM2-XLAM1;
CTA12 = (cos(PHI1).*TPSI2 - sin(PHI1).*cos(DLAM))./sin(DLAM);
A12 = atan((1.)./CTA12);
CTA21P = (sin(PSI2).*cos(DLAM) - cos(PSI2).*tan(PHI1))./sin(DLAM);
A21P = atan((1.)./CTA21P);

% GET THE QUADRANT RIGHT
DLAM2 = (abs(DLAM)<pi).*DLAM + (DLAM>=pi).*(-2*pi+DLAM) + (DLAM<=-pi).*(2*pi+DLAM);
A12 = A12 + (A12<-pi)*2*pi-(A12>=pi)*2*pi;
A12 = A12 + pi*sign(-A12).*(sign(A12) ~= sign(DLAM2));
A21P = A21P + (A21P<-pi)*2*pi - (A21P>=pi)*2*pi;
A21P = A21P + pi*sign(-A21P).*(sign(A21P) ~= sign(-DLAM2));
% A12*180/pi
% A21P*180/pi

SSIG = sin(DLAM).*cos(PSI2)./sin(A12);

% At this point we are OK if the angle < 90 but otherwise
% we get the wrong branch of asin!
% This fudge will correct every case on a sphere, and *almost*
% every case on an ellipsoid (wrong handling will be when
% angle is almost exactly 90 degrees)
dd2 = [cos(long).*cos(lat) sin(long).*cos(lat) sin(lat)];
dd2 = sum((diff(dd2).*diff(dd2))')';
if (any(abs(dd2-2) < 2*((B-A)/A))^2),
   disp('dist: Warning...point(s) too close to 90 degrees apart');
end
bigbrnch = dd2>2;

SIG = asin(SSIG).*(bigbrnch==0) + (pi-asin(SSIG)).*bigbrnch;
A21 = A21P - DPHI2.*sin(A21P).*tan(SIG/2.0);

% COMPUTE RANGE
G2 = EPS*(sin(PHI1)).^2;
G = sqrt(G2);
H2 = EPS*(cos(PHI1).*cos(A12)).^2;
H = sqrt(H2);
TERM1 = -SIG.*SIG.*H2.*(1.0-H2)/6.0;
TERM2 = (SIG.^3).*G.*H.*(1.0-2.0*H2)/8.0;
TERM3 = (SIG.^4).*(H2.*(4.0-7.0*H2)-3.0*G2.*(1.0-7.0*H2))/120.0;
TERM4 = -(SIG.^5).*G.*H/48.0;

range = xnu1.*SIG.*(1.0 + TERM1 + TERM2 + TERM3 + TERM4);

% CONVERT TO DECIMAL DEGREES
A12 = A12*180/pi;
A21 = A21*180/pi;
if (rowvec),
   range = range';
   A12 = A12';
   A21 = A21';
end
```

8.5.2 Test points

The following table provides results of calculation distances from the LPO distance algorithm.

Test #	Longitude point #1	Latitude point #1	Longitude point #2	Latitude point #2	Distance (m)
1	59.137	81.450	132.862	-71.971	17452769.38
2	245.057	-75.309	331.764	-77.086	2110391.35
3	185.622	87.327	183.692	-17.999	11689986.02
4	182.640	20.009	49.196	5.048	14227739.39
5	150.579	41.603	208.973	39.188	4868529.07
6	0.000	0.000	332.341	19.629	3717195.47
7	356.228	79.610	254.896	-47.763	15364005.55
8	199.871	88.917	70.224	52.035	4312751.18

9	287.193	-35.107	200.803	52.926	12831368.01
10	102.486	-83.242	312.077	75.131	18753227.55
11	69.797	88.120	207.543	18.708	8087967.56
12	93.492	-16.942	304.265	20.978	16765984.94
13	199.115	-39.885	182.679	60.574	11263499.39
14	303.234	77.720	332.681	-0.149	8830419.21
15	152.391	-4.042	179.072	-21.859	3490115.84
16	38.772	-90.000	252.147	9.952	11097348.67
17	170.518	85.414	311.396	-28.009	13474193.18
18	83.708	44.039	273.558	48.297	9728568.10
19	325.393	4.457	60.402	-18.541	10702629.73
20	201.680	15.173	142.753	-29.194	8013018.54

9 ANNEX H: Cookbook entry point

Given the large amount of information included in this cookbook and the way it changes for all the different float types, there needs to be an easy way for DACs to use this cookbook to make calculations for the trajectory file. This Annex has been created to make it easy for DACs to find out what calculations are needed based on float version. The Annex consists of tables including all float versions versus all measurement codes which are needed to fill the trajectory files. This should prevent against forgetting anything.

In the final version of the cookbook, the cells should be filled:

- By N/A (for Not Applicable) if the concerned data cannot (float capability) be produced from the given float version,
- Otherwise, the list of paragraphs in the cookbook that explain how to process.

These tables thus provide an overview of all the data expected to be in the TRAJ file for a given float version and a direct access (through the ability to jump to linked paragraphs (CTRL+Click)) to the concerned specifications.

These table are updated through 2014 for PROVOR, ARVOR, NINJA and SOLO floats.

All other tables are not updated or usable at this time.

9.1 PROVOR floats

Format Id	Code 0 Launch	Code 100 DST	Code 150 FST	Code 200 DET	Code 250 PST	Code 300 PET	Code 400 DDET
101001							
101002	2.1.1	2.2.14.1.6.1	2.2.14.1.6.2	FillValue	2.2.14.1.6.3	2.2.14.1.6.4	FillValue
101003	2.1.1	2.2.14.1.7.1	2.2.14.1.7.2	FillValue	2.2.14.1.7.3	2.2.14.1.7.4	FillValue
101004	2.1.1	2.2.14.1.7.1	2.2.14.1.7.2	FillValue	2.2.14.1.7.3	2.2.14.1.7.4	FillValue
101005	2.1.1	2.2.14.1.6.1	2.2.14.1.6.2	FillValue	2.2.14.1.6.3	2.2.14.1.6.4	FillValue
101006	2.1.1	2.2.14.1.3.1	2.2.14.1.3.2	FillValue	2.2.14.1.3.3	2.2.14.1.3.4	FillValue
101007	2.1.1	2.2.14.1.5.1	2.2.14.1.5.2	FillValue	2.2.14.1.5.3	2.2.14.1.5.4	FillValue
101008 - 101010	2.1.1	2.2.14.1.3.1	2.2.14.1.3.2	FillValue	2.2.14.1.3.3	2.2.14.1.3.4	FillValue
101011 - 101015	2.1.1	2.2.14.1.1.1	2.2.14.1.1.2	FillValue	2.2.14.1.1.3	2.2.14.1.1.4	FillValue
101016							
101017 - 101019	2.1.1	2.2.14.1.1.1	2.2.14.1.1.2	FillValue	2.2.14.1.1.3	2.2.14.1.1.4	FillValue
101020							

Format Id	Code 450 DPST	Code 500 AST	Code 550 DAST	Code 600 AET	Code 700 TST	Codes 702-704 FMT, LMT	Code 800 TET
101001							
101002	2.2.14.1.6.5	2.2.14.1.6.6	FillValue	2.2.14.1.6.7	2.2.14.1.6.8	2.2.1.1.1	FillValue
101003	2.2.14.1.7.5	2.2.14.1.7.6	FillValue	2.2.14.1.7.7	2.2.14.1.7.8	2.2.1.1.1	FillValue
101004	2.2.14.1.7.5	2.2.14.1.7.6	FillValue	2.2.14.1.7.7	2.2.14.1.7.8	2.2.1.1.1	FillValue
101005	2.2.14.1.6.5	2.2.14.1.6.6	FillValue	2.2.14.1.6.7	2.2.14.1.6.8	2.2.1.1.1	FillValue
101006	2.2.14.1.3.5	2.2.14.1.3.6	FillValue	2.2.14.1.3.7	2.2.14.1.3.8	2.2.1.1.1	FillValue
101007	2.2.14.1.5.5	2.2.14.1.5.6	FillValue	2.2.14.1.5.7	2.2.14.1.5.8	2.2.1.1.1	FillValue
101008 - 101010	2.2.14.1.3.5	2.2.14.1.3.6	FillValue	2.2.14.1.3.7	2.2.14.1.3.8	2.2.1.1.1	FillValue
101011 - 101015	2.2.14.1.1.5	2.2.14.1.1.6	FillValue	2.2.14.1.1.7	2.2.14.1.1.8	2.2.1.1.1	FillValue
101016							
101017 - 101019	2.2.14.1.1.5	2.2.14.1.1.6	FillValue	2.2.14.1.1.7	2.2.14.1.1.8	2.2.1.2.1	FillValue
101020							

Format Id	Code 189	Code 190	Code 198	Code 203	Code 290	Code 297	Code 298
101001							
101002 - 101005		2.4.3.4.1		2.4.3.5	2.4.1.2.1		
101006 - 101010		2.4.3.4.1		2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101011		2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101012		2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101013		2.4.3.4.1		2.4.3.5	2.4.1.2.1		
101014		2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101015		2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101016							
101017 - 101019	2.4.3.10	2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**
101020							

Format Id	Code 301 RPP	Code 389	Code 398	Code 503	Code 589	Code 590	Code 901 GRND
101001							
101002 - 101015	2.4.2			2.4.3.5		2.4.3.4.1	2.4.4
101016							
101017 - 101019	2.4.2	2.4.3.10	2.4.3.8	2.4.3.5	2.4.3.10	2.4.3.4.1	2.4.4
101020							

9.2 PROVOR-MT floats

Format Id	Code 0 Launch	Code 100 DST	Code 150 FST	Code 200 DET	Code 250 PST	Code 300 PET	Code 400 DDET
??????							
100001	2.1.1	2.2.14.1.1.1	2.2.14.1.1.2	FillValue	2.2.14.1.1.3	2.2.14.1.1.4	FillValue
100002	2.1.1	2.2.14.1.6.1	2.2.14.1.6.2	FillValue	2.2.14.1.6.3	2.2.14.1.6.4	FillValue
100003 - 100006	2.1.1	2.2.14.1.4.1	2.2.14.1.4.2	FillValue	2.2.14.1.4.3	2.2.14.1.4.4	FillValue
100007							
100008	2.1.1	2.2.14.1.4.1	2.2.14.1.4.2	FillValue	2.2.14.1.4.3	2.2.14.1.4.4	FillValue
100009							

Format Id	Code 450 DPST	Code 500 AST	Code 550 DAST	Code 600 AET	Code 700 TST	Codes 702-704 FMT, LMT	Code 800 TET
??????							
100001	2.2.14.1.1.5	2.2.14.1.1.6	FillValue	2.2.14.1.1.7	2.2.14.1.1.8	2.2.1.1.1	FillValue
100002	2.2.14.1.6.5	2.2.14.1.6.6	FillValue	2.2.14.1.6.7	2.2.14.1.6.8	2.2.1.1.1	FillValue
100003 - 100006	2.2.14.1.4.5	2.2.14.1.4.6	FillValue	2.2.14.1.4.7	2.2.14.1.4.8	2.2.1.1.1	FillValue
100007							
100008	2.2.14.1.4.5	2.2.14.1.4.6	FillValue	2.2.14.1.4.7	2.2.14.1.4.8	2.2.1.1.1	FillValue
100009							

Format Id	Code 198	Code 203	Code 290	Code 297	Code 298	Code 301 RPP
??????						
100001	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**	2.4.2
100002 - 100006		2.4.3.5	2.4.1.2.1			2.4.2
100007						
100008		2.4.3.5	2.4.1.2.1			2.4.2
100009			2.4.1.2.1			

Format Id	Code 390	Code 398	Code 497	Code 498	Code 503	Code 590	Code 901 GRND
??????							
100001 - 100006					2.4.3.5		2.4.4
100007							
100008					2.4.3.5		2.4.4
100009							

9.3 ARVOR floats

Format Id	Code 0 Launch	Code 100 DST	Code 150 FST	Code 200 DET	Code 250 PST	Code 300 PET	Code 400 DDET
102001							
102002 - 102004	2.1.1	2.2.14.1.2.1	2.2.14.1.2.2	FillValue	2.2.14.1.2.3	2.2.14.1.2.4	FillValue

Format Id	Code 450 DPST	Code 500 AST	Code 550 DAST	Code 600 AET	Code 700 TST	Codes 702-704 FMT, LMT	Code 800 TET
102001							
102002	2.2.14.1.2.5	2.2.14.1.2.6	FillValue	2.2.14.1.2.7	2.2.14.1.2.8	2.2.1.1.1	FillValue
102003	2.2.14.1.2.5	2.2.14.1.2.6	FillValue	2.2.14.1.2.7	2.2.14.1.2.8	2.2.1.1.1	FillValue
102004	2.2.14.1.2.5	2.2.14.1.2.6	FillValue	2.2.14.1.2.7	2.2.14.1.2.8	2.2.1.2.1	FillValue

Format Id	Code 190	Code 198	Code 203	Code 290	Code 297	Code 298	Code 301 RPP
102001							
102002 - 102004	2.4.3.4.1	2.4.3.6	2.4.3.5	2.4.1.2.1	**Erreur ! Source du renvoi introuvable.**	**Erreur ! Source du renvoi introuvable.**	2.4.2

Format Id	Code 390	Code 398	Code 497	Code 498	Code 503	Code 590	Code 901 GRND
102001							
102002					2.4.3.5	2.4.3.4.1	2.4.4
102003					2.4.3.5	2.4.3.4.1	2.4.4
102004		2.4.3.8			2.4.3.5	2.4.3.4.1	2.4.4

9.4 NINJA floats

Format Id	Code 0 Launch	Code 100 DST	Code 150 FST	Code 200 DET	Code 250 PST	Code 300 PET	Code 400 DDET
300001 - 300003	2.1.1	2.2.11.1.1.1	2.2.11.1.1.2	FillValue	2.2.11.1.1.3	2.2.11.1.1.4	2.2.11.1.1.5
300004	2.1.1	2.2.11.1.2	2.2.11.1.2	FillValue	2.2.11.1.2	2.2.11.1.2	2.2.11.1.2

Format Id	Code 450 DPST	Code 500 AST	Code 550 DAST	Code 600 AET	Code 700 TST	Code 702-704 FMT, LMT	Code 800 TET
300001 - 300003	FillValue	2.2.11.1.1.6	FillValue	2.2.11.1.1.7	2.2.11.1.1.8	2.2.1.1.1	2.2.11.1.1.9
300004	FillValue	2.2.11.1.2	FillValue	2.2.11.1.2	2.2.11.1.2	2.2.1.1.1	2.2.11.1.2

Format Id	Code 290	Code 301 RPP	Code 498	Code 503	Code 590	Code 901 GRND

300001 - 300003	2.4.1.3.1.2	2.4.2	2.4.3.9	2.4.3.5	2.4.3.4.2	2.4.4
300004	2.4.1.3.1.2	2.4.2		2.4.3.5		2.4.4

10 ANNEX I: APEX APF8 Estimation methods for PST, PET, AST

10.1 Park Start Time (PST)

The mean descent rate to use depends on the PARKING depth, the recommended values are provided in the following table (see also §5.1).

PARKING depth	250 dbar	500 dbar	1000 dbar	1500 dbar	2000 dbar
Mean descent rate (cm/s)	2.6	3.6	5.9	12.4	9.0

Table 4: Recommended descent rates

Thus for cycle #i:

PST(i) = DST(i) + (PARKING_PRESSURE(i) * 100 * 36)/(mean descent rate * 864)

where PARKING_PRESSURE(i) is the theoretical PARKING_PRESSURE of cycle #i.

The PST value (MC=250) should be stored in the N_MEASUREMENT arrays only in the JULD_ADJUSTED variable since the time is estimated based on float behavior. The STATUS should be set to 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour. Float clock offset corrections can also be applied.

The JULD variables should be fill value.

The PST value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 1 (estimated using procedures that rely on typical float behavior).

If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If no estimate is made, fill value should be stored in the JULD variable in the N_MEASUREMENT array with the measurement code set to 250 and the STATUS set to 9.

N_CYCLE arrays: fill value should be stored in the JULD_PARK_START variable and the JULD_PARK_START_STATUS set to 9.

10.2 Park End Time (PET)

We must check first that, for the corresponding cycle, the theoretical PARKING and PROFILE depths differ (be careful with PnP floats, see §3.1).

If not, there is no PET, do not include it in the N_MEASUREMENT array and put fill value in the N_CYCLE array JULD_PARK_END and JULD_PARK_END_STATUS variables.

Otherwise PET = TET - UP_TIME - DPDP hours.

Where DPDP is the value of the Deep Profile Descent Period, a programmed meta-data parameter that determines the maximum amount of time given to the float for diving from PARKING to PROFILE depth. In older floats without this metadata, DPDP is often between 4 and 6 hours. For the newer APF9 firmware, this time period is user-specified.

If the float clock offset has been estimated during the TET determination, the CLOCK_OFFSET (N_CYCLE) variable should also be filled. Place PET in the JULD_ADJUSTED (N_MEASUREMENT) variables with MC=300 and a STATUS of 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour.

For the N_CYCLE array, PET value should be stored in the JULD_PARK_END variable and the JULD_PARK_END_STATUS set to. If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

10.3 Ascent Start Time (AST)

If the PARKING and PROFILE depths are equal for cycle #i, then:

(1) : AST(i) = TET(i) - UP TIME

If not, we can however **roughly** estimate AST using AET and the profile duration.

(2) : AST(i) = AET(i) - duration of profile #i

The duration of profile #i can be estimated with the profile deepest pressure (ProfMaxPres(i)) and a mean ascent rate.

ProfMaxPres(i)) is the maximum pressure of the profile **if the Argos message of the first profile measurement has been received (otherwise, AST(i) should not be estimated).**

The mean Ascent rate to use can be **9.5 cm/s** (see §5.2).

Thus AST(i) = AET(i) - (PARKING_ ProfMaxPres (i) * 100 * 36)/(9.5 * 864)

We can also verify that AST(i) is in the interval

[TET(i) - UP TIME - DPDP hours; TET(i) - UP TIME].

Note that AST estimated in (1) is much more reliable than AST estimated in (2), associated JULD_QC should reflect it.

The AST value should also be stored in the JULD_ADJUSTED variables with an MC = 500 and STATUS set to 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour. Apply clock offset if it has been determined.

For the N_CYCLE array, the AST value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 1. If float clock offset has been estimated and applied, make sure to fill in the CLOCK_OFFSET (N_CYCLE) variable so users know it has been applied.

If the AST value is not estimated, fill value should be stored in the JULD variable with an MC=500 and STATUS set to 9.

N_CYCLE arrays: fill value should be stored in the JULD_ASCENT_START variable and the JULD_ASCENT_START_STATUS set to 9.

Ascent Start Time provided by the float

Some float versions (see ANNEX H: Cookbook entry point) directly provide the time at the end of DOWN TIME period ($DTET_{FL}$).

These float versions also provide, in the Auxiliary Engineering Data (AED), the "Time of profile initiation". This information is defined as the time difference, in minutes, between profile start and end of DOWN TIME (negative for start before expiration and positive for start after expiration, thus in this latter case, necessarily when TOD feature has been set).

The AED are not always transmitted (depending on the remaining space in the last Argos message) but if received, this "Time of profile initiation" (TPI) can be used to compute a second value of AST provided by the float (AST_{FL}).

$AST_{FL} = DTET_{FL}$ + TPI minutes

AST_{FL} value computed from $DTET_{FL}$ (corrected from clock offset) does not need to be corrected from clock offset but the information should be set in the AST_{FL} storage.

AST_{FL} is stored in the JULD_ADJUSTED N_MEASUREMENT arrays with the MC = 502 and the STATUS equal to 3: value is computed from information transmitted by the float. Clock offset has been applied in the $DTET_{FL}$ variable.

Argo program measurement codes (MC) for APEX APF8 floats in REAL TIME				
Code (timing)	**APF8 Variable**	**Description**	**Units**	**JULD_STATUS**
0	Float does not know when it is launched. If the launch time and location are available from the ship, enter that time and location If the launch time and location are not available, use fill value.	Launch time and location	Time, position	0: value is estimated from pre-deployment information found in the metafile Or 9: value is not immediately known, but believe it can be estimated later
100 (DST)	TET from previous cycle OR Fill Value	If TET is estimated in real time, use the TET from previous cycle. OR If TET is not estimated in real time, use FillValue	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
200 (DET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
250 (PST)	Not available, so use Fill Value OR Park Start Time estimated in 11.1			9: value is not immediately known, but believe it can be estimated later OR 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour
During the drift phase, the APF8 makes drift measurements. Common codes are listed below. See 3.4.1.1 for CTD measurements during drift for APEX floats				
296	Average pressure Average temperature	Any averaged measurements made during drift	Pressure Temp	2: value is transmitted by the float
297	Minimum pressure Minimum temperature	Minimum value taken during drift	Pressure Temp	2: value is transmitted by the float
298	Maximum pressure	Maximum value taken	Pressure	2: value is transmitted by the

	Maximum temperature	during drift	Temp	float
End of drift measurements				
300 (PET)	Not available, so use Fill Value OR Park End Time estimated using 11.1 CTD performed at end of drift		Time Time P, T, S	9: value is not immediately known, but believe it can be estimated later OR 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour
301	Average pressure during drift	Best estimate of drift depth. See section 3.4.3 for more details	Pressure	3: value is directly computed from relevant, transmitted float information
400 (DDET)	Not available, so use Fill Value			9: value is not immediately known, but believe it can be estimated later
500 (AST)	If PARK and PROFILE depths are equal and TET is estimated in real time: AST(i)=TET(i) – UP TIME OR Ascent Start Time estimated in 11.1 OR FillValue	See 3.2.2.1.7	Time	1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 1: value is estimated using information not transmitted by the float or by procedures that rely on typical float behaviour OR 9: value is not immediately known, but believe it can be estimated later
501	DownTimeEpoch/UNIX epoch when the down-time expired	Down-time end time – time out	Time	2: value is transmitted by the float
600 (AET)	Float does not know when it reaches the surface, so Fill Value		Time	9: value is not immediately known, but believe it can be estimated later
700 (TST)	See section 3.2.2.1.9 & 6.2	Based on Argos messages	Time	3: value is directly computed from relevant, transmitted float information
701 TST sent by APEX floats	TST_{FL} = $DTET_{FL}$ + TOTPI minutes	See 3.2.2.1.9.2	Time	3: value is directly computed from relevant, transmitted float information
702 (FMT)	Earliest time of all Argos messages received	Time	Time	4: value is determined by satellite
703 (ST)	All Argos times and locations		Time, Position	4: value is determined by satellite
704 (LMT)	Latest time of all Argos messages received		Time	4: value is determined by satellite
800 (TET)	3.2.2.1.1 and Annex B (5.3) OR FillValue	DACs can choose to make this estimate in real time or not. Annex B explains how to make the estimate. 3.2.2.1.1 gives guidance how to implement the	Time	3: value is directly computed from relevant, transmitted float information OR 9: value is not immediately known, but believe it can be estimated later

		method in Annex B		