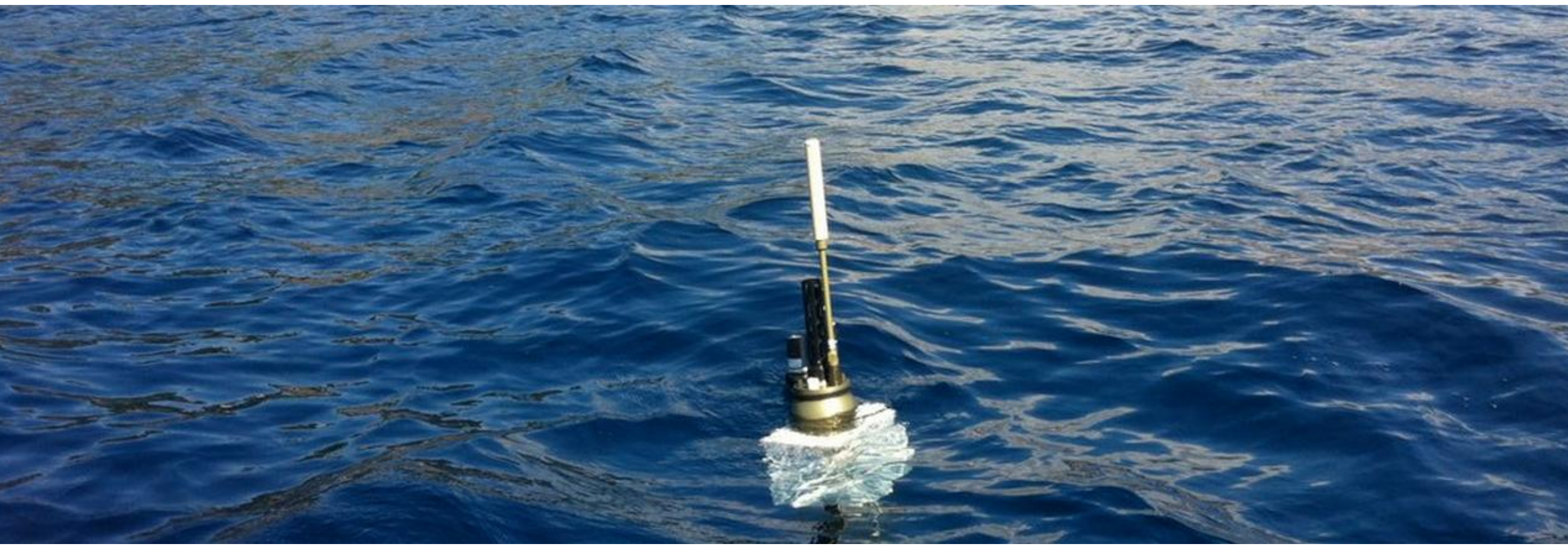
APMT Profiler - File management

AUTOMATED **M**ULTI-**T**ASK **P**ROFILER



Revision 1.1 (2018-02-15)

Table of contents

1. Revision history

Revision	Release date	Notes	Author
1.0	2017-09-04	Original	C. SCHAEFFER
1.1	2018-02-15	Upgrade of technical information: - Adding internal temperature information in [SYSTEM] - Adding pressure activation information in [ACTIVATION] - Adding nke ID information in [USER] - Changing grounding information during navigation phases in [PROFILE] and [ALARM] - Changing feedback information in [ALARM]	C. SCHAEFFER

2. File system

Float management is based on a specific file system developed by nke. This file system allows information and data to be stored when the float is turned off, together with the mission configuration.

These files can be accessed by an FTP client via the Bluetooth connection.

2.1 Memory organisation

The memory space is divided into different areas intended for different uses. In particular, the following spaces can be found:

- Reserved: restricted system use to store file manipulation and management information
- Working: used for temporary storage (awaiting transmission).
- Storage (optional): used to expand the temporary storage capacity of the workspace

2.2 File types

The system uses various file types, as needed:

- Configuration file for mission and sensor settings
- Sensor data files (raw, processed and transmitted data)
- Technical data files
- Execution trace files (mission progress monitoring)
- Specific files (restricted system use)

2.3 Main memory – Working memory

The working memory (fast access) is itself divided into two physical spaces:

- Area for repeated access files: used for files with recurring access (configuration file...)
- Area for temporary files: used mainly for the generation of sensor information and data during the various navigation phases (a process that automatically deletes the oldest files in favour of new ones allows a minimum free memory space to be maintained)

2.4 Extended memory – Storage memory (optional)

When the application requires it, the use of an additional memory (slow access) allows the storage capacity of the system to be increased.

In this case, the files (awaiting transmission) deleted from the working memory are moved to the extended memory. They will then be repatriated to the working memory (for transmission) when the main memory space is sufficient.

3. Configuration files

The mission configuration is saved as a text file organised by sections and tags.

3.1 Local configuration via Bluetooth

During the mission configuration phase performed locally via Bluetooth, this file can be written and/or modified by the user. It can be manually uploaded in full using an FTP client or partially modified with a TELNET command interpreter.

3.2 Remote configuration via remote commands

When the mission is in progress, the file can be modified by sending remote commands via the satellite communication system (RUDICS...).

3.3 Transmission of the configuration file

The system transmits the current configuration file at various stages of the mission:

- At the start of the mission (initial configuration)
- Upon a change of configuration following the application of a remote command or pre-programmed change (script)

The various files generated are named using the format "xxxx_ccc_mm_apmt.ini", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)

4. Technical files

The various pieces of technical information associated with a profile are saved in a text file organised by sections.

4.1 Transmission of technical information

The system transmits technical files at various stages of the mission:

- During system verification prior to the mission (self-test)
- During the mission at the end of each pattern
- At end of life

4.1.1. Self-test files

The files generated are named using the format "xxxx_ccc_autotest_nnnnn.txt", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "nnnnn": identification of self-test file number associated with the cycle (5 characters)

4.1.2. Mission files

The files generated are named using the format "xxxx_ccc_mm_technical.txt":

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)

<u>Note</u>: A pattern numbered "00" corresponds to a pre-mission file.

4.1.3. End-of-life files

The various files generated are named using the format "xxxx_ccc_mm _default_nnnnn.txt", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)
- "nnnnn": identification of end-of-life file number associated with the cycle/pattern pair (5 characters)

4.2 Technical information available

The data is organised in sections. The various sections and fields that compose them are optional and may not be present in the files.

[SYSTEM]

Information	Format
External pressure offset	Pe offset=xx.xx dbar
Internal pressure	Pi=xxx.x mbar
No-load battery voltage	Vbatt=xx.x V
Battery voltage under load (min.)	Vbatt peak min=xx.x V
External temperature (air)	Te air=xx.x°C
External pressure (air)	Pe air=xx.xx dbar
Internal temperature	Ti=xx.x°C

[GPS]

information	Format
Time stamping, position and clock drift	UTC=yy-mm-dd hh:mm:ss Lat=xxxx.xxxxxy Long=xxxxx.xxxxxy Clock drift=xx.xxx s

(*) Drift = Float time - GPS time. The offset is reset to 0 after every GPS retiming.

[USER]

Information	Format
APMT version	APMT=x.xx.xxx
Payload version	Payload=xxxxxxxxxxxxxxx
SIM card ID	CID=xxxxxxxxxxxxxxx
nke ID	WC=xxxxxxxxxx

[ACTIVATION]

Information	Format
Pressure activation start	UTC=yy-mm-dd hh:mm:ss Detection start
Pressure activation stop	UTC=yy-mm-dd hh:mm:ss Detection stop

[PROFILE]

Information	Format
Emergence reduction: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Flotation=xxx.x cm3 (xx)
First stabilization - Date/time - Depth	UTC=yy-mm-dd hh:mm:ss First stabilization=xxx dbar
Park descent: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Descent=xxx.x cm3 (xx)
Grounding park descent: - Start date/time	UTC=yy-mm-dd hh:mm:ss Grounding Descent=xxx dbar

- Grounding pressure	
Grounding park descent: - End date/time - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Descent escape=xxx.x cm3
Park drift: - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Park=xxx/xxx dbar (xx/xx) stability=x/x
Stabilized park drift: - Stabilization date/time - Stabilization depth	UTC=yy-mm-dd hh:mm:ss Park stabilization=xxx dbar
Grounding park drift: - Start date/time - Grounding number (1-5) - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Park x=xxx dbar
Grounding park drift: - End date/time - Grounding number (1-5) - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Park x escape=xxx.x cm3
Measurement descent: - Start date/time - Oil volume transferred - Nb. solenoid valve actions	UTC=yy-mm-dd hh:mm:ss Deep profile=xxx.x cm3 (xx)
Grounding measurement descent: - Start date/time - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Deep profile=xxx dbar
Grounding measurement descent: - End date/time - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Deep profile escape=xxx.x cm3
Measurement drift: - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Short Park=xxx/xxx dbar (xx/xx) stability=x/x
Grounding short park drift: - Start date/time - Grounding number (1-5) - Grounding pressure	UTC=yy-mm-dd hh:mm:ss Grounding Short park x=xxx dbar
Grounding short park drift: - End date/time - Grounding number (1-5) - Oil volume transferred	UTC=yy-mm-dd hh:mm:ss Grounding Short park x escape=xxx.x cm3
Ascent (standard): - Start date/time	UTC=yy-mm-dd hh:mm:ss Ascent=xxx.x cm3 (xx/xx) from xxx dbar

- Oil volume transferred - Nb. pump actions (total) - Nb. actions for take-off - Maximum depth	
Ascent (slow): - Start date/time - Oil volume transferred - Nb. pump actions	UTC=yy-mm-dd hh:mm:ss Ascent (slowly)=xxx.x cm3 (xx)
Ascent (resume): - Start date/time - Oil volume transferred - Nb. pump actions	UTC=yy-mm-dd hh:mm:ss Ascent (resume)=xxx.x cm3 (xx)
Ascent (end) - End date/time	UTC=yy-mm-dd hh:mm:ss Ascent end
Surface - Start date/time	UTC=yy-mm-dd hh:mm:ss Surface
Hanging: - Start date/time - Snagging pressure	UTC=yy-mm-dd hh:mm:ss Hanging=xxx dbar
Hanging: - End date/time	UTC=yy-mm-dd hh:mm:ss Hanging escape
Park drift (ice): - Start date/time - Min. depth - Max. depth - Nb. solenoid valve actions - Nb. pump actions - Nb. set point inputs - Nb. set point outputs	UTC=yy-mm-dd hh:mm:ss Ice Park=xxx/xxx dbar (xx/xx) stability=x/x
Stabilized park drift (ice): - Stabilization date/time - Stabilization depth	UTC=yy-mm-dd hh:mm:ss Ice Park stabilization=xxx dbar
Pressure switch activation	UTC=yy-mm-dd hh:mm:ss Pressure switch activation
Emergency ascent - Start date/time	UTC=yy-mm-dd hh:mm:ss Emergency ascent

[DATA]

information	Format
Data transmission (*): - Total data size - Nb. files - Average baud rate - Nb. sessions	Upload=xx.x kB of x file(s) at x.x kB/min in x session(s)
Remote command reception: - Nb. accepted - Nb. refused - Nb. unknown	Download=command file (x accepted, x refused, x unknown)
Payload configuration download	Download=payload file
Script download	Download=script file
Number of files associated with	Pattern=x files

the pattern (**)	
Number of points per sensor: - Sensor name (SBE41…) - Nb. park descent - Nb. park drift - Nb. measurement descent - Nb. measurement drift - Nb. ascent - Nb. subsurface	xxxxx=x/x/x/x/x/x points

(*) Information on previous pattern/cycle.
(**) The following files are not taken into account: technical for self-test or pre-mission, configuration nor command/script acknowledgement. The number of files is expressed prior to the files being broken down for transmission.

[POWER]

information	Format
Pattern duration	Pattern=x min
Processing/standby ratio	Processing=xx %
Cumulated hydraulic activations: - Solenoid valve - Pump	SV/Pump=xxx/xxx cs
Cumulated SBE41 activations	SBE41=xxx min
Cumulated modem activations (*)	Transmission=xxx min
Cumulated GPS activations	GPS=xxx s

(*) Information on previous pattern/cycle.

[ALARM]

information	Format
Deployment	
Start up	Power-on
Invalid configuration	Bad configuration
Excessively heavy float	Flotation (heavy)
Excessively light float	Flotation (light)
Self-test failure: - Source(s) xxx among "FRAM, FLASH, RTC, Vbatt, Pi, Pe, SBE41-Cutoff, SBE41-Offset, GPS, Payload, Sensor(xxx), Transmitter"	Autotest fail=xxx, …
State	
No-load battery voltage low	Vbatt low
Battery voltage under load low	Vbatt peak low
Low external pressure	Pe low (xxx dbar)
High external pressure	Pe high (xxx dbar)
External pressure fault	Pe default
External breaking pressure	Pe broken

Gear skip	Pe SR high
High internal pressure	Pi high
Presence of water	Water inside
Navigation	
Grounding during park descent	Grounding Descent (xxx dbar)
Grounding during park drift	Grounding Park x (xxx dbar)
Grounding during measurement descent	Grounding Deep profile (xxx dbar)
Grounding during short park drift	Grounding Short park x (xxx dbar)
Snagging during ascent	Hanging (xxx dbar)
Braking during descent	Braking
Operating errors	
System fault	System
Payload board fault	Payload
GPS fault	GPS
Hydraulic fault	Hydraulic
ADC fault	ADC
File fault	File (skip)
RTC fault	RTC
Pressure switch fault	Pressure switch
Mode switch	
End of life: - Source xxx among "Pi high, Pe broken, Pe high, Vbatt low, Vbatt peak low, Water inside, Flotation (heavy), Flotation (light)"	End of life (xxx)
Rescue procedure	Rescue
Feedback: - Type xxx among "Early profile, Early surface, Abort profile, Abort cycle 0, Abort cycle 1, Go to deep, Standard speed, Lower speed" - Nb. accepted - Nb. refused	Feedback=xxx (x accepted, x refused) Feedback=xxx (x accepted, x refused) …

Example:

```
[SYSTEM]
Pi=750.3 mbar
Vbatt=10.7 V
Vbatt peak min=10.2 V
[PROFILE]
UTC=15-09-03 16:32:26 Flotation=6.6 cm3 (2) down to 17 dbar
UTC=15-09-03 16:33:09 Descent=7.5 cm3 (5)
UTC=15-09-03 17:28:26 Park=51/51 dbar (0/0) stability=0/0
UTC=15-09-03 17:32:29 Deep profile=0.0 cm3 (0)
UTC=15-09-03 17:35:11 Ascent=15.2 cm3 (3/2) from 54 dbar
UTC=15-09-03 17:47:29 Ascent end
[POWER]
Pattern=89 min
Treatment=2 %
EV/Pump=221/448 cs
SBE41=79 min
[ALARM]
Payload
```

5. Sensor data files

The system can generate sensor data files under three formats:

- Text format (*.csv)
- Standard binary format (*.hex)
- Extended binary format (*.hex)

5.1.Transmission of sensor data

The various sensor data files generated are named using the format "xxxx_ccc_mm_ssssss.ext", with:

- "xxxx": float's hexadecimal serial number (4 characters)
- "ccc": cycle number (3 characters)
- "mm": pattern number (2 characters)
- "sssss": sensor ID (N characters, e.g. "sbe41")
- "ext": file extension

5.2.ID tags

All files are organised in sections using ID tags for the phase and processing type.

5.2.1. Navigation phase identification

At every change of navigation phase, an ID string is inserted. The various ASCII strings are as follows:

- **[DESCENT]**: Data from the descent phase, from the surface to the drift depth
- **[PARK]**: Data from the drift phase
- **[DEEP_PROFILE]**: Data from the descent phase, from the drift depth to the measurement depth
- **[SHORT_PARK]**: Data from the drift phase, from the drift depth to the measurement depth
- **[ASCENT]**: Data from the ascent phase

5.2.2. Processing stage identification

At every change of phase or processing area, an ID string for the processing stages is inserted. The various ASCII strings are as follows:

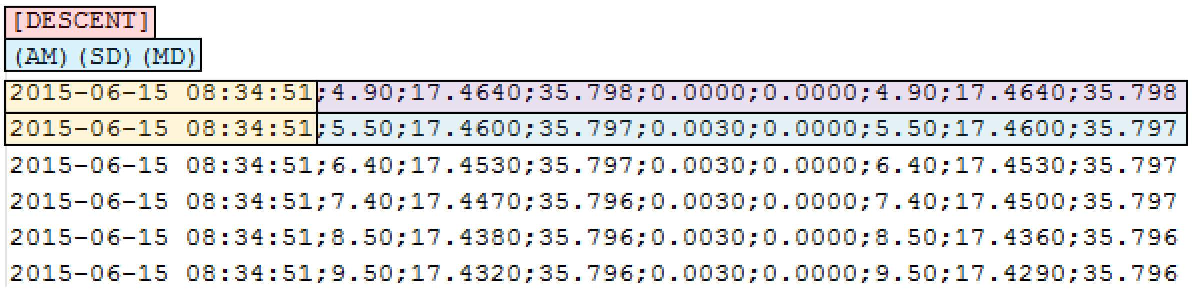
- **(RW)**: For raw data
- **(AM)**: For arithmetic mean
- **(SD)**: For standard deviation
- **(MD)**: For median
- **(SS)**: For subsurface point

Several processing stages can follow one another. The combinations are as follows:

- (RW)
- (AM)
- (AM)(SD)
- (AM)(MD)
- (AM)(SD)(MD)
- (SS)

5.3.Recording in text format

Spreadsheet format with the possibility of choosing the decimal separator, the column separator as well as the timestamping format.



```
[DESCENT]
(AM)(SD)(MD)
2015-06-15 08:34:51;4.90;17.4640;35.798;0.0000;0.0000;4.90;17.4640;35.798
2015-06-15 08:34:51;5.50;17.4600;35.797;0.0030;0.0000;5.50;17.4600;35.797
2015-06-15 08:34:51;6.40;17.4530;35.797;0.0030;0.0000;6.40;17.4530;35.797
2015-06-15 08:34:51;7.40;17.4470;35.796;0.0030;0.0000;7.40;17.4500;35.797
2015-06-15 08:34:51;8.50;17.4380;35.796;0.0030;0.0000;8.50;17.4360;35.796
2015-06-15 08:34:51;9.50;17.4320;35.796;0.0030;0.0000;9.50;17.4290;35.796
```

5.4. Recording in binary format

5.4.1. Encoding format

The data is saved in "little endian" format:

- Byte (char)
- Short integer (2 bytes)
- Long integer (4 bytes)
- Timestamping (4 bytes – long integer) - Unix Epoch format - January 1st, 1970 at 0:00
- Floating (4 bytes - float)

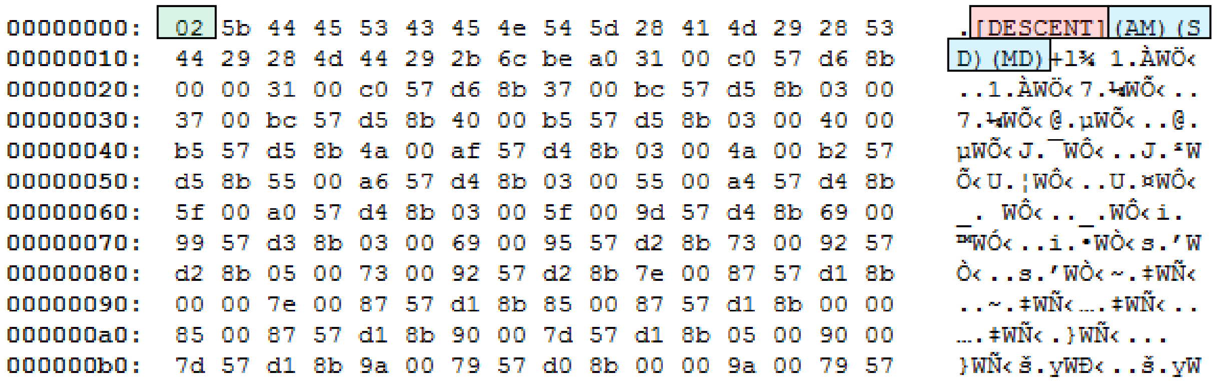
5.4.2. File structure

Each file begins with an encoding identifying byte:

- 0x01: Extended format
- 0x02: Standard format (compact)

This is followed by groups of recordings identified by the tags.

Example: Standard format file, descent phase



```
00000000: 02 5b 44 45 53 43 45 4e 54 5d 28 41 4d 29 28 53    .[DESCENT](AM)(S
00000010: 44 29 28 4d 44 29 2b 6c be a0 31 00 c0 57 d6 8b    D)(MD)+l¾ 1.ÀWÖ‹
00000020: 00 00 31 00 c0 57 d6 8b 37 00 bc 57 d5 8b 03 00    ..1.ÀWÖ‹7.¼WÕ‹..
00000030: 37 00 bc 57 d5 8b 40 00 b5 57 d5 8b 03 00 40 00    7.¼WÕ‹@.µWÕ‹..@.
00000040: b5 57 d5 8b 4a 00 af 57 d4 8b 03 00 4a 00 b2 57    µWÕ‹J.¯WÔ‹..J.²W
00000050: d5 8b 55 00 a6 57 d4 8b 03 00 55 00 a4 57 d4 8b    Õ‹U.¦WÔ‹..U.¤WÔ‹
00000060: 5f 00 a0 57 d4 8b 03 00 5f 00 9d 57 d4 8b 69 00    _. WÔ‹.._.WÔ‹i.
00000070: 99 57 d3 8b 03 00 69 00 95 57 d2 8b 73 00 92 57    ™WÓ‹..i.•WÒ‹s.'W
00000080: d2 8b 05 00 73 00 92 57 d2 8b 7e 00 87 57 d1 8b    Ò‹..s.'WÒ‹~.‡WÑ‹
00000090: 00 00 7e 00 87 57 d1 8b 85 00 87 57 d1 8b 00 00    ..~.‡WÑ‹….‡WÑ‹..
000000a0: 85 00 87 57 d1 8b 90 00 7d 57 d1 8b 05 00 90 00    ….‡WÑ‹.}WÑ‹...
000000b0: 7d 57 d1 8b 9a 00 79 57 d0 8b 00 00 9a 00 79 57    }WÑ‹š.yWÐ‹..š.yW
```

Note: In Iridium RUDICS transmission, files can be supplemented by padding bytes (0x1A). These bytes must not be decoded. Each file can end with a sequence of 0 to 1023 padding bytes.

5.4.3. Data encoding

Data are coded as follows:

- Pressure in cbar (unsigned integer)
- Temperature in m°C and +5.0°C offset (unsigned integer)
- Salinity in mpsu (unsigned integer)
- Temperature standard deviation in m°C (signed integer)
- Salinity standard deviation in mpsu (signed integer)

5.4.4. Standard binary format (compact)

This format corresponds to the version currently in place on the other nke floats:

- Timestamping of the first point of each phase and/or processing area
- Standard accuracy

During the "descent" and "ascent" navigation phases, the first point of each processing area is timestamped in absolute value, then the recordings for the next points are not timestamped.

During the "drift" phases, all points are timestamped in absolute value.

The recording structures are as follows:

Subsurface

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
tSSStd;
```

Raw data (drift)

```
struct
 {
  unsigned long int uliDateTime;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
tRStdPark;
```

Raw data (navigation)

```
struct
 {
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
tRStdNav;
```

Averaged data

```
struct
 {
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
 }
tMStd;
```

Averaged data and standard deviation

```
struct
 {
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
 }
tMECStd;
```

Averaged data and median

```
struct
 {
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
 }
tMMStd;
```

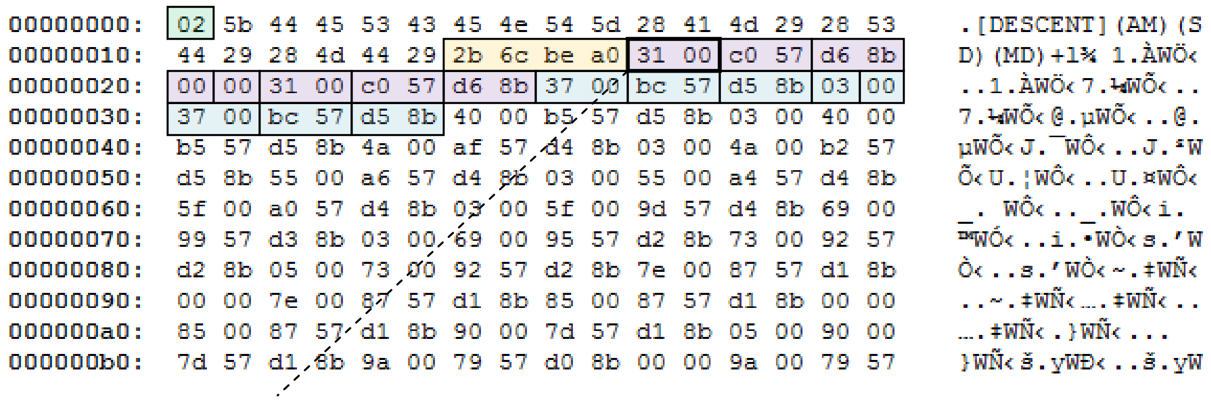
Averaged data, standard deviation and median

```
struct
 {
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
 }
```

tMECMStd;

Example:

2015-06-15 08:34:51
4.9;17.464;35.798;0.000;0.000;4.9;17.464;35.798
5.5;17.460;35.797;0.003;0.000;5.5;17.460;35.797



```
00000000:  02 5b 44 45 53 43 45 4e 54 5d 28 41 4d 29 28 53    .[DESCENT](AM)(S
00000010:  44 29 28 4d 44 29 2b 6c be a0 31 00 c0 57 d6 8b    D)(MD)+l¾ 1.ÀWÖ‹
00000020:  00 00 31 00 c0 57 d6 8b 37 00 bc 57 d5 8b 03 00    ..1.ÀWÖ‹7.¼WÕ‹..
00000030:  37 00 bc 57 d5 8b 40 00 b5 57 d5 8b 03 00 40 00    7.¼WÕ‹@.µWÕ‹..@.
00000040:  b5 57 d5 8b 4a 00 af 57 d4 8b 03 00 4a 00 b2 57    µWÕ‹J.¯WÔ‹..J.²W
00000050:  d5 8b 55 00 a6 57 d4 8b 03 00 55 00 a4 57 d4 8b    Õ‹U.¦WÔ‹..U.¤WÔ‹
00000060:  5f 00 a0 57 d4 8b 03 00 5f 00 9d 57 d4 8b 69 00    _. WÔ‹.._.WÔ‹i.
00000070:  99 57 d3 8b 03 00 69 00 95 57 d2 8b 73 00 92 57    ™WÓ‹..i.•WÒ‹s.’W
00000080:  d2 8b 05 00 73 00 92 57 d2 8b 7e 00 87 57 d1 8b    Ò‹..s.’WÒ‹~.‡WÑ‹
00000090:  00 00 7e 00 87 57 d1 8b 85 00 87 57 d1 8b 00 00    ..~.‡WÑ‹….‡WÑ‹..
000000a0:  85 00 87 57 d1 8b 90 00 7d 57 d1 8b 05 00 90 00    ….‡WÑ‹.}WÑ‹...
000000b0:  7d 57 d1 8b 9a 00 79 57 d0 8b 00 00 9a 00 79 57    }WÑ‹š.yWÐ‹..š.yW
```

Pressure = 49 cbar (0x0031)

5.4.5. Extended binary format

Advantages of this format compared to the compact version:

- Timestamping of each point
- Improved resolution of recordings (pressure and temperature)

During the "descent" and "ascent" navigation phases, the first point of each processing area is timestamped in absolute value (reference date), then the recordings for the next points are timestamped in relative value by encoding the date in relation to the reference date (deviation in seconds).

During the "drift" phases, all points are timestamped in absolute value.

The resolution extension for pressure and temperature measurements is encoded on a shared byte using masks, as follows:

- Pressure extension (0.01 dbar) = value & 0xF0
- Temperature extension (0.1 m°C) = value & 0x0F

The recording structures are as follows:

Subsurface

```
struct
  {
   unsigned long int uliDateTime;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
  }
 tSSExt;
```

Raw data (park)

```
struct
  {
   unsigned long int uliDateTime;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
  }
 tRExtPark;
```

Raw data (navigation)

```
 struct
  {
   unsigned short int uiDateTimeDelta;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
  }
 tRExtNav;
```

Averaged data

```
struct
  {
   unsigned short int uiDateTimeDelta;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
  }
 tMExt;
```

Averaged data and standard deviation

```
struct
  {
   unsigned short int uiDateTimeDelta;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
   signed char cStdDeviationTemperature;
   signed char cStdDeviationSalinity;
  }
 tMECExt;
```

Averaged data and median

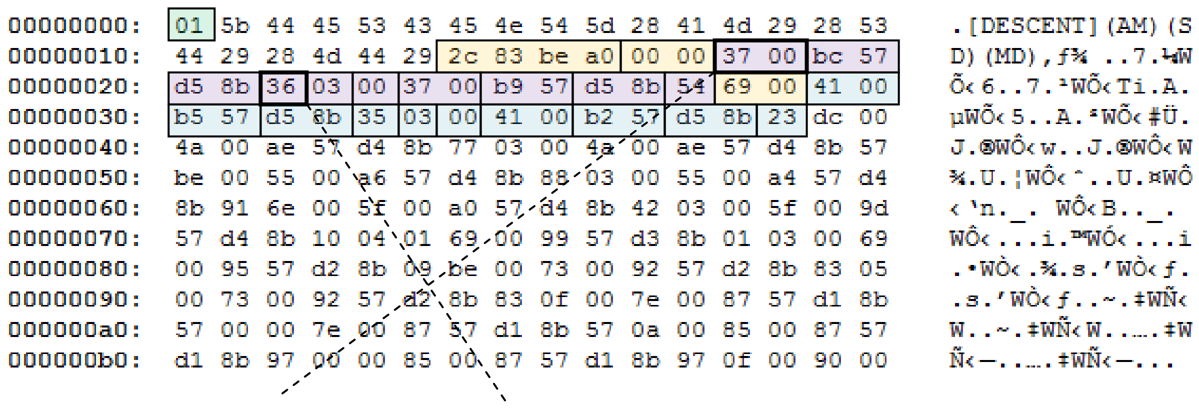
```
struct
  {
   unsigned short int uiDateTimeDelta;
   unsigned short int uiAveragePressure;
   unsigned short int uiAverageTemperature;
   unsigned short int uiAverageSalinity;
   unsigned char ucAveragePTExtra;
   unsigned short int uiMedianPressure;
   unsigned short int uiMedianTemperature;
   unsigned short int uiMedianSalinity;
   unsigned char ucMedianPTExtra;
  }
 tMMExt;
```

Averaged data, standard deviation and median

```
struct
 {
  unsigned short int uiDateTimeDelta;
  unsigned short int uiAveragePressure;
  unsigned short int uiAverageTemperature;
  unsigned short int uiAverageSalinity;
  unsigned char ucAveragePTExtra;
  signed char cStdDeviationTemperature;
  signed char cStdDeviationSalinity;
  unsigned short int uiMedianPressure;
  unsigned short int uiMedianTemperature;
  unsigned short int uiMedianSalinity;
  unsigned char ucMedianPTExtra;
 }
tMECMExt;
```

<u>Example</u>: Recording

2015-06-15 10:13:00;5.53;17.4606;35.797;0.003;0.000;5.55;17.4574;35.797
2015-06-15 10:14:45;6.53;17.4535;35.797;0.003;0.000;6.52;17.4503;35.797



```
00000000:  01 5b 44 45 53 43 45 4e 54 5d 28 41 4d 29 28 53   .[DESCENT](AM)(S
00000010:  44 29 28 4d 44 29 2c 83 be a0 00 00 37 00 bc 57   D)(MD),ƒ¾ ..7.¼W
00000020:  d5 8b 36 03 00 37 00 b9 57 d5 8b 54 69 00 41 00   Õ‹6..7.¹WÕ‹Ti.A.
00000030:  b5 57 d5 8b 35 03 00 41 00 b2 57 d5 8b 23 dc 00   µWÕ‹5..A.²WÕ‹#Ü.
00000040:  4a 00 ae 57 d4 8b 77 03 00 4a 00 ae 57 d4 8b 57   J.®WÔ‹w..J.®WÔ‹W
00000050:  be 00 55 00 a6 57 d4 8b 88 03 00 55 00 a4 57 d4   ¾.U.¦WÔ‹ˆ..U.¤WÔ
00000060:  8b 91 6e 00 5f 00 a0 57 d4 8b 42 03 00 5f 00 9d   ‹'n._. WÔ‹B.._.
00000070:  57 d4 8b 10 04 01 69 00 99 57 d3 8b 01 03 00 69   WÓ‹...i.™WÓ‹...i
00000080:  00 95 57 d2 8b 09 be 00 73 00 92 57 d2 8b 83 05   .•WÒ‹.¾.s.'WÒ‹ƒ.
00000090:  00 73 00 92 57 d2 8b 83 0f 00 7e 00 87 57 d1 8b   .s.'WÒ‹ƒ..~.‡WÑ‹
000000a0:  57 00 00 7e 00 87 57 d1 8b 57 0a 00 85 00 87 57   W..~.‡WÑ‹W.….‡W
000000b0:  d1 8b 97 00 00 85 00 87 57 d1 8b 97 0f 00 90 00   Ñ‹—..….‡WÑ‹—...
```

Pressure = 55 cbar (0x0037) + 0.03 dbar (0x3x) = 5.53 dbar

nke Instrumentation
Rue Gutenberg, ZI de Kerandré
56700 Hennebont, France
Tel (+33) 2 97 36 10 12 – Fax (+33) 2 97 55 17

www.nke-instrumentation.com