Integrator's Guide

N3015-007

Table of Contents

1 Introduction

The primary objective of this manual is to provide the information needed to control a Nortek product that is based on the AD2CP hardware platform. This includes all instruments in the Signature series. It is aimed at system integrators and engineers with interfacing experience, but it also includes examples on how to configure and start the instrument for more unexperienced integrators. The document's scope is limited to interfacing and does not address general performance issues of the instrument. For a more thorough understanding of the principles, we recommend the Principles of Operation and for information about how to operate the instrument, we recommend the Operation Manual.
The document is complete in the sense that it describes all available commands and modes of communication. For most users, it will make sense to let the supplied Nortek software do most of the hardware configuration and then let the controller limit its task to starting/stopping data collection.

As always, these types of documents are subject to change. We recommend that you check http://www.nortek-as.com/en/support or contact Nortek to ensure you have the all the latest information and versions of any software you plan to use.

If you have any comments or suggestions on the information given here, please let us know. Your comments are always appreciated; our general e-mail address is inquiry@nortek.no.

Revision

Version 1 - Initial Document	
Version 2	20.10.2015
Version 3	01.03.2016
Version 4	14.03.2016
Version 5 - New commands	30.09.2016

2 Basic interface concept

The Nortek Signature Series products command interface are ASCII based and line oriented. Before diving into the chapters covering interfaces and commands, the operational modes and how to change between the modes are described. Understanding the use and constrains of the modes is important as it is used frequently when communicating with the instrument.

2.1 Modes

The current profiler operates in distinct modes. These modes will have several explicit commands in order control the instrument. The majority of the commands are initiated from the Command mode. The possible modes for the instrument are:

- Command = Command and control
- Data Retrieval = Data download from recorder
- Measurement = Data collection mode
- Confirmation = Confirmation mode

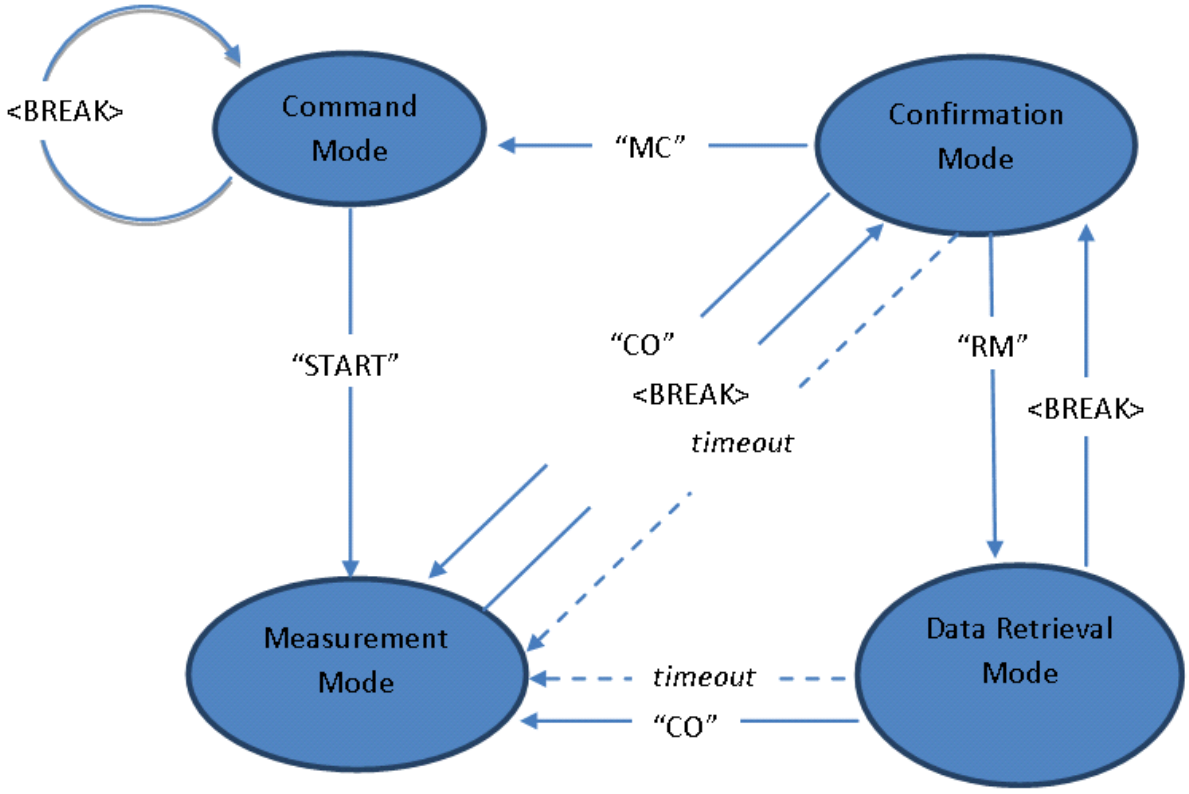

Figure: Instrument modes of operation

Initializing communication with the instrument is performed by sending a < BREAK >, which is defined below. The <BREAK> will either set the instrument in Confirmation mode or restart Command mode. The options for changing mode depends on the present mode of the instrument (see diagram above for clarity). The timeout shown in the diagram occurs if no commands are received in the various modes. A timer will then ensure that instrument operation continues. The

timeout value in Confirmation and Data Retrieval modes is 60 seconds. There is also a timeout in Command Mode when operating over the serial interface. If no commands are received for 5 minutes, a break or a sequence of @@@@@@ must be sent to wake up the processor.

2.2 Break

<BREAK> over the serial RS232/RS422 interface is defined as:

@@@@@@ <delay 100 milliseconds> **K1W%!Q** <delay 300 milliseconds> **K1W%!Q**

The @@@@@@ are used to wake up the processor when it is in sleep mode since the instrument will only be able to monitor activity on the serial line when it sleeps. The second sequence of the actual break characters is there to ensure that a break is detected even when the instrument is waking up due to some other cause (e.g. alarm from the real time clock). This ensures that the processor will interpret the following command correctly.

The figure and the table below show the specified timing of the BREAK sequence:

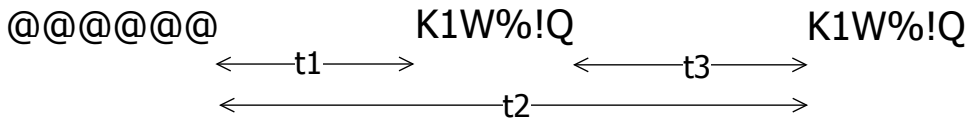

Figure: Break timing

Symbol	Parameter	Min.	Typical	Max.	Unit
t1	Time from end of @-sequence to start of ***first*** K1W%!Q-sequence.	100	150		ms
t2	Time from end of @-sequence to start of ***second*** K1W%!Q-sequence.	500	1000	2000	ms
t3	Time between ***first*** and ***second*** K1W%!Q-sequence.	300	400		ms

3 Interfaces

In addition to the traditional serial port interface for real time data output there are several options for communication over Ethernet. The Ethernet communication is handled by a dedicated processor in the instrument. This network processor runs a Linux operating system, which makes it possible to connect to the instrument via telnet, raw connections and FTP. The network processor mainly provides Ethernet connectivity. The other processor in the instrument, called the Doppler processor, is where the commands end up and where they are used to perform the measurements as specified.

The concept of a telemetry file has also been introduced which can be utilized in several ways depending on the chosen interface. Below are some details:

3.1 Command interface

The command interface makes it possible to communicate with a Signature instrument using terminal software, the serial port and a set of commands. The interface is also available over Telnet. Some highlights:

- ASCII based and line oriented. Commands are terminated with CR/LF
- Optional capsulation of commands using NMEA style prefix and checksum to ensure data integrity
- NMEA style commands will return argument names in their response
- Argument limits can be retrieved through commands
- Comprehensive validation and error handling is implemented.
- Invalid configurations return the erroneous argument with limits directly, so that each subsequent error can be handled until a valid configuration is achieved
- A single command can be used to retrieve the complete configuration of the instrument with optional output to file
- Commands to set default parameters
- External controllers can use commands to store data in the raw data file (e.g. GPS position)

3.2 Telemetry

Our use of the telemetry term implies a "subset transfer system", that is, storing a subset of data for transfer over low-bandwidth links (for example over iridium links, acoustic modems, etc). For online data transmission a versatile scheme for telemetry options is available. The telemetry file can be read out over the serial interface either in chunks or as a complete file while checksum or CRC on the downloaded data can be applied in a configurable manner. This enables external controllers to configure separate handling of all, or a subset, of the measured data. Since the instruments store individual ping data to file, the telemetry option can also be used to average velocity data within the instrument.

The file can be output directly as they are ready, or the data can be stored to a telemetry file for later retrieval. The data format can be selected from a number of formats, including both binary and ASCII data formats.

Since the telemetry file can be retrieved also in data retrieval mode, the instrument will continue measuring after a timeout delay if the data transfer was interrupted. Erasing the telemetry file after data retrieval will ensure that no data is lost if the transfer is interrupted.

To FTP

The telemetry option implemented in the Doppler processor enables system integrators to regularly offload subsets of the data by using FTP. When the network processor receives an incoming FTP request, it will interrupt the Doppler processor by entering data retrieval mode and mount the file system of the recorder. The data files on the recorder can then be accessed over FTP. The telemetry file can be deleted after it has been downloaded, which is particularly suitable for event driven data downloads. If the instrument was started with the DEPLOY command, it will resynchronize to its measurement time base after the FTP session has ended.

For an example on how to configure the instrument to output a telemetry file and download the file to FTP, check out this section.

3.3 Ethernet Operation

The AD2CP uses TCP (transmission control protocol) for both command processing and data transmission. The Internet Protocol uses a combination of the IP address and port to uniquely identify a communications channel between two computers. For the AD2CP, different ports represent different means of communicating with the instrument. TCP ports 9001, 2002, 9004 are assigned for the following uses:

- Port 9000 is a telnet-protocol ASCII interface (require username / password authentication)
- port 9001 is a raw (binary) interface (requires username / password authentication)
- port 9002 is a data only channel (no input accepted)
- port 9004 is an ASCII data only channel (no input accepted).

The password entry is ignored if password authentication, as shown in the web page configuration, is disabled (so any input, including an empty password, is accepted). The command and data record formats for the interfaces are the same as for the serial port.

Commands available in measurement mode should be preceded by the command BBPWAKEUP. This ensures that the BBP is ready to process the command when it is received (see Checking instrument state over Ethernet). In measurement mode, another BBPWAKEUP must be sent when more than 2 seconds has elapsed since the previous command.

If uncertain of the active mode it is good practice to send BBPWAKEUP before sending GETSTATE or INQ.

3.3.1 Telnet Connection

The telnet interface (TCP/IP port 9000) is used for user interaction with the instrument. This dedicated port can be used for entering commands and getting human readable responses (ASCII). The supported command set is available in the Commands section. The Windows telnet client can be used to connect into the instrument using the command line telnet ip_address 9000. You will get prompted for a username (nortek) and password (hit Enter if password protection hasn't been enabled via the Web interface).

```
Signature Username: nortek
Password:
Nortek Signature Command Interface
```

The interface is very similar to the direct serial interface over RS232/RS422 but some additions are made to simplify the interfacing. Most notable is the ability to send a <break> to the Doppler processor just by using **Ctrl-C** (ASCII 0x03). The internal application takes care of waking up the Doppler DSP and timing the delivery of the break string.

The telnet server is not configured to echo characters, so users wishing to see and/or edit commands before sending them to the instrument should enable local echo and local line editing. If those features are desired, a telnet client capable of supporting local echo and local line editing must be used (e.g. PuTTY).

Port 9000 is dedicated for ASCII only communication whereas the ports described in the next section provide the complete set of data, including binary output of the measurements. A telnet client should *not* be used to access these ports. Read more about this in the next section.

To terminate the telnet connection, enter **Ctrl-X** (ASCII 0x18).

3.3.2 Raw Connections

A port can be understood as a address point between two communicating parts. When first connecting to a data listening port, the string "\r\nNortek *name* Data Interface\r\n" (*name* is replaced by the instrument host name) is sent to identify the instrument that has responded to the connection request. TCP ports 9001, 9002 and 9004 are assigned for the following uses:

- Port 9001 is used for machine driven control. This port requires username/password. The serial port data is translated directly into TCP/ IP over Ethernet. Binary data generated in measurement mode is visible on this port. Standard streaming record delineation techniques must be used in order to make sure that the received data is properly synchronized for decoding. A break can be sent by sending the string K1W%!Q<CR><LF> to the instrument or a Ctrl-C character (ASCII 0x03) (Ctrl-C has to be sent on its own and *not* embedded in any command). The internal application takes care of the appropriate timing of the break sent over the internal serial port. This port require username / password authentication. Refer to previous section for example. The password entry is ignored if password authentication, as shown in the web page configuration, is disabled (so any input, including an empty password, is accepted). The command and data record formats for the interfaces are the same as for the

serial port.

- Port 9002 is a data only channel which will output all data that is configured for serial output. This can, for example, be used by display only software while configuration is done by another application.
- Port 9004 outputs ASCII data (no binary) that is configured for serial output.

A telnet client should *not* be used to access these ports. Telnet incorporates its own binary protocol which is neither interpreted nor sent via the raw connection. Using a telnet client on these ports will result in extraneous characters being sent and certain binary characters being interpreted by the client.

3.3.3 FTP

The internal data recorder is accessed over Ethernet using a standard FTP (File Transfer Protocol) client. Together with the various telemetry options, the FTP data download serves as a simple way to download measured data at regular intervals if true real time operation is not required. Only the telemetry file can be deleted using FTP.

When an FTP connection is active, the internal state of the machine is changed so that commands are no longer processed (and an error is returned when commands are entered). Terminating the FTP connection or sending a BREAK followed by the CO command will switch the instrument back to the mode it was in before the FTP session began. If a break command is sent while an FTP transaction is in progress, the FTP connection will be forcibly terminated.

If an FTP connection is done when the instrument is in measurement mode (see Figure 1), the FTP connection is made through data retrieval mode. When the FTP connection is terminated, the instrument will then return to measurement mode. If there is no data transferred or FTP commands sent for 120 seconds, the FTP connection will terminate and the instrument will return to measurement mode.

For an example on how to configure the instrument to output a telemetry file and download the file to FTP, check out the next section.

3.3.4 HTTP

HTTP (Hypertext transfer protocol) can also be used for data transmission. For organizations with strong security / firewall restrictions, FTP access to the instrument may not be permitted. For that reason, a web page allowing individual data files to be downloaded has been implemented in the Ethernet processor. The web page can be accessed by clicking on the “Data Download (HTTP)” link from the main web page.

3.3.5 UDP

UDP (user datagram protocol) can also be used for data transmission. When using UDP, the data collection software simply waits for data to be sent from the instrument without having to establish a connection first. This may be useful for cases in which instrument power is intermittently interrupted and re-connecting to the instrument is not desirable. One downside to UDP communications is that transmission of the data is not guaranteed. On a noisy / error-prone connection, it is possible that the occasional datagram may be dropped. If every data record must be received, then TCP is recommended.

In order to use UDP in a power-safe configuration, the IP address of the data collection software and port must first be configured using the web interface. The IP address identifies the client to which the data is to be sent and the port may be used to uniquely identify the instrument to the application. The same port may be used for all instruments if the data collection software examines the IP address of the received datagram to identify the instrument. Once this information has been configured, the Ethernet processor will automatically send real-time data records to the configured address / port. An instrument in measurement mode re-enters measurement mode shortly after a power-cycle, so the data collection software will immediately receive new data without having to re-establish a connection.

3.3.6 PTP

Precision Time Protocol (IEEE-1588) is a standard used for distributing a high-resolution absolute time throughout an Ethernet network. The Signature series instrument can be configured to act as a slave to an existing PTP master clock (customer supplied) located in the same Ethernet LAN. The instrument contains a high-resolution clock which is synchronized and conditioned using PTP when enabled. The timestamps contained within the data records are then generated from this clock. When synchronized, these timestamps are typically aligned to within ~10 microseconds.

The PTP master clock must use UDP (layer three) and be configured for two-step operation with an end-to-end delay mode in order to be compatible with the Signature series PTP implementation. Using PTP does not affect the choice of UDP or TCP for the transport of data.

4 User Cases

Note that the Nortek Signature Series products command interface are ASCII based and line oriented (commands terminated with CR/LF). All commands should be set explicitly. The .deploy file created by the Deployment software is command-based and can be read directly into the command interface. Entering the .deploy file into the command interface can be a good starting point before modifying certain parameters using individual commands. Alternatively, use the Deployment software's "Customize..." function to create a .deploy file and input commands in the #CustomCommands section.

Comprehensive validation and error handling is implemented. The setup is verified when sending the SAVE command. If there is anything wrong with the deployment plan, i.e. some of the parameters are entered with values outside their specific range, an ERROR will be returned. The GETERROR command will describe why. If SAVE is not used, the deployment plan will be validated when sending the START or DEPLOY command. Note the difference between DEPLOY and START, the latter will immediately start a measurement any time the instrument state returns to Measurement mode such as by applying power or timeout from Data Retrieval Mode. If DEPLOY is used, be aware that if the deployment time has passed when the battery is connected, the instrument will resynchronize its data sampling according to the deployment time and the instrument configuration. This means you may have to wait for one average measurement interval or one burst measurement interval before the instrument starts to ping.

Invalid configurations return the ERROR with limits directly, so that each subsequent error can be handled until a valid configuration is achieved. Argument limits can be retrieved through commands. For example, if entering SETPLAN,MIAVG=5000, you will receive an OK. But when saving or deploying, you will receive an ERROR. When using GETERROR: 134,"Invalid setting: Plan Profile Interval","GETPLANLIM,MIAVG=([1;3600])" OK. The measurement interval must be within 1:3600 seconds. The valid range for the various arguments can also be verified by using the GETPLANLIM and GETAVGLIM commands.

Below you will find four examples illustrating the format and how to use the telemetry file.

4.1 Average velocity data and NMEA, Signature 55

Either use the Deployment wizard to create a .deploy file which can be uploaded via the Terminal Emulator, or set the configuration through commands (seen below). The .deploy file can also be uploaded then customized via commands once in the Terminal Emulator.

In this example: Signature55, configured to alternate between fine and coarse current profiles (3:1). In this case the user wanted to download the averaged fine profile upon request.

Configuration example:

```
%Recommended starting point for configuration file
SETDEFAULT,ALL
OK
%Setting plan for "Fine" profile
```

```
SETPLAN,MIAVG=600,AVG=1,DIAVG=0,VD=0,MV=10,SA=35,BURST=0,
MIBURST=120,DIBURST=0,SV=0,FN="Data.ad2cp",SO=0,FREQ=75
OK
SETAVG,NC=109,CS=5,BD=2,CY="ENU",PL=-6,AI=180,VR=1,DF=3,NPING=137,
NB=3,CH=0,MUX=0,BW="BROAD",ALTI=0,BT=0,ICE=0
OK
%Setting plan for "Coarse" profile
SETPLAN1,MIAVG=1800,AVG=1,DIAVG=0,VD=0,MV=10,SA=35,BURST=0,
MIBURST=120,DIBURST=0,SV=0,FN="Data.ad2cp",SO=0,FREQ=55
OK
SETAVG1,NC=54,CS=20,BD=2,CY="ENU",PL=-2,AI=180,VR=1,DF=3,NPING=60,
NB=3,CH=0,MUX=1,BW="NARROW",ALTI=0,BT=0,ICE=0
OK
%Setting the alternating measurement intervals and ratios of "Fine"
and "Coarse"
SETALTERNATE,EN=1,PLAN=1380,IDLE=10,PLAN1=180,IDLE1=230
OK
%Setting the telemetry file to average the "Fine" profile over the
averaging interval
SETTMAVG,EN=1,CD=1,PD=1,AVG=180,TV=1,TA=1,TC=1,CY="ENU",FO=1,SO=0,
DF=100
OK
SAVE,ALL
ERROR
%Finding where the error in the configuration is
GETERROR
"Invalid setting: Avg Average Interval too low for the configured
number of pings and profiling distance",LIM="GETAVG1LIM,AI=
([360;1800])"
OK
%Number of pings too high compared to desired averaging interval
with multiplex enabled.
SETAVG1,NPING=30
OK
SAVE,ALL
OK
```

Note that SETTMAVG,AVG must equal the AI set by SETAVG,AI. To set telemetry averaging for the alternate plan use SETTMAVG1, note that these will be recorded to the same telemetryfile.bin file.

Enter START or DEPLOY,TIME to begin the deployment.

4.2 Download telemetry file via FTP

In this example, a Signature1000 is set up to measure currents for 2 minutes every 10 minutes and waves every hour (4096 samples at 4 Hz). The raw current data are processed and a subset is saved as a telemetry file and made available on FTP.

Configuration example:

```
%Recommended starting point for configuration file
SETDEFAULT,ALL
OK
%Configuration for instrument:
SETPLAN,MIAVG=600,AVG=1,DIAVG=0,VD=0,MV=10,SA=35,BURST=1,
MIBURST=3600,DIBURST=0,SV=0,FN="Ex3.ad2cp",SO=0,FREQ=1000
OK
SETAVG,NC=21,CS=1,BD=0.2,CY="ENU",PL=0,AI=120,VR=2.5,DF=3,NPING=13,
NB=4,CH=0,MUX=0,BW="BROAD",ALTI=0,BT=0,ICE=0,ALTISTART=1,ALTIEND=30
OK
SETBURST,NC=13,NB=4,CS=1,BD=9.5,CY="BEAM",PL=0,SR=4,NS=4096,VR=2.5,
DF=3,NPING=1,CH=0,VR5=2.5,ALTI=1,BT=0,DISV=0,RAWALTI=1,
ALTISTART=4.8,ALTIEND=33.1
OK
%Configuration for telemetry file:
SETTMAVG,EN=1,CD=2,PD=1,AVG=120,TV=1,TA=1,TC=1,CY="ENU",FO=1,SO=0,
DF=100
OK
SAVE,ALL
OK
DEPLOY,TIME="2014-11-12 14:40:00"
OK
```

Go to *ftp://your-IP-address* to find the telemetry file (telemetryfile.bin). Here is part of the result from the above configuration. Note that the data were collected in air.

```
$PNORC,091715,142440,1,0.24,-1.35,-2.21,-1.69,1.37,169.7,C,79,84,67,102,11,13,8,11*2B
$PNORC,091715,142440,3,0.64,-0.28,-1.91,-1.32,0.70,113.9,C,79,84,66,96,12,14,7,20*13
$PNORC,091715,142440,5,0.08,-0.50,-1.76,-1.48,0.51,171.2,C,78,84,66,92,11,13,7,24*1D
$PNORC,091715,142440,7,-0.37,0.97,-1.02,-1.07,1.04,339.0,C,78,84,66,67,11,14,10,10*21
$PNORC,091715,142440,9,-0.94,0.57,-0.76,-1.11,1.10,301.1,C,78,83,65,69,12,15,9,10*10
$PNORC,091715,142440,11,-0.37,0.76,-0.95,-1.06,0.85,334.0,C,78,83,65,66,13,15,8,8*14
$PNORC,091715,142440,13,0.05,-0.25,-1.64,-1.36,0.26,168.4,C,78,84,66,82,11,14,9,33*2F
$PNORC,091715,142440,15,-0.20,0.20,-1.36,-1.32,0.28,314.6,C,78,84,66,67,11,13,9,7*16
$PNORC,091715,142440,17,0.19,0.17,-1.47,-1.13,0.25,48.0,C,78,84,65,69,12,16,9,2*0D
$PNORC,091715,142440,19,-0.91,0.45,-0.90,-1.19,1.02,296.5,C,78,84,65,66,12,14,10,8*27
$PNORC,091715,142440,21,-0.49,0.66,-1.00,-1.11,0.82,323.1,C,78,84,65,67,12,14,11,10*13
$PNORI,4,Signature1000900002,4,11,0.20,1.00,0*1B
$PNORS,091715,143440,00000000,2A4C0000,14.3,1300.0,278.3,15.7,-33.0,0.000,-262.45,0,0*65
```

```
$PNORC,091715,143440,1,0.76,-1.62,-2.45,-1.73,1.79,154.8,C,78,83,67,102,12,13,5,12*26
$PNORC,091715,143440,3,0.30,-0.77,-1.94,-1.50,0.83,158.6,C,78,83,66,97,12,14,9,17*19
$PNORC,091715,143440,5,-0.22,-1.19,-1.83,-1.66,1.21,190.4,C,78,84,66,91,11,13,8,22*36
$PNORC,091715,143440,7,-0.20,0.71,-1.09,-1.15,0.74,344.0,C,78,84,66,67,12,13,7,9*20
$PNORC,091715,143440,9,-0.30,0.94,-0.96,-0.97,0.99,342.0,C,78,84,65,66,11,15,9,8*25
$PNORC,091715,143440,11,0.20,0.82,-1.23,-1.09,0.85,13.3,C,78,84,66,67,13,14,6,8*09
$PNORC,091715,143440,13,0.11,0.46,-1.44,-1.19,0.48,13.5,C,78,84,65,75,11,13,8,1*04
$PNORC,091715,143440,15,-0.42,0.77,-1.05,-1.12,0.88,331.0,C,78,83,65,66,11,14,8,10*2D
$PNORC,091715,143440,17,-0.15,0.34,-1.29,-1.17,0.37,336.4,C,78,83,65,66,13,15,8,1*15
$PNORC,091715,143440,19,-0.79,0.50,-0.93,-1.13,0.93,302.5,C,78,84,65,66,12,15,10,10*10
$PNORC,091715,143440,21,-0.30,0.83,-1.08,-1.12,0.89,340.1,C,78,84,65,67,12,13,8,9*15
$PNORI,4,Signature1000900002,4,11,0.20,1.00,0*1B
```

After downloading the telemetry file, erase it either via FTP or commands. Only the telemetry file can be deleted using FTP.

```
%Erasing telemetry file
ERASETM,9999
OK
%Continuing the configured deployment plan
CO
OK
```

Note that the instrument does not process wave data internally (read more about this in the Operation Manual, if interested) thus only current data will be output in the telemetry file.

For use with external controller it can be interesting to note the following: If the instrument is started at e.g. 12:00, the first current profile is finished at 12:02 (120 seconds) and the next starts about 12:10. That leaves us with 8 minutes to download the telemetry file to FTP before next current profile starts. The clock drifts with about 1 sec/week. Since DEPLOY was used the measurement intervals will resynchronize according to the deployment time and the instrument configuration (see DEPLOY for more information), thus is should be easier to schedule automatic data download as the window 12:02 to 12:10 remains.

4.3 Download telemetry file over serial port

In this example the user wishes to download the telemetry file in 4096 byte chunks.

Connect via Terminal Emulator while the instrument is measuring

```
Send Break
CONFIRM
OK
%Going into Data Retrieval Mode
RM
NORTEK AS.
Version 2176 (Sep 17 2015 18:58:53)
DATA RETRIEVAL MODE
OK
%Checking the size of the telemetry file. Return in bytes
TMSTAT
95558
OK
%Outputting the telemetry file over serial port in 4096 byte chunks
DOWNLOADTM,0,4096,CKS=1
OUTPUT...
OK
%Next 4096 byte chunk, etc
DOWNLOADTM,4097,4096,CKS=1
OUTPUT...
OK
%Erasing telemetry file
ERASETM,9999
OK
%Continuing the configured deployment plan
CO
OK
```

Copy the returned text and paste to file. Or check "Record to file", the file will appear by default in: C:\Users\xxxx\Documents\Nortek\Deployment\Online

Parameters can be added to the DOWNLOADTM command to set start address, length of file, etc (see section DOWNLOADTM)

4.4 Erase telemetry file

In this example, the user wishes to erase the telemetry file after some period of time.

```
Send Break
CONFIRM
OK
%Going into Data Retrieval Mode
RM
NORTEK AS.
Version 2176 (Sep 17 2015 18:58:53)
DATA RETRIEVAL MODE
OK
%Checking the size of the telemetry file. Return in bytes
TMSTAT
34768
OK
%Erasing the telemetry file
ERASETM,9999
OK
%Continuing the configured deployment plan
CO
OK
```

The telemetry file can also be erased over FTP.

4.5 Checking instrument state over Ethernet

In this example a user connects to and powers the Ethernet port, but is unsure of the current operational state. If power is applied while in measurement mode, it will continue the measurement but not wake the Ethernet processor (BBP). If power is applied while in deployment state a re-synch will occur and resume sleep mode. Hence it is necessary to use BBPWAKEUP in both cases.

A typical sequence starts by wanting to know the state of the instrument before proceeding with either a new measurement or data retrieval.

```
%Waking up the BBP to make sure commands are received
BBPWAKEUP
OK
%Inquiring the state the of the instrument
GETSTATE
GETSTATE,MODE=0010,DEPTIME=27521,MEASTIME=27521,CURRTIME="2015-09-
28 11:21:16",WAKEUP=2
OK
```

This indicates the instrument has been configured to deploy and has started its scheduled deployment for 27521 seconds. See GETSTATE for more information.

Depending on the desired action, send Break usually followed by; either MC to enter command mode, RM for data retrieval or START/DEPLOY/CO to start/schedule/continue a deployment.

5 Commands

Valid Range: The valid range for the following commands are not listed because some of them depend on the actual instrument in use. However, the minimum and maximum values can be retrieved through the appropriate GETxxxxLIM command.

Example: send GETAVGLIM,CS to read the valid range of cell sizes.

Default values are not listed for all commands in this document as some of them depend on the actual instrument in use. Default parameters can be retrieved be setting default configuration (SETDEFAULT,ALL) and reading out the desired parameter through the appropriate GET command.

The same is the case for some of the minimum and maximum values that depend on the actual instrument in use. The parameter range for the various arguments can be retrieved through the appropriate GETxxxLIM command, e.g. GETAVGLIM,CS to read the valid range of cell sizes.

All command parameters should be set explicitly, e.g.
SETAVG,NC=10,BD=0.7

OK

A configuration of the instrument should always start with setting the default configuration, e.g.
SETDEFAULT,ALL

OK

5.1 List of Commands

Command	Description	Scope
START	Go in measurement mode	Command mode
MC	Go in command mode	Confirm mode
RM	Go in data retrieval mode	Confirm mode
CO	Continue in measurement mode.	Confirm mode, Data retrieval mode
INQ	Inquiry instrument state	All modes
SETINST / GETINST / GETINSTLIM	Set/Get Main Instrument Settings Get Instrument Setting Limits	Command mode
SETCLOCK/ GETCLOCK	Set/Get Real Time Clock	Command mode, Data retrieval mode
SETCLOCKSTR/ GETCLOCKSTR	Set/Get Real Time Clock using a string argument	Command mode, Data retrieval mode
SETPLAN/ GETPLAN/ GETPLANLIM	Set/Get Plan Settings Get Plan Limits	Command mode
SETAVG/	Set/Get Averaging Mode Settings	Command mode

Command	Description	Scope
GETAVG/ GETAVGLIM	Get Averaging Mode Limits	
SETBURST/ GETBURST/ GETBURSTLIM	Set/Get Burst Profile Settings Get Burst Limits	Command mode
SETBURSTHR/ GETBURSTHR/ GETBURSTHRLIM	Set/Get Burst HR Profile Settings Get Burst HR Limits	Command mode
SETBT/ GETBT/ GETBTLIM	Set/Get Bottom Track Settings Get Bottom Track Limits	Command mode
SETALTERNATE/ GETALTERNATE/ GETALTERNATELIM	Set alternating, dual configuration times.	Command mode
GETMEM	Returns the amount of data the various modes will store to the recorder in Mbytes/hour	Command mode
SETTRIG/ GETTRIG/ GETTRIGLIM	Sets / gets the parameters and limits for Trigger.	Command mode
TRIG	Used for trigging measurement when Trigger is enabled and trigger type equals “COMMAND”	Measurement mode
GETPWR	Returns the amount of power the various modes will consume in mWatts	Command mode
GETPRECISION/ GETPRECISION1	Returns the precision along beam and horizontally for the various velocity profiles	Command mode
SETUSER/ GETUSER	Set/Get User Settings	Command mode
GETHW	Returns Firmware versions and Board revisions.	All modes
ID	Returns system name and ID number	Command mode
SETDEFAULT	Reload default settings.	Command mode
SAVE	Save current settings for next measurement.	Command mode
DEPLOY	Deploy the instrument	Command mode
FWRITE	Write tag/String to file.	Command mode, Confirmation mode, Data retrieval mode
POWERDOWN	Go in power down.	Command mode
ERASE	Erase the recorder	Command mode
FORMAT	Format the recorder	Command mode
LISTFILES	Lists the files stored on the Instrument recorder.	Command mode, Data retrieval mode

Command	Description	Scope
DOWNLOAD	Enables reading a file at the instrument recorder.	Command mode, Data retrieval mode
INQ	Inquires the instrument state	All modes
GETSTATE	Returns information about the current operational state of the instrument	All modes
GETERROR	Retrieves a full description of the last error condition to occur	All modes
GETALL	Retrieves all relevant configuration information for the instrument.	Command mode
RECSTAT	Returns Recorder Statistics	Command mode, Data retrieval mode
GETMISCLIM	Returns configuration limits that cannot be returned through the relevant commands	Command mode
GETXFAVG/ GETXFBURST	Returns the “Beam to XYZ” transfer matrix for the current setup.	Command mode
ADDLINE		Command mode
SETTMAVG/ GETTMAVG/ GETTMAVGLIM	Set/Get Averaging Mode Telemetry Settings Get Averaging Mode Telemetry argument limits	Command mode
SETTMBURST / GETTMBURST / GETTMBURSTLIM	Set/Get Burst Telemetry Settings Get Burst Telemetry argument limits	Command mode
SETTMALTI/ GETTMALTI/ GETTMALTILIM	Set/Get Altimeter Telemetry Settings Get Altimeter Telemetry argument limits	Command mode
TMSTAT	Returns number of bytes in the telemetry file	Command mode, Data retrieval mode
DOWNLOADTM	Download telemetry data	Command mode, Data retrieval mode
STOREHEADERTM		Command mode
ERASETM	Erase telemetry file	Command mode, Data retrieval mode
BBPWAKEUP	Wakes up the Doppler processor | Ethernet interface only, see Ethernet description	All modes

5.2 SETINST/GETINST/GETINSTLIM

Set/get main instrument settings and limits.

Argument	Description
BR	Baud Rate
RS	Serial protocol
LED	Enable/disable LED blink in head. When set to “ON24H” the LED will illuminate the first 24 hours of the measurement.
ORIENT	Sets the instrument orientation
CMTOUT	Command mode timeout.
DMTOUT	Data retrieval mode timeout.
CFMTOUT	Confirmation mode timeout.

5.3 SETCLOCK/GETCLOCK

Set or retrieve the Real Time Clock. Note that all parameters must be set when using the **SETCLOCK** command.

Argument	Description
YEAR	Year
MONTH	Month
DAY	Day
HOUR	Hours (24 hour format)
MINUTE	Minutes
SECOND	Seconds

5.4 SETCLOCKSTR/GETCLOCKSTR

Set or retrieve the Real Time Clock using a string. The format must be exactly as shown.

Argument	Description
TIME	yyyy-mm-dd hh:mm:ss

Example:

```
$PNOR,GETCLOCKSTR*64
$PNOR,GETCLOCKSTR,TIME="2014-11-12 14:27:42"*42
$PNOR,OK*2B
```

5.5 SETPLAN/GETPLAN/GETPLANLIM

The plan parameters specify directly (time) or indirectly (depth) which type(s) of measurement that will be measured and at the interval between the various types of measurements.

Argument	Description
MIAVG	Averaging Interval (s)
AVG	Averaging Mode disabled/enabled
DIAVG	Depth interval (m) Not yet implemented
VD	Vertical direction
MV	Unused (Not yet implemented) (Absolute max Vertical velocity (cm/s))
SA	Salinity (ppt)
BURST	Burst measurement disabled/enabled
MIBURST	Burst Interval (s)
DIBURST	Depth interval (m)
SV	Sound velocity (m/s)
FN	Filename of the raw data file where all the measured binary data will be stored
SO	Serial output
FREQ	Transmit frequency
NSTT	Number of Time Slots. Set to 0 giving the default number of slots.

The valid range for the various arguments should be verified using the GETPLANLIM command, also for the values listed here as they may change with firmware versions and instrument frequencies.

5.6 SETAVG/GETAVG/GETAVGLIM

Set/get averaging mode settings and get the relevant limits.

Argument	Description
NC	Number of cells
CS	Cell Size (m)
BD	Blanking Distance (m)
CY	Coordinate System (“ENU”, ”XYZ“, “BEAM”)
PL	Power Level [dB] (-100 dB to switch off transmit -20.0 dB to 0.0 dB)
AI	Average interval (s)
VP	Unused
VR	Velocity range along beam [m/s]
DF	Data Format 0 – AD2CP format 1. 1 –Legacy Aquapro format 2 - AD2CP format 2 3 - AD2CP format 3
NPING	Number of pings
NB	Number of beams (Select number of beams, 0 select beams according the PLAN,VD setting)
CH	Beam selection (Select beams, 0 select beams according the PLAN,VD setting. Example: 134 select the three beams 1, 3 and 4)
MUX	Multiplexor Selection 0 - 1, 0, ping all beams in parallel 1, ping beams in sequence
BW	Bandwidth selection. (“NARROW”, “BROAD”)
ALTI	Enable altimeter in AVG measurements
BT	Enable bottom track measurement in AVG sampling
ICE	Enabled ice velocity measurement in AVG sampling
ALTISTART	Altimeter start of measurement distance [m]
ALTIEND	Altimeter end of measurement distance [m]
RAWALTI	Storage of Raw Altimeter data

The actual valid range for the various parameters for the firmware version is used can be found by using the **GETAVGLIM** command. This command has the same arguments as the **SETAVG/GETAVG** commands shown in the list above. The output format for limits is described in Data Limit Formats

5.7 SETBURST/GETBURST/GETBURSTLIM

Set/get burst profile settings and get the relevant limits

Argument	Description
NC	Number of cells
NB	Number of beams
CS	Cell Size (m)
BD	Blanking Distance (m)
CY	Coordinate System
PL	Power Level
SR	Sampling rate (Hz)
NS	Number of samples
VR	Velocity range along beam
VP	Unused
DF	Data Format
NPING	Number of pings
CH	Beam selection
ALTI	Enable altimeter in BURST measurements
VR5	Velocity range along beam 5
BT	Enabled bottom track in BURST measurements
DISV	Disable Velocity measurement.
ECHO	Unused - Not yet implemented.
RAWALTI	Enable Storage of Raw Altimeter data. Raw data are store for first and last ping in each ensemble.
ALTISTART	Altimeter start of measurement distance [m]
ALTIEND	Altimeter end of measurement distance [m]
HR	Enable HR on Slanted beams
HR5	Enable HR on Vertical Beam

The actual valid range for the various parameters for the firmware version is used can be found by using the **GETBURSTLIM** command. This command has the same arguments as the **SETBURST**/ **GETBURST** commands shown in the list above. The output format for limits is described in Data Limit Formats.

5.8 SETBURSTHR/GETBURSTHR/GETBURSTHRLIM

Set/get burst HR profile settings and get the relevant limits

Argument	Description
PROC	0 - Pulse Coherent Processing using a single Ambiguity 1 - Pulse Coherent Processing with Extended Velocity Range (EVR)
LAG	Distance between two transmit pulses (slanted beams)
LAG5	Distance between two transmit pulses (vertical beam)
SCORR	Number of Ambiguities to resolve when using EVR
NC	Number of cells
CS	Cell size (m)
BD	Blanking distance (m)
PL	Power Level

5.9 SETBT/GETBT/GETBTLIM

Set/get bottom track settings and get the relevant limits

Argument	Description
RANGE	Maximum range
VR	Velocity range along beam
NB	Number of beams
CH	Beam selection
DF	Data format
PL	Power Level

5.10 SETALTERNATE/GETALTERNATE/GETALTERNATELIM

The SETALTERNATE/GETALTERNATE command allows two different configurations to be run consecutively in time. The primary configuration (defined by SETPLAN, SETBURST, SETAVG, SETTMAVG, SETBT) is run for "PLAN" seconds, after which the unit powers down for a given period of time ("IDLE" seconds). The alternate configuration (defined by SETPLAN1, SETBURST1, SETAVG1, SETTMAVG1, SETBT1) is then run for "PLAN1" seconds and the unit powers down for "IDLE1" seconds. The configuration is then switched back to the primary and the process is repeated.

The valid range for the various arguments should be verified using the GETALTERNATELIM command. The values listed here may change with firmware versions and instrument frequencies.

Argument	Description
EN	Enable or disable the alternate configuration mode
PLAN	Primary configuration run time (s)
IDLE	Primary configuration idle time (s)
PLAN1	Alternate configuration run time (s)
IDLE1	Alternate configuration idle time (s)

The actual valid range for the various parameters for the firmware version is used can be found by using the GETAVGLIM command. This command has the same arguments as the SETAVG/GETAVG commands shown in the list above. The output format for limits is described in Data Limit Formats.

5.11 GETMEM

Returns the amount of memory that will be stored on the recorder for the burst and average data as well as the combined plan value. Alternate mode values is also supported

Argument	Description
PLAN	Combined burst and average memory [Mbytes/hour]
BURST	Burst memory [Mbytes/hour]
AVG	Average memory [Mbytes/hour]
PLAN1	Combined burst1 and average1 memory [Mbytes/hour]
BURST1	Alternate burst1 memory [Mbytes/hour]
AVG1	Alternate average1 memory [Mbytes/hour]
TOTAL	Total memory value [Mbytes/hour]

5.12 SETTRIG/GETTRIG/GETTRIGLIM

Sets / gets the parameters and limits for Trigger. The available trigger types will depend on the harness/cable used with the instrument.

When triggered the instrument will perform a complete ping (Tx and Rx) before it goes back to monitoring the trigger. Any triggers asserted during an ongoing ping will be ignored. There are no specific requirements for pulse length. The unit triggers on the edge(s) of the trigger signal and can be triggered on rising, falling or both edges.

Argument	Description
EN	Enable/disable trigger functionality
TRIG	Specifies trigger type
TRIGOUTPUT	Enable trigger output

Note: TRIGOUTPUT=1 enables master trig output, RS485EDGE trigger must be used with this option. This enables several instruments to be synchronized together through RS485 with one of the instruments acting as master. Only continuous measurement configurations are supported in this mode.

5.13 TRIG

Command used for trigging measurement when Trigger is enabled and trigger type equals “COMMAND”.

Argument	Description
ID	Counting number

5.14 GETPWR

Returns the power consumption in milliWatts for the various measurements enabled as well as the overall value. The plan values include the sleep mode power consumption in addition to the sum of average and burst mode values. The reported values are adjusted according to the input voltage to the system when the command is executed.

Argument	Description
PLAN	Combined burst and average power [mWatt]
BURST	Burst power [mWatt]
AVG	Average power [mWatt]
PLAN1	Combined burst1 and average1 power [mWatt]
BURST1	Alternate burst1 power [mWatt]
AVG1	Alternate average1 power [mWatt]
TOTAL	Total power value [mWatt]

5.15 GETPRECISION

Returns the precision in the horizontal range and along the beam in cm/s for the various measurement modes.

Argument	Description
AVGHORZ	Precision in the horizontal range in average mode [cm/s]
BURSTHORZ	Precision in the horizontal range in burst mode [cm/s]
BEAM5	Precision in the vertical range in burst mode [cm/s]
AVGBEAM	Precision along beam in average mode [cm/s]
BURSTBEAM	Precision along beam in average mode [cm/s]

5.16 GETPRECISION1

Returns the precision in the horizontal range and along the beam in cm/s for the various measurement modes during alternate mode settings.

Argument	Description
AVGHORZ	Precision in the horizontal range in average1 mode [cm/s]
BURSTHORZ	Precision in the horizontal range in burst1 mode [cm/s]
BEAM5	Precision in the vertical range in burst1 mode [cm/s]
AVGBEAM	Precision along beam in average1 mode [cm/s]
BURSTBEAM	Precision along beam in average1 mode [cm/s]

5.17 SETUSER/GETUSER

Argument	Description
POFF	Pressure offset (dbar)
DECL	Magnetic declination (degrees)
HX	Hard iron x-component
HY	Hard iron y-component
HZ	Hard iron z-component

5.18 GETHW

Returns Firmware versions and Board revisions.

Argument	Description	Type
FW	Running DSP FW version	Number
FPGA	Running FPGA FW version	Number
DIGITAL	Board Revision. Example: C-0	String
INTERFACE	Board Revision. Example: C-0	String
ANALOG	Board Revision. Example: C-0	String
SENSOR	Board Revision. Example: C-0	String
BOOT	DSP Bootloader FW Version	Number
MINOR	Running DSP FW version (minor part)	Number

5.19 ID

Returns System name and serial number.

Argument	Description
STR	System name, maximum 15 characters
SN	Serial number

Example:

```
ID
"Signature1000",900002
ID,STR
"Signature1000"
```

5.20 SETDEFAULT

Reload default settings

Argument	Description
ALL	Restore all settings below **except** USER and INST to default values.
AVG	Restore AVG default.
INST	Restore INST default.
TMAVG	Restore TMAVG default.
PLAN	Restore PLAN default.
BURST	Restore BURST default.
PTP	Restore PTP default.
BT	Restore BT default.
USER	Restore USER default.
TMBURST	Restore TMBURST default.
TMALTI	Restore TMALTI default.
DVL	Restore DVL default.

5.21 SAVE

Save current settings for next measurement. At least one argument must be specified for the **SAVE** command.

Argument	Description
ALL	Save all settings.
AVG	Save AVG parameters.
INST	Save INST parameters.
TMAVG	Save Telemetry AVG parameters.
PLAN	Save PLAN parameters.
BURST	Save BURST setting parameters.
PTP	Save PTP parameters.
TMBT	Save Telemetry BT parameters.
USER	Save USER parameters.
TMBURST	Save TMBURST setting parameters.
TMALTI	Save TMALTI Profile parameters.
DVL	Save DVL setting parameters.

5.22 DEPLOY

Deploy the instrument. Start the measurement at the time specified in the string argument. The format must be exactly as shown. If no time value is passed, the deployment will start immediately.

The number of seconds to the specified deployment time is returned.

Note the difference between DEPLOY and START, the latter will immediately start a measurement any time the instrument state returns to Measurement mode such as by applying power or timeout from Data Retrieval Mode. If DEPLOY is used, be aware that if the deployment time has passed when the battery is connected, the instrument will resynchronize its data sampling according to the deployment time and the instrument configuration. This means you may have to wait for one average measurement interval or one burst measurement interval before the instrument starts to ping.

Argument	Description
TIME	yyyy-mm-dd hh:mm:ss

Example:

```
DEPLOY,TIME="2014-11-12 14:40:00"
592
OK
```

5.23 FWRITE

Write tag/String to file.

Argument	Description
FNUM	File identifier for telling which file the info is written to. 0 – File defined in the PLAN command. 1 – telemetry file Default value : 0
ID	Identifier Tell the parser how to interpret the string. Default value: 0 0 - Comment 1 - Dive Location 2 - Surface Location
STR	String (Maximum 200 bytes long)
B64STR	B64 coded string (maximum 200 bytes long)

Note: Fwrite STR and B64STR cannot be set together.

5.24 POWERDOWN

Power down the instrument to set it in sleep mode.

5.25 ERASE

Argument	Description
CODE	Code should be 9999

Erase all files on the recorder.

5.26 FORMAT

Argument	Description
CODE	Code should be 9999

Format the recorder. Note that this can take minutes depending on the recorder size.

5.27 LISTFILES

Lists the files stored on the Instrument recorder.

Argument	Description
OPT	OPT=”la” – lists extended information.

5.28 DOWNLOAD

This command enables reading a file at the instrument recorder.

Argument	Description
FN	Filename
SA	Start address (offset) of the first byte to be returned.
LEN	Number of bytes to be downloaded.
CRC	Use Cyclic redundancy check. CRC=1 enables crc.
CKS	Use Checksum. CKS=1 enables checksum.

If no other parameters than the file name are sent with the DOWNLOAD command the complete file is directly returned, without the number of bytes to follow. The end of the file can then be detected by parsing the OK<CR><LF>.

The parameters can be used to download the file in several pieces. The number of bytes to follow will then be returned in ASCII format and terminated with <CR><LF> before the data is output. The end of file stream is terminated with OK<CR><LF>. A cyclic redundancy check or a checksum will then be added to be able to verify data integrity during download. The complete file can also be downloaded in this way by specifying SA=0 and a large value for LEN. The actual file size is then returned before the data follows.

Example:

```
DOWNLOAD,FN="TestFile.ad2cp",SA=0,LEN=4096,CRC=1,CKS=0<CR><LF>
4096<CR><LF>
<binary or ASCII data>
23432<CR><LF> (CRC value)
OK<CR><LF>
```

5.29 INQ

The INQ command inquires the instrument state. Note that when operating over RS232 or RS422 serial lines, it should be preceded with @@@@@@ <delay 400 ms> and a flush of the input buffer in case the instrument is in power down or in a low power mode taking measurements.

Consult this section a description of the Instrument modes.

Parameter	Instrument Mode
0000	Bootloader/Firmware upgrade
0001	Measurement
0002	Command
0004	Data Retrieval
0005	Confirmation
0006	FTP-mode

Example (in command mode) :

```
08:43:31   INQ<CR><LF>
08:43:31   0002<CR><LF>
```

Example (in measurement mode) :

```
08:43:31   INQ<CR><LF>
08:43:31   0001<CR><LF>
```

Example (in confirmation mode) :

```
08:43:31   INQ<CR><LF>
08:43:31   0005<CR><LF>
```

Example (in data retrieval mode) :

```
08:43:31   INQ<CR><LF>
08:43:31   0004<CR><LF>
```

Example (in firmware upgrade mode) :

```
08:43:31   INQ<CR><LF>
08:43:31   0000<CR><LF>
```

5.30 GETSTATE

Returns information about the current operational state of the instrument

Argument	Description
MODE	Current instrument state 0001 – Measurement (START command received) 0002 – Command 0003 – Deploy (DEPLOY command received and deployment running) 0004 – Data retrieval 0005 – Confirmation 0006 – Network FTP 0008 – Pre-deployment (DEPLOY command received, but deployment has not yet started) 0009 –Confirmation (measurement) 0010 – Confirmation (deploy) 0011 – Confirmation (pre-deploy) 0012 – Data retrieval (internal processing in progress)
DEPTIME	0 – DEPLOY command has not been received. < 0 – Number of seconds until deployment starts. > 0 – Number of seconds that deployment has been running.
MEASTIME	0 – START command has not been received. > 0 – Number of seconds that measurement has been running.
CURRTIME	The current instrument time Time format is “YYYY-MM-DD HH:MM:SS”
WAKEUP	Reason for instrument wakeup 0 – Bad power 1 – Power on 2 – Break 3 – Real-time clock
INTPROC	Internal processing Active

5.31 GETERROR

GETERROR retrieves a full description of the last error condition to occur. The error number is returned first followed by a string with the text description of the last error condition. A second string is also returned which contains information on the valid range of the failing argument, see example below.

Argument	Description
NUM	Integer error value
STR	Text description

Example:

```
SETAVG,CS=2.5
OK
SAVE,ALL
ERROR
GETERROR
40,"Invalid setting: Avg Cell Size","GETAVGLIM,CS=([0.20;2.00])"
OK
```

5.32 GETALL

GETALL retrieves all relevant configuration information for the instrument. This information can either be displayed on the command line or saved to a data file. For the SignatureWaves software to read a valid .ad2cp file it must contain both the Header and Data Record. The Header information can be obtained by using the command GETALL.

Argument	Description
FN	Write the output to this file

Example :

```
GETALL<CR><LF>
GETPLAN,600,1,0,0,10,0.0,1,0,0,1500,"",1<CR><LF>
GETAVG,20,1.00,0.30,"BEAM",-12.0,1,0.000,1.29,3,1,0,0<CR><LF>
GETBURST,50,4,0.400,0.200,"BEAM",0.0,1,1024,4.00,0.000,0,1,0<CR><LF
>
GETUSER,0.00,0.00,0,0,0<CR><LF>
GETINST,9600,232,1<CR><LF>
BEAMCFGLIST,1,10.00,20.00,1000,500,1,1<CR><LF>
BEAMCFGLIST,2,10.00,20.00,1000,500,1,2<CR><LF>
BEAMCFGLIST,3,10.00,20.00,1000,500,1,3<CR><LF>
BEAMCFGLIST,4,10.00,20.00,1000,500,1,4<CR><LF>
OK<CR><LF>
```

5.33 RECSTAT

Return Recorder Statistics

Argument	Description	Description
SS	SectorSize	# of Bytes in a Sector.
CS	ClusterSize	# of Bytes in one Cluster
FC	Free Clusters	# of Bytes in Free Clusters
TC	Total Clusters	Total # of bytes in Clusters
VS	Volume Size	Volume Size in bytes

5.34 GETMISCLIM

This command returns configuration limits that cannot be returned through the relevant commands.

Argument	Description
AVGPR	Returns the total profiling range for averaged measurements (SETAVG).
BURSTPR	Returns the total profiling range for burst profile measurements (SETBURST).
BURSTHRPR	Returns the total profiling range for burst HR profile measurements (SETBURSTHR)

The output format for limits is described in Data Limit Formats.

5.35 GETXFAVG / GETXFBURST

Returns the “Beam to XYZ” transfer matrix for the current setup. If the number of beams are 1 or 2 only ROWS and COLS are returned.

Argument	Description
ROWS	Number of Rows.
COLS	Number of Columns.
M11	
M12	
M13	
M14	
M21	
M22	
M23	
M24	
M31	
M32	
M33	
M34	
M41	
M42	
M43	
M44	

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \qquad \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

Figure: Matrix definitions

Examples:

```
GETXFBURST,ROWS=4,COLS=4,M11=1.183,M12=0.000,M13=-1.183,M14=0.000,
M21=0.000,M22=1.183,M23=0.000,M24=-1.183,M31=0.552,M32=0.000,
M33=0.552,M34=0.000,M41=0.000,M42=0.552,M43=0.000,M44=0.552
```

```
GETXFAVG,ROWS=3,COLS=3,M11=1.183,M12=0.000,M13=-1.183,M21=1.183,
M22=-2.366,M23=1.183,M31=0.552,M32=0.000,M33=0.552
```

5.36 SETTMAVG/GETTMAVG/GETTMAVGLIM

Set/get averaging mode telemetry settings and get the relevant argument limits.

Argument	Description
EN	Enable Averaging Mode Telemetry
CD	Cells Divisor
PD	Packets Divisor
AVG	Average Telemetry Data
TV	Store Velocity
TA	Store Amplitude
TC	Store Correlation
CY	Coordinate System
FO	Enable File Output
SO	Enable Serial Output
DF	Data format: (See Chapter 4)
DISTILT	Disable tilt
TPG	Enable storage of Percentage Good Data

The actual valid range for the various parameters for the firmware version is used can be found by using the **GETTMAVGLIM** command. This command has the same arguments as the **SETTMAVG/ GETTMAVG** commands shown in the list above. The output format for limits is described in Data Limit Formats.

5.37 SETTMBURST/GETTMBURST/GETTMBURSTLIM

Argument	Description
EN	Enable burst telemetry
NS	Number of burst samples to save 0 – Save all burst samples 1 – BURST.NS
SO	Enable Serial Output
FO	Enable File Output
DF	Data Format

The actual valid range for the various parameters for the firmware version is used can be found by using the GETTMBURSTLIM command. This command has the same arguments as the SETTMBURST/GETTMBURST commands shown in the list above. The output format for limits is described in Data Limit Formats.

5.38 SETTMALTI/GETTMALTI/GETTMALTILIM

Sets / gets the parameters and limits for altimeter.

Argument	Description
EN	Enable/disable altimeter telemetry
TS	Include time stamp
TQ	Include quality parameter
FO	File output
SO	Serial output
DF	Altimeter Telemetry format. 200 – NMEA (PNORA) format without Tags. 201 – NMEA (PNORA) format with Tags.

Example:

```
SETTMALTI,1,1,0,1,201
GETTMALTI
GETTMALTILIM
```

5.39 SETTMBT/GETTMBT/GETTMBTLIM

Sets / gets the parameters and limits for bottom track telemetry.

Argument	Description
EN	Enable/disable bottom track telemetry
FO	File output
SO	Serial output
DF	Bottom track Telemetry format. 300 – NMEA without tags. 301 – NMEA with tags. 302 – NMEA without tags and Sensor Data. 303 – NMEA with tags and Sensor Data.
NPING	Number of Pings
FOMTH	Figure of Merit threshold.
CY	Coordinate system

Example:

```
SETTMBT,1,1,1,301
GETTMBT
GETTMBTLIM
```

5.40 TMSTAT

This command returns the length (# of bytes) of the telemetry file.

Example:

```
TMSTAT<CR><LF>
13500<CR><LF>
OK<CR><LF>
```

5.41 DOWNLOADTM

This command enables reading the telemetry file which can be created during measurement by using the appropriate SETTMxxx commands. The formats are described in the section detailing Data Formats.

Argument	Description
SA	Start address (offset) of the first byte to be returned.
LEN	Number of bytes to be downloaded. (Length of file)
CRC	Use Cyclic redundancy check. CRC=1 enables CRC. CRC cannot be enabled when CKS=1
CKS	Use Checksum. CKS=1 enables checksum. CKS cannot be enabled when CRC=1

If no parameters are sent with the **DOWNLOADTM** command the complete file is directly returned, without the number of bytes to follow. The end of the file can then be detected by parsing the OK<CR><LF>.

The parameters can be used to download the telemetry file in several pieces. The number of bytes to follow will then be returned in ASCII format and terminated with <CR><LF> before the data is output. The end of telemetry stream is terminated with OK<CR><LF>. A cyclic redundancy check or a checksum will then be added to be able to verify data integrity during download. The complete file can also be downloaded in this way by specifying SA=0 and a large value for LEN. The actual file size is then returned before the data follows. See also TMSTAT for retrieving file information.

Example:

```
DOWNLOADTM,0,4096,CRC=1,CKS=0<CR><LF>
4096<CR><LF>
<binary or ASCII data>
23432<CR><LF> (checksum/crc value)
OK<CR><LF>
```

5.42 STOREHEADERTM

This command stores the GETALL (complete configuration) to the telemetry file.

5.43 ERASETM

Argument	Description
CODE	Code should be 9999

Erase the telemetry file. The telemetry file can also be erased over FTP.

6 Data formats

Note: All data of the AD2CP interface are stored/sent as Little Endian. Each output data packet sent/stored by the AD2CP consists of a Header part and a Data Record part:

Header
Synchronization, ID, length and Checksums.
Data Record
Data

For the SignatureWaves software to read a valid .ad2cp file it must contain both the Header and Data Record. The Header information can be obtained by using the command GETALL.

The following chapters describe the format of the Header and the different variants of the Data Record.

6.1 Header Definition

The Header consists of the following fields:

Field	Size	Description
Sync	8 bits	Always **0xA5**
Header Size	8 bits (unsigned)	Size (number of bytes) of the Header.
ID	8 bits	Defines type of the following Data Record. **0x15** – Burst Data Record. **0x16** – Average Data Record. **0x17** – Bottom Track Data Record. **0x18** – Interleaved Burst Data Record (beam 5). **0x1A -** Burst Altimeter Raw Record. **0x1B -** DVL Bottom Track Record. **0x1C -** Echo Sounder Record. **0x1D -** DVL Water Track Record. **0x1E -** Altimeter Record. **0x1F -** Avg Altimeter Raw Record. **0xA0** - String Data Record, eg. GPS NMEA data, comment from the FWRITE command.
Family	8 bits	Defines the Instrument Family. **0x10** – AD2CP Family
Data Size	16 bits (unsigned)	Size (number of bytes) of the following Data Record.
Data Checksum	16 bits	Checksum of the following Data Record.
Header Checksum	16 bits	Checksum of all fields of the Header (excepts the Header Checksum itself).

C-style Header Struct Definition

```
typedef struct
{
      unsigned char  sync;
      unsigned char  hdrSize;
      unsigned char  ID;
      unsigned char  family;
      unsigned short dataSize;
      unsigned short dataChecksum;
      unsigned short hdrChecksum;
} CommandHeader_t;
```

6.1.1 Checksum Definition

The Checksum is defined as a 16-bits unsigned sum of the data (16 bits). The sum shall be initialized to the value of 0xB58C before the checksum is calculated.

C-code for Checksum calculations:

```
unsigned short calculateChecksum(unsigned short *pData, unsigned
short size)
{
    unsigned short checksum = 0xB58C;
    unsigned short nbshorts = (size >> 1);
    int i;
    for (i = 0; i < nbshorts; i++)
    {
        checksum += *pData;
        size  -= 2;
        pData++;
    }
    if (size > 0)
    {
        checksum += ((unsigned short)(*pData)) << 8;
    }
    return checksum;
}
```

6.1.2 Burst/Average Data Record Definition (DF3)

Field	Size	Format	Resolution/ Unit	Description	
Version	8 bits			Version number of the Data Record Definition. (3)	
offsetOfData	8 bits	Unsigned	#Bytes	Number of bytes from start of record to start of data (velocity/amplitude/correlation)	
Configuration	16 bits			Record Configuration Bit Mask	
				Bit 0	Pressure sensor value valid

Field	Size	Format	Resolution/ Unit	Description	
				Bit 1	Temperature sensor value valid
				Bit 2	Compass sensor values valid
				Bit 3	Tilt sensor values valid
				Bit 4	-
				Bit 5	Velocity data included
				Bit 6	Amplitude data included
				Bit 7	Correlation data included
				Bit 8	Altimeter data included
				Bit 9	Altimeter Raw data included
				Bit 10	AST data included
				Bit 11	Echo Sounder data included
				Bit 12	AHRS data icluded
				Bit 13	Percentage Good data included
				Bit 14	Std. Dev. data included
				Bit 15	Unused
Serial Number	32 bits				
Year	8 bits	Unsigned	1 Year	Years since 1900 (see struct tm definition)	
Month	8 bits	Unsigned	1 Month	Jan =0, Feb= 1, etc.(see struct tm definition)	
Day	8 bits	Unsigned	1 Day	(see struct tm definition)	
Hour	8 bits	Unsigned	1 Hour	(see struct tm definition)	
Minute	8 bits	Unsigned	1 Minute	(see struct tm definition)	
Seconds	8 bits	Unsigned	1 Second	(see struct tm definition)	
Microsec100	16 bits	Unsigned	100 µsec		
Speed of Sound	16 bits	Unsigned	0.1 m/s		
Temperature	16 bits	Signed	0.01 Degree Celsius		
Pressure	32 bits	Unsigned	0.001 dBar		
Heading	16 bits	Unsigned	0.01 Deg		
Pitch	16 bits	Signed	0.01 Deg		
Roll	16 bits	Signed	0.01 Deg		
#Beams & Coordinate system & #Cells	16 bits			Definition: (Standard)	
				Bit 9 - 0	Number of Cells (NC)

Field	Size	Format	Resolution/ Unit	Description	
				Bit 11 - 10	Coordinate system, b00: ENU, b01: XYZ, b10: BEAM, b11: -
				Bit 15 – 12	Number of Beams (NB)
				Definition: (Echo Sounder)	
				Bit 15-0	Number of echo sounder cells
Cell Size	16 bits	Unsigned	1 mm		
Blanking	16 bits	Unsigned	1 cm		
Nominal Correlation	8 bits	Unsigned	%	The nominal correlation for the configured combination of cell size and velocity range	
Temperature pressure sensor	8 bits	Unsigned	0.2 deg C	Temperature of pressure sensor: T=(Val/5)-4.0	
Battery Voltage	16 bits	Unsigned	0.1 Volt		
Magnetometer Raw(X-axis)	16 bits	Signed		Magnetometer Raw, X axis value in last measurement interval.	
Magnetometer Raw(Y-axis)	16 bits	Signed		Magnetometer Raw, Y axis value in last measurement interval.	
Magnetometer Raw(Z-axis)	16 bits	Signed		Magnetometer Raw, Z axis value in last measurement interval.	
Accelerometer Raw (X-axis)	16 bits	Signed		Accelerometer Raw X axis value in last measurement interval. (16384 = 1.0)	
Accelerometer Raw (Y-axis)	16 bits	Signed		Accelerometer Raw Y axis value in last measurement interval. (16384 = 1.0)	
Accelerometer Raw (Z-axis)	16 bits	Signed		Accelerometer Raw Z axis value in last measurement interval. (16384 = 1.0)	
Ambiguity Velocity / Echo sounder Frequency	16 bits	Unsigned	Standard Definition		
			10^(Velocity scaling) m/s	Ambiguity velocity, corrected for sound velocity, scaled according to Velocity scaling	
			Echo Sounder Definition		
				Number of Echo Sounder Cells	
Data Set Description	16 bits			Bits	Description
				0-3	Physical beam used for 1st data set.
				4-7	Physical beam used for 2nd data set.
				8-11	Physical beam used for 3th data set.
				12-16	Physical beam used for 4th data set.

Field	Size	Format	Resolution/ Unit	Description		
Transmit Energy	16 bits	Unsigned				
Velocity Scaling	8 bits	Signed		Used to scale velocity data.		
Power level	8 bits	Signed	dB	Configured power level		
Magnetometer temperature	16 bits	Signed	Uncalibrated	Magnetometer temperature reading		
Real Time Clock Temperature	16 bits	Signed	0.01 deg C	Real Time Clock temperature reading		
Error	16 bits			See error description (version 1)		
Status0	16 bits			Bit 0	ProcIdle3 - Indicates that the processor Idles less than 3 percent	
				Bit 1	ProcIdle6 - Indicates that the processor Idles less than 6 percent	
				Bit 2	ProcIdle12 - Indicates that the processor Idles less than 12 percent	
				Bit 3-14	_Unused	
				Bit 15	Status0 in use. If this bit is set the rest of the word should be interpreted	
Status	32 bits			Bit 31-28	Wakeup state	10=break, 11=RTC alarm, 00=bad power, 01=power applied
				Bit 27-25	Orientation	See table 1
				Bit 24-22	Autoorientation	See table 2
				Bit 21-18	Previous wakeup state	10=break, 11=RTC alarm, 00=bad power, 01=power applied
				Bit 17	Last measurement low voltage skip	0=normal operation, 1=last measurement skipped due to low input voltage
				Bit 16	Active configuration	0=Settings for PLAN,BURST, AVG, 1=Settings for PLAN1, BURST1,AVG1
				Bit 15-12	Echo Sounder Index	

Field	Size	Format	Resolution/ Unit	Description		
				Bit 11	Telemetry data	
				Bit 10	Boost Running	
				Bit 9-5	Echo Sounder Frequency Bin	
				Bit 4	_Unused	
				Bit 3	_Unused	
				Bit 2	_Unused	
				Bit 1	bdScaling, Set bit indicates cm scaling of blanking distance	
				Bit 0	_Unused	
Ensemble counter	32 bits	Unsigned		Counts the number of ensembles in both averaged and burst data		
Velocity data	NB*NC* 16 bits	Signed	10^(Velocity Scaling) m/s	This field exists if the Velocity data included bit of the Config byte is set.		
Amplitude data	NB*NC* 8 bits	Unsigned	1 Count = 0.5 dB	This field exists if the Amplitude data included bit of the Config byte is set.		
Correlation data	NB*NC* 8 bits	Unsigned	[0 – 100 %]	This field exists if the Correlation data included bit of the Config byte is set.		
Altimeter distance	32 bits	Float	Meters			These fields exists if the Altimeter data included bit if the config byte is set
Altimeter quality	16 bits	Unsigned				
Altimeter status	16 bits			Bit 0	Pitch or roll > 5 deg	
				Bit 1	Pitch or roll > 10 deg	
AST distance	32 bits	Float	Meters			These fields exists if the AST data included bit if the config

Field	Size	Format	Resolution/ Unit	Description	
AST quality	16 bits	Unsigned			byte is set
AST_offset_100us	16 bits	Signed	100 us	Offset in step of 100 us from AST measurement to velocity measurement	
AST pressure	32 bits	Float	dbar	Pressure value measured during the AST/altimeter ping	
Altimeter spare	8*8 bits			Spare Values	
Altimeter Raw Data – Number of Samples	32 bits	Unsigned		Altimeter Raw Data – Number of Samples	These fields exist if the Altimeter Raw Data is set.
Altimeter Raw Data – Sample distance	16 bits	Unsigned	0.1 mm	Distance between samples	
Altimeter Raw Data – Samples	16 bits	Signed fract		Altimeter Raw Data – Samples	
Echo Sounder data	NC*16 bit	Unsigned		Echo Sounder Amplitude Data	These fields exist if the Echo Sounder data included bit of the Config byte is set.
AHRS Rotation Matrix – M11	32 bits	Float		AHRS Rotation Matrix [3x3]	These fields exist if the AHRS data included bit of the Config byte is set.
AHRS Rotation Matrix – M12	32 bits	Float		AHRS Rotation Matrix [3x3]	
AHRS Rotation Matrix – M13	32 bits	Float		AHRS Rotation Matrix [3x3]	
AHRS Rotation Matrix – M21	32 bits	Float		AHRS Rotation Matrix [3x3]	
AHRS Rotation Matrix – M22	32 bits	Float		AHRS Rotation Matrix [3x3]	
AHRS Rotation Matrix – M23	32 bits	Float		AHRS Rotation Matrix [3x3]	

Field	Size	Format	Resolution/ Unit	Description		
AHRS Rotation Matrix – M31	32 bits	Float		AHRS Rotation Matrix [3x3]		
AHRS Rotation Matrix – M32	32 bits	Float		AHRS Rotation Matrix [3x3]		
AHRS Rotation Matrix – M33	32 bits	Float		AHRS Rotation Matrix [3x3]		
AHRS Dummy	4 * 32 bits					
AHRS Gyro X	32 bits	Float	[dps] – Degrees pr Second	AHRS Gyro		
AHRS Gyro Y	32 bits	Float	[dps] – Degrees pr Second	AHRS Gyro		
AHRS Gyro Z	32 bits	Float	[dps] – Degrees pr Second	AHRS Gyro		
Percent good data	NC * 8 bits	Unsigned	Percent	Percent Good Estimate per cell		These fields exist if the Percentage Good data included bit of the Config byte is set
Std. Dev. Pitch	16 bits	Signed	0.01 Degrees			These fields exist if the Std. Dev. data included bit of the Config byte is set
Std. Dev. Roll	16 bits	Signed	0.01 Degrees			
Std. Dev. Heading	16 bits	Signed	0.01 Degrees			
Std. Dev. Pressure	16 bits	Signed	0.001 Bar			
Std. Dev. Dummy	12 * 16 bits			_Unused		

Value	Instrument Vertical Definition	Description
0	“XUP”	Instrument x-axis defined up, heading reference axis is Z positive
1	“XDOWN”	Instrument x-axis defined down, heading reference axis is Z positive
4	“ZUP”	Instrument z-axis defined up, heading reference axis is X positive
5	“ZDOWN”	Instrument z-axis defined down, heading reference axis is X positive

Table 1 Orientation Description

Value		Description
0		Fixed orientation
1		Auto Up Down

Table 2 Automatic Orientation Detection Status

Error Description

Bit	Field	Description
Bit 15	Tag Error Beam 3 (Quadrature-phase)	1 = Error / 0 = OK
Bit 14	Tag Error Beam 3 (In-phase)	1 = Error / 0 = OK
Bit 13	Tag Error Beam 2 (Quadrature-phase)	1 = Error / 0 = OK
Bit 12	Tag Error Beam 2 (In-phase)	1 = Error / 0 = OK
Bit 11	Tag Error Beam 1 (Quadrature-phase)	1 = Error / 0 = OK
Bit 10	Tag Error Beam 1 (In-phase)	1 = Error / 0 = OK
Bit 9	Tag Error Beam 0 (Quadrature-phase)	1 = Error / 0 = OK
Bit 8	Tag Error Beam 0 (In-phase)	1 = Error / 0 = OK
Bit 7	Not Used	
Bit 6	Not Used	
Bit 5	Sensor read failure	1 = Error / 0 = OK.
Bit 4	Measurement not finished	1 = Error / 0 = OK. The Measurement and data storage/transmit didn’t finish before next measurement started.
Bit 3	Data retrieval Samples missing.	1 = Error / 0 = OK.
Bit 2	Data retrieval Underrun.	1 = Error / 0 = OK.
Bit 1	Data retrieval Overflow.	1 = Error / 0 = OK.
Bit 0	Data retrieval FIFO error.	1 = Error / 0 = OK.

Status Description

Bit	Field	Description
Bit 15-8	Not used	
Bit 7-6	Power level	00 = 0 (high) 01 = 1 10 = 2 11 = 3 (low)
Bit 5-4	Wakeup State	00 = bad power 01 = power applied 10 = break 11 = RTC alarm
Bit 3-0	Not used	

Version 3 VelocityData Record Struct Definition (C99 standard)

```
typedef struct
{
   unsigned short beamData1      : 4;
   unsigned short beamData2      : 4;
   unsigned short beamData3      : 4;
   unsigned short beamData4      : 4;
} t_DataSetDescription4Bit;

typedef struct
{
   unsigned long _empty1         : 1;
   unsigned long bdScaling       : 1;
   unsigned long _empty2         : 1;
   unsigned long _empty3         : 1;
   unsigned long _empty4         : 1;
   unsigned long echoFreqBin     : 5;
   unsigned long boostRunning    : 1;
   unsigned long telemetryData   : 1;
   unsigned long echoIndex       : 4;
   unsigned long activeConfiguration    : 1;
   unsigned long lastMeasLowVoltageSkip : 1;
   unsigned long prevWakeUpState : 4;
   unsigned long autoOrient      : 3;
   unsigned long orientation     : 3;
   unsigned long wakeupstate     : 4;
} t_status;
```

```
typedef struct
{
   unsigned short procIdle3      : 1;
   unsigned short procIdle6      : 1;
   unsigned short procIdle12     : 1;
   unsigned short _empty1        : 12;
   unsigned short stat0inUse     : 1;
} t_status0;


#define VERSION_DATA_STRUCT_3   3

/* Data field */
typedef struct
{
   unsigned char  version; // 3
   unsigned char  offsetOfData; // offsetof(BurstData3_t, data)
   struct  {
      unsigned short pressure        : 1; // 0
      unsigned short temp            : 1; // 1
      unsigned short compass         : 1; // 2
      unsigned short tilt            : 1; // 3
      unsigned short _empty          : 1; // 4
      unsigned short velIncluded     : 1; // 5
      unsigned short ampIncluded     : 1; // 6
      unsigned short corrIncluded    : 1; // 7
      unsigned short altiIncluded    : 1; // 8
      unsigned short altiRawIncluded : 1; // 9
      unsigned short ASTIncluded     : 1; // 10
      unsigned short echoIncluded    : 1; // 11
      unsigned short ahrsIncluded    : 1; // 12
      unsigned short PGoodIncluded   : 1; // 13
      unsigned short stdDevIncluded  : 1; // 14
      unsigned short _unused         : 1;
   } headconfig;
   unsigned long  serialNumber;
   unsigned char  year;
   unsigned char  month;
   unsigned char  day;
   unsigned char  hour;
   unsigned char  minute;
   unsigned char  seconds;
   unsigned short microSeconds100;
   unsigned short soundSpeed;     /* resolution: 0.1 m/s */
   short          temperature;    /* resolution: 0.01 degree
```

```
Celsius */
   unsigned long  pressure;
   unsigned short heading;
   short          pitch;
   short          roll;
union {
      unsigned short beams_cy_cells; ///< bit 15-12: Number of
beams,
                                     ///< bit 11-10: coordinate
system,
                                     ///< bit 9-0: Number of cells.
      unsigned short echo_cells;     ///< OR, Number of echo
sounder cells.
};
unsigned short cellSize;
   unsigned short blanking;
   unsigned char  nominalCorrelation;
   unsigned char  pressTemp;
   unsigned short battery;
   short          magnHxHyHz[3];     ///< Magnetometer Min data
   short          accl3D[3];         ///< Accelrometer Data
   union {
     unsigned short ambVelocity;
     unsigned short echoFrequency;
   };
   t_DataSetDescription4Bit DataSetDescription4bit; /* unsigned
short */
   unsigned short transmitEnergy;
   char           velocityScaling;
   char           powerlevel;
   short          magnTemperature;
   short          rtcTemperature;
   unsigned short error;
   t_status0      status0;       /* Unsigned short */
   t_status       status;        /* Unsigned long  */
   unsigned long  ensembleCounter;
   unsigned char data[SIZE_VAR_DATA];
   ///< actual size of the following =
   ///<   int16_t hVel[nBeams][nCells];    // velocity
   ///<   uint8_t  cAmp[nBeams][nCells];   // amplitude
   ///<   uint8_t  cCorr[nBeams][nCells];  // correlation  (0-100)
} OutputData3_t;


/* Altimeter result */
```

```
typedef struct _altiData_t
{
   float          fDistanceLE;              ///< Distance Leading
Edge [m].
   unsigned short qualityLE;                ///< Quality parameter
Leading Edge
   struct
   {
      unsigned short pitchRoll5deg     : 1; ///< Pitch or Roll more
> 5 deg.
      unsigned short pitchRoll10deg    : 1; ///< Pitch or Roll more
> 10 deg.
      unsigned short multiBeamIncluded : 1; ///< Multi beam data
included
      unsigned short nBeams            : 4; ///< Number of
altimeter beams
      unsigned short powIndex          : 3; ///< Power level index
for current ping
      unsigned short _unused           : 6;
   } status;                                ///< Status.
   float          fDistanceAST;             ///< Distance Leading
AST [m].
   unsigned short qualityAST;               ///< Quality parameter
Leading AST.
   short          offsetAST_100us;          ///< Time from pressure
measurement.
   float          fAltiPressure;            ///< Altimeter Pressure
[dBar]
   float          fPowLev;                  ///< Power level for
current ping
   float          fSpare;                   ///< Spare
} altiData_t;
```

6.1.3 Bottom Track Data Record Definition (DF20)

Field	Size	Format	Resolution/ Unit	Description	
Version	8 bits			Version number of the Data Record Definition. (3)	
offsetOfData	8 bits	Unsigned	#Bytes	Number of bytes from start of record to start of data (velocity/amplitude/correlation)	
Configuration	16 bits			Record Configuration Bit Mask	
				Bit 0	Pressure sensor value valid.
				Bit 1	Temperature sensor value valid.
				Bit 2	Compass sensor values valid.
				Bit 3	Tilt sensor values valid.
				Bit 4	-
				Bit 5	Velocity data included
				Bit 6	-
				Bit 7	-
				Bit 8	Distance data included
				Bit 9	Figure of Merit data included
				Bit 10-15	Unused
Serial Number	32 bits	Unsigned			
Year	8 bits	Unsigned	1 Year	Years since 1900 (see struct tm definition)	
Month	8 bits	Unsigned	1 Month	Jan =0, Feb= 1, etc.(see struct tm definition)	
Day	8 bits	Unsigned	1 Day	(see struct tm definition)	
Hour	8 bits	Unsigned	1 Hour	(see struct tm definition)	
Minute	8 bits	Unsigned	1 Minute	(see struct tm definition)	
Seconds	8 bits	Unsigned	1 Second	(see struct tm definition)	
Microsec100	16 bits	Unsigned	100 µsec		
Speed of Sound	16 bits	Unsigned	0.1 m/s		
Temperature	16 bits	Signed	0.01 Degree Celsius		
Pressure	32 bits	Unsigned	0.001 dBar		
Heading	16 bits	Unsigned	0.01 Deg		
Pitch	16 bits	Signed	0.01 Deg		
Roll	16 bits	Signed	0.01 Deg		
#Beams & Coordinate system &	16 bits			Definition:	

Field	Size	Format	Resolution/ Unit	Description	
#Cells				Bit 9 - 0	
				Bit 11 - 10	Coordinate system, b00 : ENU b01 : XYZ b10 : BEAM b11 : -
				Bit 15 – 12	Number of Beams (NB)
Cell Size	16 bits	Unsigned	1 mm		
Blanking	16 bits	Unsigned	1 mm		
Nominal Correlation	8 bits	Unsigned	%	The nominal correlation for the configured combination of cell size and velocity range.	
Unused value	8 bits				
Battery Voltage	16 bits	Unsigned	0.1 Volt		
Magnetometer Raw(X-axis)	16 bits	Signed		Magnetometer Raw, X axis value in last measurement interval.	
Magnetometer Raw(Y-axis)	16 bits	Signed		Magnetometer Raw, Y axis value in last measurement interval.	
Magnetometer Raw(Z-axis)	16 bits	Signed		Magnetometer Raw, Z axis value in last measurement interval.	
Accelerometer Raw (X-axis)	16 bits	Signed		Accelerometer Raw X axis value in last measurement interval. (16384 = 1.0)	
Accelerometer Raw (Y-axis)	16 bits	Signed		Accelerometer Raw Y axis value in last measurement interval. (16384 = 1.0)	
Accelerometer Raw (Z-axis)	16 bits	Signed		Accelerometer Raw Z axis value in last measurement interval. (16384 = 1.0)	
Ambiguity Velocity	32 bits	Unsigned	10^(Velocity Scaling) m/s	Ambiguity velocity, corrected for sound velocity, scaled according to Velocity Scaling	
Data Set Description	16 bits			**Bits**	**Description**
				0-3	Physical beam used for 1st data set.
				4-7	Physical beam used for 2nd data set.
				8-11	Physical beam used for 3th data set.
				12-16	Physical beam used for 4th data set.
Transmit Energy	16 bits	Unsigned			
Velocity Scaling	8 bits	Signed		Used to scale velocity data.	
Power level	8 bits	Signed	dB	Configured power level	
Magnetometer Temperature	16 bits	Signed	Uncalibrated	Magnetometer temperature reading	

Field	Size	Format	Resolution/ Unit	Description		
Real Time Clock Temperature	16 bits	Signed	0.01 Degree Celsius	Real time clock temperature reading		
Error	32 bits			See Error Description (version 1)		
Status	32 bits			Bit 31-28	Wakeup State	10 = break 11 = RTC alarm 00 = bad power 01 = power applied
				Bit 27-25	Orientation	See Table 1.
				Bit 24-22	Auto orientation	See Table 2.
				Bit 21	Active Configuration	0 = BT Settings 1 = BT1 Settings
Ensemble counter	32 bits	Unsigned		Counts the number of ensembles in both averaged data and burst data		
Velocity data	NB* 32 bits	Signed	10^(Velocity Scaling) m/s	This field exists if the *Velocity data included* bit of the *Config* byte is set.		
Distance data	NB* 32 bits	Signed	mm	This field exists if the Distance *data included* bit of the *Config* byte is set.		
Figure Of Merit data	NB * 16 bits	Unsigned		This field exists if the *FOM data included* bit of the *Config* byte is set.		

Version 1 Bottom Track Data Record Struct Definition (C99 standard)

```
typedef struct
{
   unsigned char  version;
   unsigned char  offsetOfData;
   struct  {
      unsigned short pressure     : 1; // 0
      unsigned short temp         : 1; // 1
      unsigned short compass      : 1; // 2
      unsigned short tilt         : 1; // 3
      unsigned short _empty       : 1; // 4
      unsigned short velIncluded  : 1; // 5
      unsigned short _unused1     : 1; // 6
      unsigned short _unused2     : 1; // 7
      unsigned short distIncluded : 1; // 8
      unsigned short fomIncluded  : 1; // 9
      unsigned short _unused3     : 6;
   } headconfig;
```

```
   unsigned long  serialNumber;
   unsigned char  year;
   unsigned char  month;
   unsigned char  day;
   unsigned char  hour;
   unsigned char  minute;
   unsigned char  seconds;
   unsigned short microSeconds100;
   unsigned short soundSpeed;        ///< resolution: 0.1 m/s
   short          temperature;       ///< resolution: 0.01 degre
Celsius
   unsigned long  pressure;
   unsigned short heading;
   short          pitch;
   short          roll;
   unsigned short beams_cy;      ///< bit 15-12: Number of beams,
bit 11-10: coordinate system
   unsigned short cellSize;
   unsigned short blanking;
   unsigned short velocityRange;
   unsigned short battery;
   short          magnHxHyHz[3]; ///< Magnetometer Data
   short          accl3D[3];     ///< Accelerometer Data
   unsigned int   ambVelocity;
   t_BottomTrackDataSetDescription4Bit DataSetDescription4bit; /*
unsigned short */
   unsigned short transmitEnergy;
   char           velocityScaling;
   char           powerlevel;
   short          magnTemperature;
   short          rtcTemperature;
   unsigned long  error;
   t_BottomTrackstatus status;        /* Unsigned long */
   unsigned long  ensembleCounter;
   unsigned char data[SIZE_VAR_DATA_BT];
   ///< actual size of the following:
   ///<   int32_t        velocity[nBeams];  // velocity
   ///<   int32_t        distance[nBeams];  // distance
   ///<   unsigned short FOM[nBeams];       // Figure Of Merit
} OutputBottomTrackFormat1_t;
```

6.2 String Data Record Definition

The String Data Record is written to the SD memory card using the FWRITE command. The string data record is also used to store the instrument configuration. The ID parameter is then set to 16 (0x10).

Field	Size	Description
ID	8 bits	The ID parameter (0-15) of the FWRITE command.
String	Variable	The STR parameter of the FWRITE command. The string is zero terminated.

6.3 Data Limit Formats

The limits for the various arguments are returned as a list of valid values, and/or ranges, enclosed in parenthesis **()**. An empty list, **()**, is used for arguments that are unused/not yet implemented. Square brackets **[]** signify a range of valid values that includes the listed values. String arguments are encapsulated with **“”**, like for normal parameter handling. A semicolon, **;**, is used as separator between limits and values.

The argument format can also be inferred from the limits, integer values are shown without a decimal point, floating point values are shown with a decimal point and strings are either shown with the string specifier, **“”**, or as a range of characters using **‘** for specifying a character.

Examples:

[1;128] – Integer value, valid from 1 to 128

([1300.00;1700.00];0.0) – Floating point value, valid values are 0.0 and the range from 1300.00 to 1700.00.

(['0';'9'];['a';'z'];['A';'Z'];'.') – String argument with valid characters being . and the character ranges a-z, A-Z, 0-9 .

("BEAM") – String argument with BEAM being the only valid string.

(0;1) – Integer value with two valid values, 0 and 1.

NMEA interface example:

```
$PNOR,GETAVGLIM*22
$PNOR,GETAVGLIM,NC=([1;128]),CS=([0.25;2.00]),BD=([0.10;45.00]),CY=
("BEAM"),PL=([-40.0;0.0];-100.0),AI=([1;300]),VP=([0.000;0.100]),
VR=([1.25;5.00]),DF=([0;3]),NPING=([1;4])*46
$PNOR,OK*2B
```

Regular interface example:

```
GETPLANLIM
([1;3600]),(0;1),(),([0;2]),(),([0.0;50.0]),(0;1),([10;21600]),(),
([1300.00;1700.00];0.0),(['0';'9'];['a';'z'];['A';'Z'];'.'),(0;1)
OK
```

7 Telemetry Data Formats

This section describes the Telemetry Data formats.

7.1 Averaging Mode

The telemetry if the AVG mode is controlled by the SET/GETTMAVG command. The DF parameter of this command sets the data format.

Data format (DF)	Description
3	Binary format as described in 'Data Record Definition (version 3)'.
100	Same format as AWAC NMEA format. (NMEA sentences: PNORI, PNORS and PNORC).
101	NMEA format 1 (without Tags). (NMEA sentences: PNORI1, PNORS1 and PNORC1).
102	NMEA format 2 (with Tags). (NMEA sentences: PNORI2, PNORS2 and PNORC2).
103	NMEA format 3 (with Tags). (NMEA sentences: PNORH3, PNORS3 and PNORC3).
104	NMEA format 4 (without Tags). (NMEA sentences: PNORH4, PNORS4 and PNORC4).
150	RDI Workhorse PD0 data format.

Table 3 Available Telemetry Data formats for AVG.

7.1.1 AWAC NMEA Format (DF=100)

Data with variants of -9 (-9.00, -999…) are invalid data.
Empty files are fields not used.
The checksum calculation is part of the NMEA standard. It is the representation of two hexadecimal characters of an XOR if all characters in the sentence between – but not including – the $ and the * character.

Information (configuration) $PNORI

Column	Description	Data format	Example
0	Identifier	"$PNORI"	
1	Instrument type	N, 4=Signature	4
2	Head ID	Signaturexxx xNNNNNN	Signature100 0900002
3	Number of beams	N	4
4	Number of cells	dd.dd	20
5	Blanking (m)	dd.dd	0.20
6	Cell size (m)	dd.dd	1.00
7	Coordinate system	ENU=0, XYZ=1, BEAM=2	0
8	Checksum	*hh	2E

Example (DF=100): `$PNORI,4,Signature1000900002,4,5,0.20,1.00,0*2E`

Sensor data $PNORS

Column	Description	Data format	Example
0	Identifier	"$PNORS"	
1	Date	MMDDYY	102115
2	Time	hhmmss	090715
3	Error Code (hex)	hh	00000000
4	Status Code (hex)	hh	2A480000
5	Battery Voltage	dd.d	14.4
6	Sound Speed	dddd.d	1523.0
7	Heading	ddd.d	275.9
8	Pitch (deg)	dd.d	15.7
9	Roll (deg)	dd.d	-2.3
10	Pressure (dBar)	ddd.ddd	0.000

11	Temperature (deg C)	dd.dd	22.45
12	Analog input #1	Not in use	0
13	Analog input #2	Not in use	0
14	Checksum	*hh	1C

Example (DF=100):
`$PNORS,102115,090715,00000000,2A480000,14.4,1523.0,275.9,15.7,2.3,0.000,22.45,0,0*1C`

Current velocity data $PNORC

Column	Description	Data format	Example
0	Identifier	"$PNORC"	
1	Date	MMDDYY	102115
2	Time	hhmmss	090715
3	Cell number	N	4
4	Velocity 1 (m/s) (Beam1/X/East)	dd.dd	0.56
5	Velocity 2 (m/s) (Beam2/Y/North)	dd.dd	-0.80
6	Velocity 3 (m/s) (Beam3/Z1/Up1)	dd.dd	-1.99
7	Velocity 4 (m/s) (Beam4/Z2/Up2)	dd.dd	-1.33
8	Speed (m/s)	dd.dd	0.98
9	Direction (deg)	ddd.d	305.2
10	Amplitude unit	C= counts	C
11	Amplitude (Beam 1)	dd	80
12	Amplitude (Beam 2)	dd	88
13	Amplitude (Beam 3)	dd	67
14	Amplitude (Beam 4)	dd	78
15	Correlation (%) (Beam1)	dd	13
16	Correlation (%) (Beam2)	dd	17
17	Correlation (%) (Beam3)	dd	10
18	Correlation (%) (Beam4)	dd	18
19	Checksum	*hh	22

Example (DF=100): `$PNORC,102115,090715,4,0.56,-0.80,-1.99,-1.33,0.98,305.2,C,80,88,67,78,13,17,10,18*22`

Note that the amplitude can be converted to a dB scale by multiplying by 0.50 dB/count.

7.1.2 NMEA Format 1 and 2 (DF=101/102)

Information Data:
Identifier:
PNORI1 for DF = 101
PNORI2 for DF = 102

Column	Field	Description	TAG	Data format	Example
1	Instrument type	4 = Signature	IT	N	IT=4
2	Head ID		SN	N	SN=123456
3	Number of Beams		NB	N	NB=3
4	Number of Cells		NC	N	NC=30
5	Blanking Distance	[m]	BD	dd.dd	BD=1.00
6	Cell Size	[m]	CS	dd.dd	CS=5.00
7	Coordinate System		CY	ENU,BEAM, XYZ	CY=BEAM

Table 4 PNORI1/2 NMEA sentence parameter description

Example (DF=101): `$PNORI1,4,123456,3,30,1.00,5.00,BEAM*5B`
Example (DF=102): `$PNORI2,IT=4,SN=123456,NB=3,NC=30,BD=1.00,CS=5.00,CY=BEAM*68`

Sensors Data:
Identifier:
PNORS1 for DF = 101
PNORS2 for DF = 102

Column	Field	Description	TAG	Data format	Example
1	Date		DATE	MMDDYY	DATE=083013
2	Time		TIME	hhmmss	TIME=132455
3	Error Code		EC	N	EC=0
4	Status Code		SC	hhhhhhhh	SC=34000034
5	Battery Voltage	[V]	BV	dd.d	BV=23.9
6	Sound Speed	[m/s]	SS	dddd.d	SS=1500.0
7	Heading Std. Dev.	[deg]	HSD	dd.dd	HSD=0.02
8	Heading	[deg]	H	ddd.d	H=123.4
9	Pitch	[deg]	PI	dd.d	PI=45.6
10	Pitch Std.Dev	[deg]	PIS	dd.dd	PISD=0.02

			D		
11	Roll	[deg]	R	dd.d	R=23.4
12	Roll Std.Dev.	[deg]	RSD	dd.dd	RSD=0.02
13	Pressure	[dBar]	P	ddd.ddd	P=123.456
14	Pressure StdDev	[dBar]	PSD	dd.dd	PSD=0.02
15	Temperature	[deg C]	T	dd.dd	T=24.56

Table 5 PNORS1/2 NMEA sentence parameter description

Example (DF=101):
$PNORS1,083013,132455,0,34000034,23.9,1500.0,0.02,123.4,45.6,0.02,
R=23.4,0.02,123.456,0.02,24.56*39

Example (DF=102):
$PNORS2,DATE=083013,TIME=132455,EC=0,SC=34000034,BV=23.9,SS=1500.0,HSD=0.02,H=123.4,
PI=45.6,PISD=0.02,R=23.4,RSD=0.02,P=123.456,PSD=0.02,T=24.56*3F

Averaged Data:
Identifier:
PNORC1 for DF = 101
PNORC2 for DF = 102

The averaged data is repeated for each measurement cell.

Column	Field	Description	TAG	Data format	Example
1	Date	Date	DATE	MMDDYY	DATE=083013
2	Time	Time	TIME	hhmmss	TIME=132455
3	Cell Number	#	CN	dd	CN=3
4	Cell Position	[m]	CP	dd.d	CP=11.0
5	Velocity East	[m/s] Only if CY=ENU	VE	dd.ddd	VE=0.332
6	Velocity North	[m/s] Only if CY=ENU	VN	dd.ddd	VN=0.332
7	Velocity Up	[m/s] Only if CY=ENU and #beams >= 3	VU	dd.ddd	VU=0.332
8	Velocity Up2	[m/s] Only if CY=ENU and #beams = 4	VU2	dd.ddd	VU2=0.332
9	Velocity X	[m/s] Only if CY=XYZ	VX	dd.ddd	VX=0.332
10	Velocity Y	[m/s] Only if CY=XYZ	VY	dd.ddd	VY=0.332
11	Velocity Z	[m/s] Only if CY=XYZ and #beams >= 3	VZ	dd.ddd	VZ=0.332
12	Velocity Z2	[m/s] Only if CY=XYZ and #beams = 4	VZ2	dd.ddd	VZ2=0.332

Column	Field	Description	TAG	Data format	Example
13	Velocity Beam 1	[m/s] Only if CY=BEAM	V1	dd.ddd	V1=0.332
14	Velocity Beam 2	[m/s] Only if CY=BEAM and #beams >=2	V2	dd.ddd	V2=0.332
15	Velocity Beam 3	[m/s] Only if CY=BEAM and #beams >=3	V3	dd.ddd	V3=-0.332
16	Velocity Beam 4	[m/s] Only if CY=BEAM and #beams =4	V4	dd.ddd	V4=-0.332
17	Amplitude Beam 1	[dB]	A1	ddd.d	A1=78.9
18	Amplitude Beam 2	[dB] Only if #beams >=2	A2	ddd.d	A2=78.9
19	Amplitude Beam 3	[dB] Only if #beams >=3	A3	ddd.d	A3=78.9
20	Amplitude Beam 4	[dB] Only if #beams =4	A4	ddd.d	A4=78.9
21	Correlation Beam 1	[%]	C1	dd	C1=78
22	Correlation Beam 2	[%] Only if #beams >=2	C2	dd	C2=78
23	Correlation Beam 3	[%] Only if #beams >=3	C3	dd	C3=78
24	Correlation Beam 4	[%] Only if #beams =4	C4	dd	C4=78

Table 6 PNORC1/2 NMEA sentence parameter description

Example (DF=101 (ENU, 3 beams):

```
$PNORC1,083013,132455,3,11.0,0.332,0.332,0.332,78.9,78.9,78.9,78,78
,78*46
```

Example (DF=102 (ENU, 3 beams):

```
$PNORC2,DATE=083013,TIME=132455,CN=3,CP=11.0,VE=0.332,VN=0.332,
VU=0.332,A1=78.9,A2=78.9,A3=78.9,C1=78,C2=78,C3=78*6D
```

Example (DF=102 (BEAM, 4 beams):

```
$PNORC2,DATE=083013,TIME=132455,CN=3,CP=11.0,V1=0.332,V2=0.332,V3=-
0.332,V4=-0.332,A1=78.9,A2=78.9,A3=78.9,A4=78.9,C1=78,C2=78,C3=78,
C4=78*49
```

7.1.3 NMEA Format 3 and 4 (DF=103/104)

Header Data:
Identifier:
PNORH3 for DF = 103
PNORH4 for DF = 104

Column	Field	Description	TAG	Data format	Example
1	Date	Date	DATE	YYMMDD	DATE=141112
2	Time	Time	TIME	hhmmss	TIME=081946
3	Error Code	See DF3 (3.2.3)	EC	N	EC=0
4	Status Code	See DF3 (3.2.3)	SC	HHHHHHHH	SC=2A4C0000

Table 7 PNORH3/4 NMEA Header sentence parameter description

Example (DF=103): `$PNORH3,DATE=141112,TIME=081946,EC=0,SC=2A4C0000*5F`
Example (DF=104): `$PNORH4,141112,083149,0,2A4C0000*4A68`

Sensors Data:
Identifier:
PNORS3 for DF = 103
PNORS4 for DF = 104

Column	Field	Description	TAG	Data format	Example
1	Battery	[V]	BV	dd.d	BV=23.9
2	Sound Speed	[m/s]	SS	dddd.d	SS=1500.0
3	Heading	[deg]	H	ddd.d	H=123.4
4	Pitch	[deg]	PI	dd.d	PI=45.6
5	Roll	[deg]	R	dd.d	R=23.4
6	Pressure	[dBar]	P	ddd.ddd	P=123.456
7	Temperature	[deg C]	T	dd.dd	T=24.56

Table 8 PNORS3/4 NMEA Sensor sentence parameter description

Example (DF=103):
`$PNORS3,BV=33.0,SS=1546.1,H=151.1,PI=-12.0,R=-5.2,P=705.669,T=24.96*7A`

Example (DF=104):
`$PNORS4,33.0,1546.1,151.2,-11.9,-5.3,705.658,24.95*5A`

Averaged Data:
Identifier:
PNORC3 for DF = 103
PNORC4 for DF = 104

The averaged data is repeated for each measurement cell.

Column	Field	Description	TAG	Data format	Example
1	Cell position	[meter]	CP	D.D	CP=2.5
2	Speed	[m/s]	SP	d.ddd	SP=0.751
3	Direction	[deg]	DIR	ddd.d	DIR=170.1
4	Averaged Correlation		AC	N	AC=5
5	Averaged Amplitude		AA	N	AA=28

Table 9 PNORC3/4 NMEA Averaged data sentence parameter description

Example (DF=103):

```
$PNORC3,CP=4.5,SP=3.519,DIR=110.9,AC=6,AA=28*3B
```

Example (DF=104):

```
$PNORC4,27.5,1.815,322.6,4,28*70
```

7.1.4 RDI Workhorse PD0 data format.

See RDI documentation.

7.2 Burst

The data format for the Burst telemetry is given by the BURST,DF parameter.

7.3 Altimeter

The telemetry for the Altimeter is controlled by the SET/GETTMALTI command. The DF parameter of this command sets the data format.

Data format (DF)	Description
200	NMEA (PNORA) format without Tags.
201	NMEA (PNORA) format with Tags.

Table 10 Available Data formats for Altimeter.

Column	Field	Description	TAG	Data format	Example
1	Date	Date	DATE	YYMMDD	DATE=130830
2	Time	Time	TIME	hhmmss	TIME=132455
3	Pressure	[dBar]	P	ddd.ddd	P=123.456
4	Altimeter Distance	[m]	A	ddd.ddd	A=112.233
5	Quality Parameter		Q	N	Q=78
6	Status	Status	ST	XX	ST=00

Table 11 PNORA NMEA sentence parameter description.

Example (DF=200): `$PNORA,130920,134824,37.604,125.583,42,=00*46`
Example (DF=201): `$PNORA,DATE=130920,TIME=134824,P=37.604,A=125.583,Q=42,ST=00*3D`

7.4 DVL Bottom Track

The telemetry for the Bottom track is controlled by the SET/GETTMBT command. The DF parameter of this command sets the data format.

Data format (DF)	Description
300	NMEA (PNORBT) format without Tags.
301	NMEA (PNORBT) format with Tags.

Table 12 Available Data formats for Bottom track.

Column	Field/TAG	Description	Data format	Example
1	BEAM	Beam number	n	BEAM=3
2	DATE	Date	MMDDYY	DATE=112813
3	TIME	Time	hhmmss.ssss	TIME=072228.2345
4	DT1	Diff. time 1	s.ssss	DT1=0.1234
5	DT2	Diff. time 2	s.ssss	DT2=0.1234
6	BV	Bottom Velocity [m/s]	f.fffff	BV=1.11111
7	FM	Figure of Merit	f.f	FM=122.2
8	DIST	Distance [meter]	f.ff	DIST=36.66
9	WV	Water Velocity [m/s]	f.fffff	WV=2.22222
10	STAT	Status	hh	STAT=F7

Table 13 PNORBT NMEA sentence parameter description.

Example (DF=300):

```
$PNORBT,3,112813,072228.2345,0.1234,0.1234,1.11111,122.2,36.66,2.22222,F7*7A
```

Example (DF=301):

```
$PNORBT,BEAM=3,DATE=112813,TIME=072228.2345,DT1=0.1234,DT2=0.1234,BV=1.11111,FM=122.2,DIST=36.66,WV=2.22222,STAT=F7*75
```

7.5 ASCII Data Input Using Ethernet

```
/* Sample code showing how to connect to and receive data from the Nortek Signature series
 * of instruments using the ASCII only data port.
 * Compiles on both Windows (requires ws2_32 library) and Linux.
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#ifdef __WIN32__
#include <winsock2.h>
#else
#include <sys/socket.h>
#include <sys/types.h>

#include <sys/time.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#endif

#include <sys/time.h>
#include <unistd.h>
#include <stdarg.h>

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

static int socket_fd = -1;

#define ASCII_DATA_PORT    9004

char dataBuffer[4096];

int main(void) {
    struct sockaddr_in server;
    struct hostent *hp;
    char *ip_address = "192.168.20.10";

#ifdef __WIN32__
    WSADATA version;
    WORD mkword = MAKEWORD(2, 2);
    int what = WSAStartup(mkword, &version);
    if (what != 0) {
        printf("Version not supported\n\n");
        exit(-1);
    }
#endif

    /* Create socket */
```

```
    socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        printf("Could not create socket %s\n\n", strerror(errno));
        exit(-1);
    }

    memset((char *) &server, 0, sizeof(server));

    /* Connect socket using name specified name / IP address.  */
    server.sin_family = AF_INET;
    hp = gethostbyname(ip_address);
    if (hp == 0) {
        printf("Invalid host name\n\n");
        exit(-1);
    }
    memcpy(&server.sin_addr, hp->h_addr, hp->h_length);
    server.sin_port = htons((unsigned short) ASCII_DATA_PORT);

    /* 30 second receive timeout.  The actual timeout to use will depend
     * upon the instrument configuration and other considerations.
     */
#ifdef __WIN32__
    /* On windows, the timeout is number of ms. */
    int ts = 30000;
#else
    /* Other OSes use timeval structure. */
    struct timeval ts;
    ts.tv_sec = 30;
    ts.tv_usec = 0;
#endif

    if (setsockopt(socket_fd, SOL_SOCKET, SO_RCVTIMEO, (void *) &ts, sizeof(ts))
            < 0) {
        printf("Could not set receive timeout\n\n");
        exit(-1);
    }

    /* Connect to the instrument... */
    if (connect(socket_fd, (struct sockaddr *) &server, sizeof(server)) < 0) {
        printf("Could not connect to host %s\n\n", ip_address);
        exit(-1);
    }

    int length = 0;
    while (1) {
        char c;
        int r;
#ifdef __WIN32__
        if ((r = recv(socket_fd, &c, 1, 0)) <= 0) {
            if (r == 0) {
                /* Instrument terminated socket for some reason.  Re-connect required. */
                printf("Instrument terminated socket.\n\n");
            } else {
```

```
            if (WSAGetLastError() == WSAETIMEDOUT) {
                /* No data received within timeout period.  Could either loop or
                 * re-open / check connection at this point.
                 */
                printf("Socket read timed out\n\n");
            } else {
                /* Local socket error.  Re-connect required. */
                wchar_t *s = NULL;
                FormatMessageW(
                        FORMAT_MESSAGE_ALLOCATE_BUFFER
                                | FORMAT_MESSAGE_FROM_SYSTEM
                                | FORMAT_MESSAGE_IGNORE_INSERTS,
                        NULL, WSAGetLastError(),
                        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                        (LPWSTR) &s, 0, NULL);
                printf("Socket read failed %S\n\n", s);
                LocalFree(s);
            }
        }
        break;
    }
#else
    if ((r = read(socket_fd, &c, 1)) <= 0) {
        if (r == 0) {
            /* Instrument terminated socket for some reason.  Re-connect required. */
            printf("Server terminated socket\n\n");
        } else {
            if (errno == EAGAIN) {
                /* No data received within timeout period.  Could either loop or
                 * re-open / check connection at this point.
                 */
                printf("Socket read timed out\n\n");
            } else {
                /* Local socket error.  Re-connect required. */
                printf("Socket read failed (%d) %s\n\n", errno, strerror(errno));
            }
        }

        break;
    }
#endif

    dataBuffer[length++] = c;

    if (length >= sizeof(dataBuffer)) {
        printf("Truncating data input...\n\n");
        length = sizeof(dataBuffer) - 1;
    }

    /* Set last byte to 0 so that strings are zero terminated. */
    dataBuffer[length] = 0;
    if ('\n' == c) {
        /* '\n' indicates end-of-line for ASCII data. */
```

```
            printf("Received: %s", dataBuffer);
            /* Receive next line of data. */
            length = 0;
        }
    }
    close(socket_fd);

    exit (1);
}
```