www.bathyswath.com
support@iter-systems.com

Reference	**ETD-2020**
Classification	**Confidential**
Version	**8.01**
Date	**07/11/17**
Copy number *(if applicable)*	**N/A**

Bathyswath File Formats

This page is left blank intentionally

VOIDS

Void	Section	Notes
10		
9		
8		
7		
6		
5		
4		
3		
2		
1		

LIST OF MODIFICATIONS

Version	Date	Modifications	Pages	Writer	Checker
8.01	07/11/17	Small clarifications added		MFG	
8.00	21/06/17	Reformat and update			
07.09	16/08/16	Section on example code for reading files			
07.08	09/08/16	Note on correction to SXP file reading			
07.07	14/10/15	Corrected block number for SONAR_DATA4			
07.06	19/11/14	Added new raw data block, SONAR_DATA4			
07.05	04/11/14	Added SCP Bathyswath Position Correction File			
07.04	30/10/14	Added geoid and height offset files			
07.03	04/06/14	Documented more interface formats			
07.02	03/09/13	Reserved a set of block IDs for client use			
07.01	22/08/13	Corrected description of ping position in SXP			
07.00	28/02/13	Bathyswath version, derived from "SWATHplus File Formats" rev 6E			

TABLE OF CONTENTS

1 INTRODUCTION

1.1 REFERENCES

Ref 1 First reference …

1.2 GLOSSARY & ACRONYMS

WORDS & ACRONYMS	DEFINITION
Bathyswath	A seabed mapping sonar system. Also the name of the organisation that builds and sells it
Swath	The Bathyswath Swath Processor software application
SWATHplus	The forerunner of the Bathyswath system

1.3 SCOPE

This document describes the format and interpretation of the data files written by the Bathyswath and SWATHplus sonar systems.

There are several file types written by the software, all using the same format. See “File Types” below.

These data files are written onto the PC’s hard disk by the sonar software and contain all the information recorded by the system during the survey.

These data items can also be written or read by the Bathyswath software in real time over communications links, including TCP/IP, UDP and serial ports.

1.4 CONTEXT

Bathyswath is a swath bathymetry sonar system. It is derived from the SWATHplus sonar system, and uses the same file formats. In turn, SWATHplus was derived from the Submetrix sonars, built by Submetrix Ltd.

2 FILE TYPES

The following file types are described in detail in this document. They all use the Bathyswath block-orientated format.

Type	Format	Suffix	Contains	Notes	See Section
Raw sonar data	binary	.sxr	All the data collected by the real-time system	Processed off-line to produce one or more of the processed data file types.	0
Coverage maps	binary	.swc	Coverage data used by the coverage plot view	These files are a summary of the data coverage, but contain no sonar data, so are not commonly read by third-party software.	
Processed sonar data	binary	.sxp	Processed data derived from the real-time software	These files have all corrections applied, including: attitude, position, tide, speed of sound. Includes down-sampled position, attitude and tide information. Processed data files can be created from raw data files or parsed data files.	0
Parsed data	binary	.sxi	Raw data, but parsed into a format that is easier for third-party code to interpret.	These files have none of the above corrections applied. Parsed data files can be created from raw data files, but not the other way around.	0

The following file types are used by the Bathyswath and SWATHplus software, but do not use the block-orientated format and are not easily read by third-party software:

Type	Format	Suffix	Contains
Swath processor session file	MFC “serialization” file	.sxs	The set-up of the Swath Processor program, including the configuration of the sonar, input port configurations, filter settings and view window parameters
Grid file	Binary format, followed by MFC “serialization” data	.sxg	The grid data, in an open binary format, followed by the set-up of the Grid Processor, including filter settings and view window parameters

3 DATA STORAGE AND TRANSMISSION

3.1 DISK FORMAT

The data files are written using a Microsoft Windows operating system and therefore follow the conventions of that system in terms of file naming and low-level disk format.

3.2 FILE NAMES

The file name is specified by the user during the operation of the real-time sonar software. The extension is supplied by the software according to the file type (see the table above).

3.3 ENDIANNESS

All Bathyswath and SWATHplus data is little-endian, i.e. in the natural 80x86 format with the least significant byte at the lower address.

3.4 REAL-TIME TCP/IP DATA TRANSMISSION

Bathyswath software can output data in real time, and be controlled, over a TCP/IP interface (for example, using an Ethernet wired or wireless LAN). This interface uses the same block-orientated structure as the data files.

The Bathyswath Swath Processor application can output the Parsed Data format (section 7.3) and the Raw Data format (section 0) by TCP/IP in real time.

4 BLOCK-ORIENTATED DATA FORMAT

4.1 GENERAL

All of the files listed in the first table of section 0 use the same block-orientated data format. They can be read using the same software code, and the blocks that they contain may be included in any of the files. The difference between these file types is therefore simply the types of data block that they tend to contain.

Each file contains a file header block, followed by a series of data blocks.

Every block contains a header that identifies the block, followed by the length of the block. Therefore, the reading software can identify the blocks that it wishes to read, and ignore and skip over any other block type that it encounters. In this way, new blocks can be added to a file without necessarily having to update the reading software.

4.2 FILE HEADER

A file header is used to identify each file type. It is formatted in the same way as data blocks, but with a “magic number” as the block type. Each file type uses a different magic number. See the table below.

However, the file header may not be present in some circumstances, and the file may start immediately with data blocks.

This magic number appears in the file as the sequence of bytes. The second 32-bit integer is the block length, which is currently set to a value of 8 bytes. The content of the header block is two 32-bit integers representing the software version number and the file format version number.

The software version number is encoded as an integer, as follows: (Major version- Minor version- Release- Build). For example, a version number of 3065601 means: Major version 3, Minor version 06, Release 56, Build 01.

The file format version number is now obsolete: use the data block identifiers as a way of checking file versions. For example, Swath version 3.7 writes "SONAR_DATA3" in its raw data files, and version 3.6 writes "SONAR_DATA2" blocks. To allow an application to read both kinds of file, add parsing code for both block types.

4.2.1 Magic Numbers

The file type magic numbers are:

File type	Magic number identifier	Magic number (hexadecimal)
Raw sonar data	SXR_HEADER_DATA	0xbad0bad0
Configuration data	SXC_HEADER_DATA	0xf1c0f1c0
Coverage maps	SWC_HEADER_DATA	0xc311c311
Processed sonar data	SXP_HEADER_DATA	0x01df01df
Grid data	SXG_HEADER_DATA	0xd1edede0
Parsed data	SXI_HEADER_DATA	0x521d52d1

4.2.2 File Header Data

The data part of the file header block can be read as:

```
struct {
   int swver;     // Version of software used to record data file
   int fmtver;    // Version of file format
};
```

4.3 DATA BLOCKS

4.3.1 Block Header

Data is stored in blocks; each block has a header consisting of type and length.

```
{
   unsigned int blockType;     // Block type code
   unsigned int blockLength;   // Number of bytes in block, not incl. header
};
```

4.3.2 Block Types

Block types are 32-bit integer values encoded as follows.

These data blocks can occur in any Bathyswath block-encoded data files. They are also used in TCP/IP communications between applications. However, blocks of a certain type are most commonly found in particular files, and these file types are listed in the table below.

Type	Value	Notes	Usual File Type
SONAR_DATA	0x00	Written by version 2 code	Raw (sxr)
SONAR_DATA2	0x16	Replacement raw data type, used in version 3 code until May 08	sxr

Type	Value	Notes	Usual File Type
SONAR_DATA3	0x17	Replacement raw data type, used in version 3 code from May 08	sxr
SONAR_DATA4	0x19	Raw data from V2, complex (IQ) samples.	sxr
COMPASS_DATA	0x01	Obsolete	sxr
MRU_DATA	0x02	Obsolete	sxr
GPS_DATA	0x03	Obsolete	sxr
HWARE_DATA	0x04	Not used	sxr
FILE_DATA	0x05	Not used	sxr
GUI_DATA	0x06	Not used	sxr
NET_DATA	0x07	Not used	sxr
COMPASST_DATA	0x08	Timestamp + ASCII string	sxr
MRUT_DATA	0x09	Timestamp + data from instrument, in the instrument's native format: may be ASCII string or binary	sxr
GPST_DATA	0x0a	Timestamp + data from instrument, in the instrument's native format: may be ASCII string or binary	sxr
AUX1_DATA	0xb	Auxiliary port data (obsolete)	sxr
AUX2_DATA	0x0f	Auxiliary port data, channel 2 (obsolete)	sxr
AUX2_DATA	0x0f	Auxiliary port data	sxr
AUX1T_DATA	0xc	Auxiliary port data, Timestamp + ASCII string	sxr
AUX2T_DATA	0x10	Auxiliary port data, Timestamp + ASCII string	sxr
AUX3T_DATA	0x61	Auxiliary port data, Timestamp + ASCII string	sxr
AUX4T_DATA	0x62	Auxiliary port data, Timestamp + ASCII string	sxr
AUX5T_DATA	0x63	Auxiliary port data, Timestamp + ASCII string	sxr
AUX6T_DATA	0x64	Auxiliary port data, Timestamp + ASCII string	sxr
AUX7T_DATA	0x65	Auxiliary port data, Timestamp + ASCII string	sxr
PHCAL_DATA	0x0d	Phase calibration offsets	sxr
CNF_SENSOR_CORR	0x20	Sensor corrections	sxc
CNF_SENSOR_FILT	0x21	Sensor filters	sxc
CNF_SENSOR_INTERP	0x22	Sensor interpolation	sxc
CNF_DERIVE_ATT	0x23	Attitude derivation	sxc
CNF_ENVIR	0x24	Environment parameters	sxc
CNF_DERIVE_POSN	0x25	Position derivation	sxc
CNF_TOW_POSN	0x26	Tow offsets: towed vehicles	sxc
CNF_POSN_OFFSETS	0x27	Offsets between sensors	sxc

Type	Value	Notes	Usual File Type
SBP_XYZA_PING	0x28	Processed ping data. Written by version 3 code up to January 2010, version 3.6.N	sxp
SBP_XYZA_PING2	0x52	Replacement processed ping data type. Written by version 3.7 onwards.	sxp
COVRG_DATA	0x0e	Coverage map data	swc
TEXT_DATA	0x11	Text message	TCP/IP
SYSTEM_COMMAND_DATA	0x12	Command from one system to another. See §	TCP/IP
TIME_SYNCH_DATA	0x13	Time synchronisation between systems	TCP/IP
PARSED_PING_DATA	0x29	Sonar data in parsed data	sxi & TCP/IP
PARSED_ATTITUDE	0x2b	Attitude data in parsed data	sxi & TCP/IP
PARSED_POSITION_LL	0x2C	Lat-long position data in parsed data	sxi & TCP/IP
PARSED_POSITION_EN	0x2d	Easting-Northing position in parsed data	sxi & TCP/IP
PARSED_SVP	0x2e	Speed of sound data in parsed data	sxi & TCP/IP
PARSED_ECHOSOUNDER	0x2f	Echosounder data in parsed data	sxi & TCP/IP
PARSED_TIDE	0x30	Tide data in parsed data	sxi & TCP/IP
PARSED_AGDS	0x31	AGDS data in parsed data	sxi & TCP/IP
PARSED_AUX_STR	0x32	Auxiliary string in 'parsed' data	sxi & TCP/IP
PARSED_POSITION_LL_2	0x33	Lat-long position data in parsed data, plus altitude	sxi & TCP/IP
CMS_CMD	0x40	Commands in	TCP/IP
CMS_STATUS	0x41	Status out	TCP/IP
AUX_ATTPOS	0X42	Attitude and position	TCP/IP
PARSED_FILTER_CMD	0x43	Controls for Parsed data filtering	TCP/IP
SBP_PROJECTION	0x50	Information on the projection used in processing (PLACEHOLDER)	sxp
SBP_PROCESS_INFO	0x51	Information on the processes used in processing (PLACEHOLDER)	sxp
Reserved	0x100 – 0x1ff	Reserved for client use; contact Bathyswath for details	

At present, only some of these block types are used. All other possible block types are reserved for future expansion. Types `COMPASS_DATA`, `MRU_DATA` and `GPS_DATA` are considered obsolete. Data blocks are concatenated with no further padding and in no particular order (the header record is, however, always the first record in the file).

4.3.3 Block Length

Immediately following the block type is the block length, again as a 32-bit integer. The block length is the number of bytes in the block, not including the header.

5 RAW DATA FILE BLOCKS

5.1 GENERAL

Raw data files are written with the file extension "SXR". They contain the following data blocks.

Note that some of the data block types are now obsolete. For example, the current distribution of Bathyswath software writes the sonar data in the "SONAR_DATA3" format. The "SONAR_DATA2" and "SONAR_DATA" block types only need to be decoded if data files written before 2009 need to be decoded.

5.2 SONAR DATA BLOCK, "SONAR_DATA4"

This format was implemented in Bathyswath code from November 2014, to support Bathyswath V2.

The header is 71 bytes long, and the information is:

Byte num	Num bytes	Item	Data Type	Code	Notes
0	4	Ping number	unsigned long int	pingNum	Unique in each survey session
4	1	Transducer channel	unsigned char	tdrChannel	Up to 4 channels active at any one instant (selected from any number of connected TEMs)
5	1	FPGA code version	unsigned char	fpgaIdent	
6	1	Transducer type	unsigned char	tdrType	See "Transducer type Identifiers" below
7	1	Board type	unsigned char	boardType	See "Board type Identifiers" below
8	8	Board serial number	unsigned short int	boardIdent	
16	4	Operating frequency	float	operatingFreq	See "Transducer Frequencies" below
20	1	Error byte	unsigned char	error	0 = no error
21	1	Calibration	unsigned char	calBit	Set 1 if board is in calibration mode
22	1	Transmit power	unsigned char	txPower	As a percentage of available voltage
23	2	Transmit pulse length	short int	txCycles	In sonar cycles
25	2	Samples in ping	short int	rxSamples	Number of samples in this ping
27	4	Sample period	float	rxPeriod	In seconds
31	1	Code for which ADC channels are enabled	unsigned char	sidescanAdcEnable	Bit code for four possible channels, A, B, C, D

Byte num	Num bytes	Item	Data Type	Code	Notes
32	4	Acquisition time: seconds	long int	timeSecPC	PC clock time
36	2	Acquisition time: milliseconds	short int	timeMsecPC	PC clock time
38	4	Sonar clock time: seconds	long int	timeSecSonar	TEM clock time
42	2	Sonar clock time: milliseconds	short int	timeMsecSonar	TEM clock time
44	1	Position in alternating and simultaneous scans	unsigned char	firstInScan	Used with alternating and simultaneous transmission: first = always 1; alternating or double-sided goes 1,0,1,0 …
45	1	PPS settings & status bits	unsigned char	m_PPS	Bitfield representing 1PPS setting & status
46	1	Ping mode	unsigned char	m_pingMode	Enum indicating mode (single/alt/sim), tx on/off, and port/stbd
47	2	Trigger flags	unsigned short	m_triggerFlags	0: no triggers See 5.3.2 below
49	2	PGA gain	unsigned short	m_pgagain	Gain in the Programmable Gain Attenuators
51	2	LNA gain	unsigned short	m_lnagain	Gain in the Low Noise Amplifiers
53	2	Base gain	unsigned short	m_basegain	Gain in 1/100th db at the end of the transmit pulse
55	2	Linear gain	unsigned short	m_lingain	Amount in 1/100th db/ms the gain rises linearly
57	2	Square gain	unsigned short	m_sqgain	Amount in 1/100th db/ms^2 the gain rises as the square of time
59	4	SW gain	float	m_swgain	Final gain in dB that is applied in software
63	2	Decimation	unsigned short	m_RxDecimation	Hardware sends 1 in n samples; 1 = no decimation
65	4	Filter BW	float	m_rxBandwidth	Filter bandwidth in Hz
69	1	Preamp power	unsigned char	m_preampPowerOn	0 = power off, 1 = on
70	1	Sample type	unsigned char	m_sampleType	0: Phase Diff 1: 4-phase 2: IQ format

Differences from SONAR_DATA3 are:

- Expanded “interval between samples”, to cover a full range of possible values
- Board serial number stored as unsigned short int instead of character string
- Hardware Gain (analogueGain) is removed, as it wasn’t used
- Phase clock full scale (noClocksIn360) removed, as it wasn’t used
- Standard sample type stores phase of all four staves, as unsigned short int, instead of three unsigned char values of phase differences.
- Optionally stores the complex (IQ) data from the V2 system, instead of phase, amplitude and time, as specified in new “Sample type” field in the header

5.3 SAMPLE DATA

The header is followed by a number of sample information items. The number of items is given by the “`rxSamples`” field of the sonar data header.

The sample format can be selected by the operator, and is defined in the Sample type field:

5.3.1.1 Phase Difference Format

Sample type 0, 8 bytes per sample

Byte num	Num bytes	Item
0	1	Phase AB
1	1	Phase AC
2	1	Phase AD
3	1	Transducer number
4	2	Sample number
6	2	Amplitude

5.3.1.2 4-phase Format

Sample type 1, 12 bytes per sample

Byte num	Num bytes	Data	Item
0	2	u short int	Sample number
2	2	u short int	Phase A
4	2	u short int	Phase B
6	2	u short int	Phase C
8	2	u short int	Phase D
10	2	u short int	Amplitude

5.3.1.3 IQ Format

Sample type 2; 34 bytes per sample

Byte num	Num bytes	Data	Item
0	2	u short	Sample number
2	4	float	Stave A, real
6	4	float	Stave A, imag
10	4	float	Stave B, real
14	4	float	Stave B, imag
18	4	float	Stave C, real
22	4	float	Stave C, imag
26	4	float	Stave D, real
30	4	float	Stave D, imag

5.3.2 Trigger

The Trigger Flags parameter is encoded as a bit-field, as follows.

Bit	Value if set	Name	Meaning
Input triggers			
3	8	trigger_gpi1	trigger on GPI1 rising edge
4	16	trigger_gpi2	Trigger on GPI2 rising edge
5	32	trigger_rs485_1	Trigger on RS485 1 rising edge
6	64	trigger_rs485_2	Trigger on RS485 2 rising edge
7	128	trigger_invert	Trigger on falling instead of rising edge
Output triggers			
8	256	trigger_out_gpo1	Trigger on GPO1 rising edge
9	512	trigger_out_gpo2	Trigger on GPO2 rising edge
10	1024	trigger_out_rs485_1	Trigger on RS485 1 rising edge
11	2048	trigger_out_rs485_2	Trigger on RS485 2 rising edge
12	4096	trigger_out_invert	Trigger on falling instead of rising edge

5.4 SONAR DATA BLOCK, "SONAR_DATA3" (CODE FROM MAY 2009)

This version is written by Bathyswath and SWATHplus code distributed after May 2009.

Data within the sonar data block consists of a header followed by the raw sonar samples. The header is always 49 bytes long.

The header information is:

Byte num	Num bytes	Item	Data Type	Code	Notes
0	4	Ping number	long int	pingNum	Unique in each survey session
4	1	Transducer channel	unsigned char	tdrChannel	Up to 4 channels active at any one instant (selected from any number of connected TEMs)

Byte num	Num bytes	Item	Data Type	Code	Notes
5	1	FPGA code version	unsigned char	fpgaIdent	
6	1	Transducer type	unsigned char	tdrType	See "Transducer type Identifiers" below
7	1	Board type	unsigned char	boardType	See "Board type Identifiers" below
8	8	Board identifier string	char string	boardIdent	In effect, the serial number. Often only the first two bytes populated
16	4	Operating frequency	float	operatingFreq	See "Transducer Frequencies" below
20	4	Hardware Gain	float	analogueGain	Not used on current boards
24	1	Phase clock full scale	unsigned char	noClocksIn360	256 in most boards
25	1	Error byte	unsigned char	error	0 = no error
26	1	Calibration	unsigned char	calBit	Set 1 if board is in calibration mode
27	1	Transmit power code	unsigned char	txPower	0 = low power, 15 = max power
28	2	Transmit pulse length	short int	txCycles	In sonar cycles
30	2	Samples in ping	short int	rxSamples	Number of samples in this ping
32	1	Interval between samples	unsigned char	rxPeriod	In microseconds
33	1	Code for which ADC channels are enabled	unsigned char	sidescanAdcEnable	Bit code for four possible channels, A, B, C, D
34	4	Acquisition time: seconds	long int	timeSecPC	PC clock time
38	2	Acquisition time: milliseconds	short int	timeMsecPC	PC clock time
40	4	Sonar clock time: seconds	long int	timeSecSonar	TEM clock time
44	2	Sonar clock time: milliseconds	short int	timeMsecSonar	TEM clock time
46	1	Position in alternating and simultaneous scans	unsigned char	firstInScan	Used with alternating and simultaneous transmission: first = always 1; alternating or double-sided goes 1,0,1,0 …

Byte num	Num bytes	Item	Data Type	Code	Notes
47	1	PPS settings & status bits	unsigned char	m_PPS	Bitfield representing 1PPS setting & status
48	1	Ping mode	unsigned char	m_pingMode	Enum indicating mode (single/alt/sim), tx on/off, and port/stbd

5.4.1 Sample Data

The header is followed by a number of sample information items. Each item is 8 bytes long. The number of items is given by the "`rxSamples`" field of the sonar data header, or by reading the block size, subtracting the size of the sonar data header (49), and dividing by the size of the sample item (8).

The sample information is:

Byte num	Num bytes	Item
0	1	Phase AB
1	1	Phase AC
2	1	Phase AD
3	1	Transducer number
4	2	Sample number
6	2	Amplitude

Note: the phase has the opposite sign in the new SONAR_DATA2 format to that in SONAR_DATA

5.4.2 Board type Identifiers

These codes are used to identify board types.

Value	Name	Description	Notes
1	BRD_TYPE_117_Q0	Quicklogic based 64 way DIN41612 interface (117kHz only)	Development only: shouldn't find in the field
2	BRD_TYPE_117	Quicklogic based 37 way D connector interface 117KHz	
3	BRD_TYPE_ISA	16-Bit ISA Board	Not a TEM. Not used in USB TEM systems.
4	BRD_TYPE_234	Quicklogic based 37 way D connector interface 234KHz	
5	BRD_TYPE_117_A	Altera based 37 way D connector interface 117KHz	
6	BRD_TYPE_234_A	Altera based 37 way D connector interface 234KHz	
7	BRD_TYPE_468_A	Altera based 37 way D connector interface 468KHz	
8	BRD_TYPE_USB	Altera based, USB interface TEM,	

Value	Name	Description	Notes
		468KHz	
9	BRD_TYPE_MK4	Altera based, USB interface TEM, initially for TOBI (freq. in a different register)	
10	BRD_TYPE_V2_A	First version of the V2 TEMs	

5.4.3 Transducer type Identifiers

These codes are used to identify the sonar frequency of the transducers. They correspond to a frequency code that is hard-wired into each transducer's connector.

Value	Name	Nominal Freq. /kHz	Actual Frequency /Hz	Bit Code	Notes
10	TXD_TYPE_117	117	117187.5	"1010"	
5	TXD_TYPE_234	234	234375	"0101"	
13	TXD_TYPE_468	468	468750	"1101"	
15	TXD_TYPE_NO_CONN	-	-	"1111"	No transducer is connected to the TEM
16	TXD_TYPE_117_V2	117			Bathyswath V2 type
17	TXD_TYPE_234_V2	234			Bathyswath V2 type
18	TXD_TYPE_468_V2	468			Bathyswath V2 type
0	TXD_TYPE_DEFAULT	-	-	-	No transducer type specified or auto-detected, so the transducer type is assumed to match the sonar frequency

Note that the hard-wired transducer codes are being removed from the later Bathyswath hardware, so this field cannot be assumed to be provided in later systems.

5.4.4 Transducer Frequencies

These are binary divisions of 30 MHz. The fixed frequencies, used in versions before Bathyswath V2, are:

Name	Value
TXD_FREQ_117	117.1875e3
TXD_FREQ_234	234.375e3
TXD_FREQ_468	468.750e3

V2 can have the frequency set differently, for example to limit cross-talk between port and starboard sides.

5.4.5 FirstInScan Field

- In both simultaneous mode and alternating mode, with a two-TEM system (port and starboard), the file consists of a series of ping records. These will alternate port-starboard-port-starboard. In an ISA TEM system, channel 1, nominally the port TEM, will have firstInScan set to 1, and the starboard TEM firstInScan will be set 0. In the new USB system, the "first" TEM is not necessarily the port one, as there is no guarantee which one the USB driver sees first.
- If you have more than two TEMs fitted and working at the same time (alternating or simultaneous), then just one of the TEMs will have firstInScan set.
- In single-sided mode, firstInScan is always set.

5.4.6 PPS settings & status bits

Information about PPS (pulse per second) is stored in a bit-field, as follows:

Bit	Name	Meaning
0	PPS_DISABLE_BIT	1PPS is enabled / 1PPS is disabled
1	PPS_EDGE_BIT	1PPS acts on rising edge
2	PPS_ACKNOWLEDGE_BIT	TEM has not received 1PPS / TEM has received 1PPS
3	PPS_PERIOD_ERROR_BIT	PPS period matches TEM clock / PPS period does not match TEM clock
4	PPS_USE_PC_TIME	Ignore TEM time altogether and use the PC clock instead

5.4.7 Ping Mode

The operational mode of the sonar is recorded as an enumerated value (enum), as follows:

Value	Name	Meaning
0	SONAR_SEL_OFF	Not used
1	SONAR_SEL_SINGLE	Single-sided pinging
2	SONAR_SEL_ALT	Alternating pinging
3	SONAR_SEL_SIM	Simultaneous pinging

5.5 SONAR DATA BLOCK, “SONAR_DATA2” (V3 CODE TO MAY 08)

This version is written by version 3 SWATHplus code, distributed between September 2006 and May 2008.

Only the ping header is different from SONAR_DATA3: the sonar samples are the same.

Byte num	Num bytes	Item	Data Type	Code	Notes
0	2	Ping number	short int	pingNum	Unique in each survey session
3	1	Transducer channel	unsigned char	tdrChannel	Up to 4 channels active at any one instant (selected from any number of connected TEMs)

Byte num	Num bytes	Item	Data Type	Code	Notes
	1	FPGA code version	unsigned char	fpgaIdent	
	1	Transducer type	unsigned char	tdrType	See "Transducer type Identifiers" below
	1	Board type	unsigned char	boardType	See "Board type Identifiers" below
	8	Board identifier string	char string	boardIdent	In effect, the serial number. Often only the first two bytes populated
	4	Operating frequency	float	operatingFreq	See "Transducer Frequencies" below
	4	Hardware Gain	float	analogueGain	Not used on current boards
	1	Phase clock full scale	unsigned char	noClocksIn360	256 in most boards
	1	Error byte	unsigned char	error	0 = no error
	1	Calibration	unsigned char	calBit	Set 1 if board is in calibration mode
	1	Transmit power code	unsigned char	txPower	
	2	Transmit pulse length	short int	txCycles	In sonar cycles
	2	Samples in ping	short int	rxSamples	Number of samples in this ping
	1	Interval between samples	unsigned char	rxPeriod	In microseconds
	1	Code for which ADC channels are enabled	unsigned char	sidescanAdcEnable	Bit code for four possible channels, A, B, C, D
	4	Time in seconds	long int	timeSec	
	2	Millisecond component of time	short int	timeMsec	
	1	Position in alternating and simultaneous scans	unsigned char	firstInScan	Used with alternating and simultaneous transmission: first = always 1; alternating or double-sided goes 1,0,1,0 …
	2	Spare bytes		spare	For expansion

5.6 SONAR DATA BLOCK, "SONAR_DATA" (V2 CODE)

This version is written by version 2 SWATHplus code, distributed before September 2006.

Data within the sonar data block consists of a header followed by the raw sonar samples. The header is always 160 bytes long – eight 32-bit integers followed by 16 8-byte structures. The following code defines the structure of a sonar header:

```
const int MAX_TX       = 15;
const int MAX_TX_SLOTS  = MAX_TX + 1;

class TxBoardInfo {
public:
   unsigned char tdcrtype;      // Register 0 bits [3:0]
   unsigned char ctrlreg;       // Register 5
   unsigned short int txcycles;  // Register 6 * 8
   unsigned short int rxsamps;   // Register 7 * 256
   unsigned char rxrate;        // Register 8
   unsigned char analch;        // Register 9

   TxBoardInfo ();
};

struct BathyIO {
   int version;
   int ping;
   int txok;
   int active;
   int sec;
   int usec;
   int spare1;
   int spare2;
   TxBoardInfo regs[MAX_TX_SLOTS];
};
```

The `BathyIO` structure contains a version number, ping number, transmit OK flag, active channel number and a timestamp. The version number is currently always set to one. The ping number increments as pings are generated. The transmit OK flag is currently always set to one. The active channel will be set to one for a port side ping and two for a starboard ping. The timestamp is represented as two integers: seconds since 1970 and microseconds since that second. The two spare fields are currently set to hexadecimal constants.

The `TxBoardInfo` class holds copies of the register values that were used to generate the ping.

Following the sonar header is the block of raw sonar samples. There are as many raw sonar samples as the sonar generated during the ping, as controlled by the 'Number of Receive Samples' register. Each sample has the following format:

```
class BathySample {
public:
   unsigned char ab;
   unsigned char ac;
   unsigned char ad;
   unsigned char txno;
   unsigned char samp0;
   unsigned char samp1;
   unsigned char anal0;
   unsigned char anal1;

   BathySample ();
};
```

The three bytes `ab`, `ac` and `ad` represent the phase differences between staves A-B, A-C and A-D. The `txno` byte represents the transducer channel number along with some flags. The pair of bytes `samp0` and `samp1` forms a 16-bit value for the sample number (zero-based). The pair of bytes `anal0` and `anal1` form a 12-bit signed value for the analogue signal amplitude.

The bytes `samp0` and `anal0` are the least-significant bytes, while `samp1` and `anal1` are the most-significant. To obtain the timestamp for a given sample, multiply the sample number by the sample rate, in microseconds. The sample rate is available as one of the register values mentioned earlier. This timestamp represents the round-trip time from sonar transmit to sample reception. The analogue value represents the received sonar signal amplitude at the moment of sampling. Minimum amplitude has a value of -4096 and maximum +4095.

5.7 OBSOLETE, NON-TIMESTAMPED COMPASS, MRU AND GPS DATA BLOCK FORMAT

All the non-timestamped string-type data blocks share the same basic format. The ASCII string simply follows the type/length header. The string is followed by a timestamp in ASCII format, consisting of seconds and microseconds since 1970.

This string-based timestamp was a temporary kludge and has been replaced by a binary-coded timestamp in the current version of this format. These records are no longer generated.

5.8 TIMESTAMPED COMPASS, MRU AND GPS DATA BLOCK FORMAT

All the timestamped string-type data blocks share the same basic format. The header is followed immediately by an eight-byte timestamp, organised as two four-byte integers. The first integer represents seconds since 1970, and the second integer represents microseconds since that second. The data string follows immediately after the timestamp, but is not null-terminated. Therefore, the length of the data block is the length of the data string plus eight. The data string is in the native format of the instrument that sent the data, and could be ASCII text or binary format.

Blocks using this style are: COMPASST_DATA (heading from a compass or dial-antenna GPS), MRUT_DATA (attitude), and GPST_DATA (position).

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Time of receiving the message. All 8-byte time codes are encoded the same: see §7.2.1.
8	N	Unsigned char	Data string	Native format of the instrument sending the data

5.9 AUXILIARY PORT DATA

The Swath Processor program supports eight “auxiliary” inputs. These inputs can be configured to receive data from a range of serial input devices. The data from these channels are stored in the time-stamped data types AUX1T_DATA (for the first channel), AUX2T_DATA (for the second), etc. There is currently no way of knowing which data type has been stored in these auxiliary ports direct from the raw data files. The Swath Processor session file (sxs) stores the configuration of the ports.

Each data item consists of an 8-byte timestamp, followed an ASCII string. The data in the string will depend on the device that supplied the data.

Data that may be present includes:

Type	Sxs Code	Value
Raw ASCII data	FORMAT_AUX_RAW	1
Position data	AUX_INPUT_TYPE_POSITION	2
Heading data	AUX_INPUT_TYPE_HEADING	3
Motion sensor data	AUX_INPUT_TYPE_MRU	4
Speed of sound data	AUX_INPUT_TYPE_SVP	5
Echosounder data	FORMAT_AUX_ECHOSOUNDER	6
Tide data	AUX_INPUT_TYPE_TIDE	7
Acoustic Ground Discrimination System (e.g. ECHOplus)	AUX_INPUT_TYPE_AGDS	8
Commands between systems	AUX_INPUT_TYPE_COMMAND	9
Pressure sensor	AUX_INPUT_TYPE_PRESSURE	10
Cable out sensor	AUX_INPUT_TYPE_CABLE_OUT	11
CTD sensor	AUX_INPUT_TYPE_CTD	12
GPS	AUX_INPUT_TYPE_GPS	13

Magnetometer	AUX_INPUT_TYPE_MAGNETOMETER	14
Light Sensor	AUX_INPUT_TYPE_LIGHT_SENSOR	15

5.10 PHASE CALIBRATION OFFSETS

The phase calibration block, PHCAL_DATA, contains a set of numerical offsets that need to be added to every phase value in the raw data file before phase to angle conversion is done.

The data area in the block consists of an array of C++ structures:

```
CPhaseOffsets m_phaseOffsets[MAX_TX_SLOTS];
```

Where `CphaseOffsets` is defined:

```
class CPhaseOffsets
{
public:
      CPhaseOffsets();
      ~CPhaseOffsets();

      BOOL m_doOffset;      // Flag: apply offsets or not
      // The offsets
      char m_AB;
      char m_AC;
      char m_AD;
};
```

and

MAX_TX_SLOTS = 16 in release 2 code

MAX_TX_SLOTS = 5 in release 3 code.

Therefore, the data area should contain 16 instances of at least four bytes each: offset (or not) flag, then three phase offsets. However, with word alignment and function pointers, this block has a data length of 40 in the R3 code. Most standard survey systems will have the port offsets in array[0] and the starboard offsets in array[1]. Note that transducers are numbered from '1', so transducer 'i' will need offsets applied from array[i-1].

Offsets should be added before conversions. Note that the phases should be stored as "`unsigned char`". This will make sure that the phase "wraps around" (past 255 = 360° phase) when the offset is added.

```
for (i = 0; i < number; i++)
{
      samp[i].m_phaseb_a += m_phaseCorrAB;
      samp[i].m_phasec_a += m_phaseCorrAC;
      samp[i].m_phased_a += m_phaseCorrAD;
}
```

5.11 HARDWARE CONFIGURATION DATA BLOCK FORMAT

A block type number is reserved for hardware configuration data, but no records of this type are generated at present.

6 PROCESSED DATA FILE BLOCKS

6.1 GENERAL

Processed data files are written with the file extension “SXP”. They contain the following data blocks.

Note that some of the data block types are now obsolete.

These items write out memory images of C++ classes, as created by Microsoft Visual Studio. These memory images may have padding between some objects, to align data objects to word boundaries, so caution may be needed when reading these objects with code created by other compilers, languages and operating systems.

The default Windows packing size of 8 bytes is used.

6.2 XYZA DATA BLOCK FORMAT, SBP_XYZA_PING2

This format is used in the processed data files written by SWATHplus code distributed after January 2010. It contains all the processed data for a single ping. It is similar to the formats used internally by the SEA SWATH software, but the structures are defined separately in order to keep control of file size.

Following the block header, there are three kinds of element in the data block:

- Ping data (class cXYZAPing)
- Transducer data header (class cXYZATxer)
- Bathymetric data samples (class cXYZAPoint)

Each block contains the data from one transducer. If the sonar is operated in “simultaneous” mode, with both transducers firing at the same time, then two separate “SBP_XYZA_PING2” blocks will be generated, with the same time stamp.

Each block therefore contains one each of the data types, in order:

- Ping data
- Transducer data header
- Bathy data array

The ping data is as follows:

```
class cXYZAPing
{
	char m_lineName[MAX_LINENAME_LEN];	// line name
	unsigned long m_pingNum;		// ping number
	double m_time;			// UNIX time of start of ping
	int m_noTxers;			// Number of transducers used (always 1)
	CPosn m_posn;			// position of transducer
	double m_roll;			// roll at start of ping
	double m_pitch;			// pitch at start of ping
	double m_heading;		// heading at start of ping
	double m_height;		// height of transducer at start of ping
	double m_tide;			// tide height applied
	double m_sos;			// speed of sound applied (mean value)
};
```

```
const int   MAX_LINENAME_LEN = 40;
```

The sign conventions are explained in section 11.1. Height is the height below datum (measured positive down), and combines the heave and datum offset, which could come from GPS height or tide, for example.

Tide is measured with the usual marine convention, positive up.

The member `m_noTxers` is used to determine the number of transducer blocks and data arrays that follow.

The transducer data is:

```
// Data for transducer
class cXYZATxer
{
public:
	unsigned char m_txNo;		// transducer identifier
	unsigned char m_txStat;		// tx status
	unsigned char m_txPower;
	short int m_analogGain;		// analog gain value

	unsigned char m_noStaves;	// no. of staves on tx
	unsigned char m_txInfo[MAX_TX_INFO];	// board type/revision/serial number
	unsigned char m_freq;		// tx frequency (identifier code)
	double m_frequency;		// frequency in hertz

	short int m_trnsTime;		// transmit time/ number of cycles
	short int m_recvTime;		// receive time/ number of samples
	unsigned char m_sampRate;	// receive sample rate micro-seconds per sample

	// sample data
	int m_noSampsOrig;		// no. of samples read in real time
	int m_noSampsFile;		// no. of samples in the processed file
	int m_noSampSlots;		// no. of sample slots

	Cposn m_posn;			// position of transducer (E,N)
	CposnOffset m_posoffset;	// position offset of this transducer from survey centre
};
```

Sub-definitions:

```
// Position
class Cposn
{
	double E;		// easting
	double N;		// northing
};
```

```
// General offset from survey centre
class CPosnOffset
{
	double height;
	double forward;
	double starboard;
	double azimuth;
	double elevation;
	double skew;
	double time;
	double water_depth;
	double pitch;
};

const int   MAX_TX_INFO = 4;
```

The number of samples stored in the array that follows is `m_noSampsFile`. The number of samples stored in the original array was `m_noSampsOrig`, but some of these samples may have been rejected by filters and not written to the file.

This data array contains the three-dimensional position (xyz) and amplitude derived from the sonar data. The data points are not ordered in any particular way, but they will usually be stored in the order of the time in which the underlying phase data was collected.

Each point is encoded as follows:

```
class cXYZAPoint
{
	int                 m_sampNum;    // sample number
	double              m_x;          // x position (northing)
	double              m_y;          // y position (easting)
	float               m_z;          // depth (positive down)
	unsigned short int  m_amp;        // raw amplitude (16 bits)
	unsigned short int  m_procAmp;    // processed amplitude (16 bits)
	unsigned char       m_status;     // extra information
	double              m_TPU;        // uncertainty
};
```

The `m_status` field gives information about the status of the data point. A value of zero indicates that the point has been rejected by a filter.

Note that the three-dimensional xyz axis set is:

x Northing

y Easting

z Depth, positive down

6.2.1 Note on Change from 17/10/15

On 16/10/15, CPosnOffset was accidentally changed to add "bool inverted", on 16/10/15. Software with this change was released as R3.10.14.1. This added 8 bytes to the size of cXYZATxer. Because this change was not intentional, the block type was not changed. This error was corrected on 09/08/16, R3.11.15.1, to restore the block size. To deal with SXP files written between those times, it is necessary to read the software version from the block header (see section 4.2.2) and use that to control the size of the offset to the start of the data items:

```
int offset = sizeof(cXYZATxer);
const int sizeChange = 8;
if ((m_swver_read >= 3101401) && (m_swver_read <= 3111501)
       offset += sizeChange;
pointArray = (cXYZAPoint*)((char*)txerfile + offset); // Array of data points
```

6.3 XYZA DATA BLOCK FORMAT, “SBP_XYZA_PING”

This format is used in the processed data files written by SWATHplus code distributed before January 2010.

The ping data element is the same as in SBP_XYZA_PING2.

Within the transducer data the position offset sub-element contains one fewer fields:

```
// General offset from survey centre
class CposnOffset
{
       double height;
       double forward;
       double starboard;
       double azimuth;
       double elevation;
       double skew;
       double time;
       double water_depth;
};
```

The individual data points each contain one fewer fields:

```
class cXYZAPoint
{
       int                 m_sampNum;    // sample number
       double              m_x;          // x position (northing)
       double              m_y;          // y position (easting)
       float               m_z;          // depth (positive down)
       unsigned short int  m_amp;        // raw amplitude (16 bits)
       unsigned short int  m_procAmp;    // processed amplitude (16 bits)
       unsigned char       m_status;     // extra information
};
```

7 PARSED DATA FILE BLOCKS

7.1 GENERAL

Raw data files are written with the file extension “SXI”. They contain the following data blocks.

7.2 COMMON PARSED DATA CODES

The parsed data blocks encode time and data channel the same way, as follows.

7.2.1 Time Codes

Eight-byte timestamps are organised as two four-byte integers. The first integer represents seconds since 1970, and the second integer represents microseconds since that second.

7.2.2 Channel Number

Channel number identifies a transducer. The software that reads the data will need to store, and account for the location and pointing angle of each transducer, in three dimensions, relative to the attitude and position system data. There are usually, but not always, two channels. Channel 1 is usually port, and channel 2 is usually starboard, but that is not guaranteed. Transducers may fire alternately (port-starboard-port-starboard …), simultaneously, or singly (port-port-port … or starboard-starboard-starboard).

7.3 PARSED_PING_DATA

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the transducer
9	4	unsigned long int	Ping number	Starts from when program starts 1 and increments. Simultaneous pings are numbered separately.
13	4	Float	Sonar frequency	Frequency of the transducer, in Hz
17	4	Float	Sample period	Time period between sonar data samples, in seconds.
21	2	unsigned short int	Number of samples	Number of samples following
23	4	Float	Sound speed	Speed of sound used to calculate angles, m/s
27	2	Short int	Tx pulse	Transmit pulse length, in sonar cycles
29	1	Char bit field	Data options	Allows options in data encoding. See §7.3.1.
30	1	Unsigned char	Ping state	Records the status of pinging: single/alternating/simultaneous etc. See §7.3.3
31	2	Unsigned short int	Max count	Maximum data count before filtering
33	2	To be determined	Reserved	Reserved for other ping information
				The header information is followed by "Number of samples" examples of the following sample data
35 +(n×7)	2	unsigned short int	Number	Sample number. Calculate range from this value: see §7.3.5. Sample number may not be sequential or increasing.
37 +(n×7)	2	signed short int	Angle	Angle coded +15 bits = 180° up, -15 bits = 180° down, relative to the transducer pointing angle. See §7.3.5.
39 +(n×7)	2	unsigned short int	Amplitude	Scaled so that 16 bits is the full scale of the ADC
41 +(n×7)	1	Unsigned char	Quality	As set by "Data options". See §7.3.1.

7.3.1 Sonar Data Options

The "Data Options" byte in the header part of the PARSED_PING_DATA block allows for options in the encoding of data, encoded as follows:

Bit codes:

7	6	5	4	3	2	1	0
Not currently used					Meaning of quality byte		

Bits 0-2 are used to encode the meaning of the quality byte. The remaining bits are currently not used.

Value	Name	Meaning The quality byte in each sample is …
0	PARSED_ QUALITY_MERGED_CALC	Generated from a combination of quality factors. A measure of relative quality. 255 = maximum quality, 0 is a rejected item.
1	PARSED_ QUALITY_PHASE_QUAL	Generated from phase decode quality. The SWATHplus transducers have several stave pairs to use: this value is a measure of how well the decodes for each pair agree
2	PARSED_ QUALITY_FILTER_ACCEPTANCE	A bit-wise set of accept-reject flags for a set of up to 8 filters. If a filter rejects a sample, then the bit corresponding to that filter is set to 1. If the data point is accepted, or that bit is not used (there isn't a filter assigned to the bit), then it is left zero. Accepted data points therefore have all bits set zero.

7.3.2 Quality Filters

If working in "PARSED_ QUALITY_FILTER_ACCEPTANCE" mode, the meaning of the bits in the quality flag is as follows:

Bit	Name	Meaning
0	FILTER_FLAG_PHASE	Phase coherence filter: checks that the angle decodes for various combinations of transducer stave pairs agree
1	FILTER_FLAG_ANGLE_CONFIDENCE	A moving-window filter in angle, checking that angles are within some set limit
2	FILTER_FLAG_AMPLITUDE	Checks that amplitude is greater than the base noise level
3	FILTER_FLAG_ZER0_ANGLE	Flags items that have an angle of exactly zero, or are within some user-defined angle of the transducer boresight
4	FILTER_FLAG_RANGE	Flags items that are outside user-defined minimum and maximum angles
5	FILTER_FLAG_WATERCOL	Flags items that either have a slant range less than an average nadir depth, or are within 30° of the nadir and are greater than the average nadir depth. In both cases, a percentage threshold margin is allowed either side of the nadir depth.
6	FILTER_FLAG_DEPTH	Flags items that are outside user-defined minimum and maximum depth

The user may choose not to output any points that have any filter flags set, in which case all filter bytes in the output will be zero.

7.3.3 Status of Pinging

This byte records the sonar activation options.

If the byte is set zero, then this byte has no meaning. This allows for successful decoding of previous versions of the data file.

7	6	5	4	3	2	1	0
Not currently used				Port-starboard (set 1 for stbd)	Tx on (otherwise receive only)	Ping mode	

Ping Mode:

Value	Name	Meaning
0	SONAR_SEL_OFF	Not used
1	SONAR_SEL_SINGLE	Single-sided pinging
2	SONAR_SEL_ALT	Alternating pinging
3	SONAR_SEL_SIM	Simultaneous pinging

Port-starboard flag is set using the transducer position data. There is no guarantee that the user has set this data, so the validity of this field is not guaranteed.

7.3.4 Sonar Data Angle

Sonar data angle is relative to the transducer pointing angle. For SWATHplus, the pointing angle is usually 30° down and -90° azimuth for port and +90° for starboard, but any configuration is theoretically possible. Angular measurement is only for the semicircle in front of the transducers.

The sign convention is positive up. On a flat seabed, samples will start negative and change to positive when the returns sweep through the boresight of the transducer.

The conversion factor into radians is (pi/ 32768), and (180/32768) for degrees.

If SWATHplus is not configured with the true speed of sound at the transducer head, then the angles need to be corrected for this speed of sound in the program that consumes this data. The correction is as follows:

True angle = arcsin(sin(angle) * (true sound speed) / (nominal sound speed))

7.3.5 Sonar Data Range

Range is calculated by:

*range = (sample number) * (sample period) * (speed of sound) / 2;*

Range is always in front of the transducer, in a line along the transducer “boresight”.

Note that the speed of sound used is likely to be a default value, not taken from a measurement. Therefore, the range should be corrected by multiplying by the ratio between measured speed of sound and the speed of sound in the PARSED_PING_DATA sample.

A possible refinement is to modify range to the mid-point of the sonar pulse, using "Tx pulse" and the sonar frequency.

7.3.6 Sonar Data Time

The precise time of a sonar data sample is calculated by:

(PARSED_PING_DATA time) + ((PARSED_PING_DATA sample period) * (sample number)

7.3.7 Height

Height could be:

- An absolute height, from a combined attitude and position system that is capable of reading GPS height to sufficient accuracy
- A height relative to the water surface: in this case a tide table will be needed
- A heave value: that is, relative to a local average height

It is up to the processing application that uses this data to make the distinction for each case and deal with the data appropriately.

7.4 PARSED_ATTITUDE

Byte num	Num bytes	Encoding	Item	Notes
0	8	See below	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the data source
9	4	Float	Roll	Positive for starboard down
13	4	Float	Pitch	Positive for nose up
17	4	Float	Heading	Positive clockwise, looking down
21	4	Float	Height	Positive for down

Attitude information can come from a mix of sensors. The Swath system builds combined attitude and heading packets from whatever comes into the system, using user-entered "attitude derivation" selections. Data packets are provided at the same rate as whatever is providing roll (which needs to be at the highest frequency, and so should not be sub-sampled). If heading is at a lower rate, it would be repeated in attitude packets until a new item comes in.

See §11.1 for notes on sign conventions.

7.5 PARSED_POSITION_LL

Byte num	Num bytes	Encoding	Item	Notes

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same.
8	1	Unsigned char	Channel	Identifies the data source
9	8	double	Latitude	Degrees
17	8	double	Longitude	Degrees

7.5.1 Position Latitude-Longitude and Easting-Northing

Position systems can provide position data either in latitude and longitude or easting and northing.

- If the positioning system provides latitude and longitude, then PARSED_POSITION_LL packets are provided. The system will also probably provide PARSED_POSITION_EN packets, using the conversion factors selected by the SWATHplus operator. However, this converted EN packet is not guaranteed in this case.
- If the positioning system only provides easting and northing, then PARSED_POSITION_EN packets only are supplied: no conversion to LL is provided.

7.6 PARSED_POSITION_LL_2

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same.
8	1	Unsigned char	Channel	Identifies the data source
9	8	double	Latitude	Degrees
17	8	double	Longitude	Degrees
25	8	double	altitude	metres
33	8	double	geoid	metres

7.7 PARSED_POSITION_EN

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same.
8	1	Unsigned char	Channel	Identifies the data source
9	8	double	Easting	Metres
17	8	double	Northing	Metres

7.8 PARSED_SVP

Byte num	Num bytes	Encoding	Item	Notes

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the data source
9	4	float	Speed of sound	Metres per second

7.9 PARSED_ECHOSOUNDER

This is only sent out if a separate single-beam echosounder is fitted to the SWATHplus system.

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the data source
9	4	float	Altitude	Height above seabed; Metres

7.10 PARSED_TIDE

This is only sent out if tide data is provided to the SWATHplus system in real-time.

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the data source
9	4	float	Tide height	Metres

7.11 PARSED_AGDS

This is only sent out if a separate Acoustic Ground Discrimination System (e.g. SEA's ECHOplus) is connected to the SWATHplus system.

Byte num	Num bytes	Encoding	Item	Notes
0	8	See §7.2.1.	Time	Start of ping time code. All 8-byte time codes are encoded the same: see §7.2.1.
8	1	Unsigned char	Channel	Identifies the data source
9	4	float	Hardness	
13	4	float	Roughness	

8 REAL-TIME COMMAND AND STATUS

8.1 REAL-TIME COMMAND AND STATUS DATA FORMATS

There are essentially two classes of command and status connections used with the Swath Processor program:

1. Connection between two Swath Processor programs. This might be used if data is collected in a remote location and processed and visualised on another computer. This connection uses the same data blocks as for the raw data file, plus the following command types: TEXT_DATA, SYSTEM_COMMAND_DATA and TIME_SYNCH_DATA.
2. Connection between Bathyswath and an external system.
    a. Several different interface formats are available; these have been developed to interface to specific platforms, and the software responds to any of them. They include:
        i. Data blocks: CMS_CMD, CMS_STATUS and AUX_ATTPOS; see section 8.3
        ii. '>' System Commands: see section 8.3.2
        iii. NMEA 0183 style messages: see 8.3.3 and 8.3.4

8.2 CONNECTION BETWEEN SWATH PROCESSORS

8.2.1 TEXT_DATA

The data payload is a character string (the length is defined by the block header). The receiving program simply displays the text in the Status view window.

8.2.2 SYSTEM_COMMAND_DATA

A single integer follows the header, defined as follows:

Type	Value	Notes
SCOMMAND_SHUTDOWN	0	Sent from one process to the other, indicating that it is shutting down. It causes the other process to shut down its TCP/IP socket.
SCOMMAND_DATARESET	1	The idea is to cause the other process to reset its data buffers. It is disabled in software at the time of writing.
SCOMMAND_KILL_APP	2	Causes the receiving application to shut down.

8.2.3 TIME_SYNCH_DATA

The data payload is a Windows SYSTEMTIME structure. It causes the receiving application to set the computer's clock to the time transmitted to it.

8.3 EXTERNAL SYSTEM INTERFACE, INPUTS

8.3.1 Swath Processor Options

Bathyswath can be controlled by sending command strings to an Aux port, on a serial or UDP connection. Options are:

Type	Format	Format name	Section
Serial Control	SEA '>' commands	'>' System Commands	8.3.2
	NMEA 0183-type[1]	$PMISS Commands	8.3.3
		$SWPCT Control Messages	8.3.4
		$RMPOS vehicle position	8.3.5
		$RMPO1 vehicle position with time stamp	8.3.6
		$RMATT vehicle attitude	8.3.7
		$RMZDA vehicle time	8.3.8

8.3.2 '>' System Commands

Swath can be controlled by control codes passed to it over the Aux ports.

The codes are simple ASCII text messages. Each message must start with a '>' character.

Available codes are:

Command	Meaning
>SNR ON	Sonar control: start pinging
>SNR OFF	Sonar control: stop pinging
>SNR PWR power	Sonar control: set power to specified number, 1 to 10. E.g. ">SNR PWR 8"
>SNR RNG range	Sonar control: set ping range in metres
>SNR PLS pulselength	Sonar control: set pulse length in cycles, 2 to 250. E.g. ">SNR PLS 100"
>SNR PRF pingrate	Sonar control: set ping repetition frequency in Hz, 1 to 30. E.g. ">SNR PRF 20"
>SNR SMP samples per ping	Sonar control: set the number of samples per ping, 512 to 8192. E.g. ">SNR SMP 4096"
>FILE ON	Start writing to raw file
>FILE OFF	Stop writing to raw file
>FILE CLOSE	Close raw file
>FILE PAUSE	Pause writing to raw file
>FILE OPEN filename	Create a new raw file, with name given by "filename". E.g. ">FILE OPEN line3".
>SESSION LOAD filename	Load a session file with the specified name
>SESSION SAVE filename	Save the session file with the specified name

[1] The system responds to any of the message types that arrive

8.3.3 "$PMISS" Commands

Bathyswath responds to mission state messages. These are headed with the string "$PMISS", and are handled by Bathyswath as described in the following table. The mission state messages are all encoded using the "NMEA0183" format. These have a leading '$' character, followed by an identifying string ("PMISS"), and terminating with a star ('*') and two-digit hexadecimal checksum (signified by "CK" in the table below).

Message	Typically Sent	Bathyswath Action	Notes
$PMISS,MISSION_START	At power-up, when vehicle starts to receive status messages from Bathyswath	Selected by the Bathyswath "session file" settings. Choose from some, all or none of: Turn on TEMs, Start receive, Start transmit, Open & write to file.	Not used if "SURVEY_" and "LINE_" messages are used
$PMISS,SURVEY_START	When the vehicle reaches the survey area	Turn on electronics (TEMs). This increases the Bathyswath hotel load from about 8W to 30W	
$PMISS,LINE_START	at the start of each survey line	Start a new data file, named according to time	
$PMISS,LINE_END	at the end of each survey line	Close the data file	
$PMISS,SURVEY_END	when the vehicle finishes the survey, and wants to start a transit without shutting down Bathyswath completely	Turn off the sonar electronics, dropping the hotel load to about 8W	
$PMISS,MISSION_END	When vehicle has finished with Bathyswath altogether	User settable, similar to MISSION_START. Can be used to shut down the single board computer operating system, thus reducing power further	Once the OS has been shut down, the only way to re-start it is to power-cycle the Bathyswath bottle

8.3.4 $SWPCT Control Messages

These messages provide finer control of the Bathyswath system than is possible with the Mission State Messages; see above. They also use the "NMEA0183" message structure.

Message	Function
$SWPCT,TEM,ON,*CK\r\n	Apply power to the sonar electronics (TEMs)

Message	Function
$SWPCT,TEM,OFF,*CK\r\n	Remove power from the sonar electronics (TEMs)
$SWPCT,SNR,ON,*CK\r\n	Sonar control: start pinging
$SWPCT,SNR,OFF,*CK\r\n	Sonar control: stop pinging
$SWPCT,SNR,PWR, power,*CK\r\n	Sonar control: set power to specified number, 0 to 100. e.g. $SWPCT,SNR,PWR, 8,CK\r\n"
$SWPCT,SNR,RNG, range,*CK\r\n	Sonar control: set ping range in metres
$SWPCT,SNR,PLS, range,*CK\r\n	Sonar control: set pulse length in cycles, 2 to 250. e.g. $SWPCT,SNR, RNG, 100,CK\r\n"
$SWPCT,SNR,PRF, pingrate,*CK\r\n	Sonar control: set ping repetition frequency in Hz, 1 to 30.
$SWPCT,SNR,SMP,samples per ping,*CK\r\n	Sonar control: set the number of samples per ping, 512 to 8192.
$SWPCT,FILE,ON,*CK\r\n	Start writing to raw file
$SWPCT,FILE,OFF,*CK\r\n	Stop writing to raw file
$SWPCT,FILE,CLOSE,*CK\r\n	Close raw file
$SWPCT,FILE,PAUSE,*CK\r\n	Pause writing to raw file
$SWPCT,FILE,OPEN,filename,*CK\r\n	Create a new raw file, with name given by "filename".
$SWPCT,FILE,FLDR,pathname,*CK\r\n	Save raw files to the folder path specified
$SWPCT,SESSION,LOAD,filename,*CK\r\n	Load a session file with the specified name
$SWPCT,SESSION,SAVE,filename,*CK\r\n	Save the session file with the specified name
$SWPCT,TTMODE,MASTER,*CK\r\n	Set the Transmit Trigger mode to "Master"
$SWPCT,TTMODE,SLAVE,*CK\r\n	Set the Transmit Trigger mode to "Slave"
$SWPCT,TTMODE,NULL,*CK\r\n	Set the Transmit Trigger mode to "Null"

8.3.5 $RMPOS vehicle position

Latitude-longitude vehicle position, e.g.

$RMPOS,041N31.0592,070W41.9350,H89.9,D2.2,A20.1*0A\r\n

8.3.6 $RMPO1 vehicle position with time stamp

Latitude-longitude vehicle position with time stamp, e.g.

$RMPO1,16435.598,04,02,2008,41N40.366,070W38.799,H324.5,D9.2,A0.0*76

8.3.7 $RMATT vehicle attitude

Vehicle attitude, e.g.:

$RMATT,164310.434,04,02,2008,R-0.59,P-13.22,D11.64,H328.69,*66

8.3.8 $RMZDA vehicle time

Vehicle time, e.g.:

$RMZDA,090414.250,08,02,2008,00,00*51

8.4 EXTERNAL SYSTEM INTERFACE, OUTPUTS

Outputs from Swath Processor Aux ports are:

Type	Format	Format name	Section
Status	NMEA 0183	NMEA 0183 STATUS	8.4.1
	JetSWATH	JetSWATH	8.4.2
Configuration	Bathyswath	Bathyswath Configuration Messages	8.4.3
Depth	Simple	Simple Depth Out	8.4.4
	NMEA 0183 DBT	NMEA DBT message	8.4.5
	NMEA 0183 DPT	NMEA DPT message	8.4.6
	NMEA 0183 DPT - water col	NMEA DPT watercolumn message	8.4.7
Info	n/a	NMEA information message	8.4.8
Attitude	n/a	Attitude echo message	8.4.9

8.4.1 NMEA 0183 STATUS

This is a message in the style of NMEA 0183 messages, made up as follows:

$P1MSG,<mode_message>,<flag><status_message>*<CS><cr><lf>

where :

<CS> is an NMEA 0183-style checksum.

<mode_message> is one of the following :

Message	Meaning
idle	The sonar is not pinging
active_rx_only	The sonar is operating, but not transmitting
active_tx	The sonar is operating and transmitting, but not writing data to disk
active_tx_write	The sonar is operating and transmitting, and writing data to disk

<status_message> is made up of a string of as many of the following as apply (and so is empty if there are no errors)

Message	Meaning
Tder_	There is a problem with a transducer
Txer_	There is a problem with a sonar transmitter
TEM_	There is a problem with the transducer electronics module
Temp_	There is a temperature problem
Leak_	There is a water leek (in Bathyswath systems that are fitted with a leak sensor)
Disk_	There is a problem with the internal data storage disk
Sonar_	There is a problem with the sonar electronics
Att_	There is a problem with the attitude sensor or the compass

Message	Meaning
Posn_	There is a problem with the position sensor
Aux_	There is a problem with an auxiliary sensor

The flag can be one of the following:

Flag	Meaning
<blank>	All OK
!	Error: an error state has been detected
@	Abort: a fault state has been detected that could endanger the vehicle (e.g. a water leak): abort the mission

8.4.2 JetSWATH

This is a message that supports the obsolete JetSWATH systems, and so is no longer used.

8.4.3 Bathyswath Configuration Messages

These are messages in the style of NMEA 0183 messages. A separate message is sent for each state. The messages are made up as follows:

$ SWCNF,<message>,*<CS><cr><lf>

where :

<CS> is an NMEA 0183-style checksum

<message> is one of the following

Message	Meaning
TEM,ON	The TEM is powered on
TEM,OFF	The TEM is powered off
SNR,ON	The sonar is active and transmit is enabled
SNR,OFF	The sonar is not active or transmit is disabled
SNR,PWR,<power>	Shows the sonar power level
SNR,RNG,<range>	Shows the sonar range in metres
SNR,PLS,<pulse>	Shows the sonar pulse length in cycles
SNR,PRF,<prf>	Shows the sonar ping repetition frequency in Hz
SNR,SMP,<samples>	Shows the number of cycles in each ping
SNR,FILE,<ON/OFF>	Writing to file is enabled or disabled
SNR,FILE,<OPEN/CLOSED/PAUSED>	The raw data output file is open, closed or paused
SESSION,OPEN,<name>	A session file (.sxs) with the given name is open

8.4.4 Simple Depth Out

The nadir depth is written out in metres, followed by carriage return & line feed.

8.4.5 NMEA DBT message

The nadir depth is written out as a standard NMEA 1083 DBT message, with talker ID set to “BS”.

8.4.6 NMEA DPT message

The nadir depth and the depth under the water surface of the transducers are written out as a standard NMEA 1083 DPT message, with talker ID set to “BS”.

8.4.7 NMEA DPT watercolumn message

The nadir depth plus the depth under the water surface of the transducers, and the depth under the water surface of the transducers, are written out as a standard NMEA 1083 DPT message, with talker ID set to “BS”.

8.4.8 NMEA information message

General information messages sent as NMEA 0183 messages. The messages are made up as follows:

$SWINFO,<message>,*<CS><cr><lf>

The messages sent are a selected subset of the messages that appear in the Swath Processor Status window.

8.4.9 Attitude echo message

If this is enabled, then any attitude information that is input to the Attitude port is echoed out to an Aux port, in whatever format the data was input.

9 OTHER FILES

9.1 COVERAGE FILES

Swath Processor writes coverage files when it writes raw data (sxr) files. These files summarise the location and extent of each sonar ping that has been recorded. Coverage files can also be re-generated from raw data files if required.

9.1.1 General

Coverage files are written with the file extension "SXC". They contain the following data blocks.

9.1.2 Coverage Data Block, "COVRG_DATA"

The block ID for coverage data blocks is "0x0e".

The data payload is a C++ class:

```
class CCoveragePoint
{
public:
	CPosn m_boatPos;			// Position of boat
	CCovTdxr m_tdxr;			// Transducer samples
	e_posType m_corrPosType;		// lat/lon or E/N
	short int m_chanNo;			// transducer channel number
};
```

With:

```
class CPosn
{
public:
	double E;		// easting or longitude
	double N;		// northing or latitude
};

class CCovTdxr
{
public:
	BOOL m_valid;		// This is a valid sample
	CPosn m_near;		// Near point
	CPosn m_far;		// Far point
};


enum e_posType
{
	POS_LL,			// latitude-longitude
	POS_EN			// easting-northing
};
```

10 HEIGHT AND POSITION CORRECTION FILES

Height correction files are used to correct the datum height (usually from GPS position), given the position at a point. These corrections are usually made to account for the geoid: the lumpy shape of the Earth relative to the smooth ovoid that is implied by the ellipsoid definition that was used for the position projection from latitude and longitude to easting and northing. These corrections take the form of a table of offsets, and the offset at a given point is looked up from the table using the position of the point. The table can be defined against latitude and longitude positions, in which case it is called a "geoid file", or against easting and northing projected positions, in which case it is called a "height offset file".

A number of "standard" formats can be read into the Bathyswath software, including BIN, MNT, GEO, and KTD. We have also defined our own formats, in case the user finds it necessary to create a new offset file.

10.1 SCG BATHYSWATH GEOID FILE

This file defines height offsets (usually from the geoid), relative to latitude and longitude position.

It is ASCII format, so is human-readable, and can be created and edited with a text editor or spreadsheet program. Fields can be separated with any "whitespace" character, such as space or tab. The length of each field is not important. Decimal fractions are separated with a point, e.g. one-and-a-half is shown as "1.5".

Field number	Item	Encoded as
0	Number of latitude values	Integer number
1	Number of longitude values	Integer number
2	Latitude of SW corner	degrees and decimal degrees
3	Longitude of SW corner	degrees and decimal degrees
4	Step in latitude between samples	degrees and decimal degrees
5	Step in longitude between samples	degrees and decimal degrees

This is followed by the table itself, consisting of exactly "Number of latitude values" x "Number of longitude values" samples, each being a correction to the height, positive for an upwards correction. (Note that heights in Bathyswath are usually negative downwards – deeper – unless otherwise stated; in this case the "normal" convention for geoid corrections is used instead of the Bathyswath one).

It may be convenient for readability to place a newline character after every row of values, but this is not necessary for the software to parse the file correctly.

Note also that, when viewed in a text editor, this table is upside-down relative to the location on the map: if arranged with a newline after each longitude row, then the south-west corner (the first value in the table) appears in the top-left position.

10.2 SCG BATHYSWATH HEIGHT OFFSET FILE

This file defines height offsets (usually from the geoid), relative to easting and northing position.

It is ASCII format, so is human-readable, and can be created and edited with a text editor or spreadsheet program. Fields can be separated with any "whitespace" character, such as space or tab. The length of each field is not important. Decimal fractions are separated with a point, e.g. one-and-a-half is shown as "1.5".

It is very similar to the geoid file, but defines easting and northing instead of latitude and longitude (but not in that order).

Field number	Item	Encoded as
0	Number of easting values	Integer number
1	Number of northing values	Integer number
2	Easting of SW corner	metres and decimal metres
3	Northing of SW corner	metres and decimal metres
4	Step in easting between samples	metres and decimal metres
5	Step in northing between samples	metres and decimal metres

This is followed by the table itself, consisting of exactly "Number of easting values" x "Number of northing values" samples, each being a correction to the height, positive for an upwards correction. (Note that heights in Bathyswath are usually negative downwards – deeper – unless otherwise stated; in this case the "normal" convention for geoid corrections is used instead of the Bathyswath one).

It may be convenient for readability to place a newline character after every row of values, but this is not necessary for the software to parse the file correctly.

Note also that, when viewed in a text editor, this table is upside-down relative to the location on the map: if arranged with a newline after each longitude row, then the south-west corner (the first value in the table) appears in the top-left position.

10.3 SCP BATHYSWATH POSITION CORRECTION FILE

This file defines position offsets relative to easting and northing position.

It is ASCII format, so is human-readable, and can be created and edited with a text editor or spreadsheet program. Fields can be separated with any "whitespace" character, such as space or tab. The length of each field is not important. Decimal fractions are separated with a point, e.g. one-and-a-half is shown as "1.5".

It is very similar to the height offset file, but specifies easting and northing offsets on each line instead of height.

Field number	Item	Encoded as
0	Number of easting values	Integer number
1	Number of northing values	Integer number
2	Easting of SW corner	metres and decimal metres
3	Northing of SW corner	metres and decimal metres
4	Step in easting between samples	metres and decimal metres
5	Step in northing between samples	metres and decimal metres

This is followed by the table itself, consisting of exactly "Number of easting values" x "Number of northing values" offsets, each consisting of two numbers:

Field number	Item	Encoded as
0	Easting offset	Decimal number
1	Northing offset	Decimal number

It may be convenient for readability to place a newline character after every row of values, but this is not necessary for the software to parse the file correctly.

Note also that, when viewed in a text editor, this table is upside-down relative to the location on the map: if arranged with a newline after each longitude row, then the south-west corner (the first value in the table) appears in the top-left position.

11 NOTES

11.1 AXIS CONVENTIONS

All angles obey the right-hand screw rule about the appropriate axis. Headings obey this rule about the z-axis, whilst at the same time maintaining the usual geographic convention, that is, positive going clockwise, measured from North.

Depth is positive for *down*.

This means that:

- Heading is positive clockwise, looking down.
- Roll is positive for starboard down.
- Pitch is positive for nose up.

The SWATHplus software uses the Euler angle convention for roll and pitch, rather than the horizontal-plane convention. This means that roll is measured about the body's own forward-aft axis, rather than relative to the horizontal plane.

Tide values are given in the usual marine convention, positive up.

12 CODE EXAMPLES

This section gives example code for reading Bathyswath data files.

12.1 STEPS IN READING A FILE

To read a data file:

- Read a block header, as two 32-bit integers
- Read in the rest of the block, using the block length from the header
- Use the block type from the header to decide which decode to use. Ignore the block if that type is not needed.
- Read the next header

12.1.1 Read the header

```
   struct{
      unsigned int m_blockType;     // Block type code
      unsigned int m_blockLength;    // Number of bytes in block, not incl.
header
   } CBlockHeader;
```

For example:

```
       switch (blockType)
       {
       case SONAR_DATA4:
              // read raw sonar data
              break;
       case MRU_DATA:
              // read attitude data
              break;
       … etc.
       }
```

12.1.2 Read raw sonar data

Read the header of the SONAR_DATA4block

```
{
       m_pingNum           = GetUInt(pBuffer);
       m_tdrChannel        = (int) GetUChar(pBuffer);
       m_fpgaIdent         = (int) GetUChar(pBuffer);
       m_tdrType           = (int) GetUChar(pBuffer);
       m_boardType         = (int) GetUChar(pBuffer);
       m_boardIdent        = GetLongLong(pBuffer);
       m_operatingFreq     = GetFloat(pBuffer);
       m_FPGAerror         = (int) GetUChar(pBuffer);
       m_calBit            = GetBOOL(pBuffer);
       m_txPower           = (int) GetUChar(pBuffer);
       m_txCycles          = (int) GetShort(pBuffer);
```

```
        m_rxSamples         = (unsigned int) GetUShort(pBuffer);
        m_rxPeriod          = GetFloat(pBuffer);
        m_sidescanAdcEnable = (int) GetUChar(pBuffer);
        m_timeSecPC         = (int) GetInt(pBuffer);
        m_timeMsecPC        = (int) GetUShort(pBuffer);
        m_timeSecSonar      = (int) GetUInt(pBuffer);
        m_timeMsecSonar     = (int) GetUShort(pBuffer);
        m_firstInScan       = (int) GetUChar(pBuffer);
        m_PPS               = GetUChar(pBuffer);
        m_pingMode          = (int) GetUChar(pBuffer);

        m_triggerFlags      = GetUShort(pBuffer);
        m_pgagain           = GetUShort(pBuffer);
        m_lnagain           = GetUShort(pBuffer);
        m_basegain          = GetUShort(pBuffer);
        m_lingain           = GetUShort(pBuffer);
        m_sqgain            = GetUShort(pBuffer);
        m_swgain            = GetFloat(pBuffer);
        m_RxDecimation      = GetUShort(pBuffer);
        m_rxBandwidth       = GetFloat(pBuffer);
        m_preampPowerOn     = GetBool(pBuffer);
        m_sampleType        = (PingHeader::eSampleType) *pBuffer++;
} // 73 bytes

inline unsigned char GetUChar(unsigned char *&pBuffer)
{unsigned char value = *((unsigned char*) pBuffer);
pBuffer+= sizeof(unsigned char); return value;}
inline unsigned short GetUShort(unsigned char *&pBuffer)
       {unsigned short value = *((unsigned short*) pBuffer);
       pBuffer+= sizeof(unsigned short); return value;}
inline short  GetShort(unsigned char *&pBuffer)
       {short value = *((short*) pBuffer);
       pBuffer+= sizeof(short); return value;}
inline unsigned int GetUInt(unsigned char *&pBuffer)
       {unsigned int value = *((unsigned int*) pBuffer);
       pBuffer+= sizeof(unsigned int); return value;}
inline int GetInt(unsigned char *&pBuffer)
       {int value = *((int*) pBuffer);
       pBuffer+= sizeof(int); return value;}
inline long long GetLongLong(unsigned char *&pBuffer)
       {long long value = *((long long*) pBuffer);
       pBuffer+= sizeof(long long); return value;}
inline float  GetFloat(unsigned char *&pBuffer)
       {float value = *((float*) pBuffer);
       Buffer+= sizeof(float); return value;}
inline double GetDouble(unsigned char *&pBuffer)
       {double value = *((double*) pBuffer);
       pBuffer+= sizeof(double); return value;}
```

```
inline bool GetBool(unsigned char *&pBuffer)
       {unsigned char value = GetUChar(pBuffer);
       return (value = 0 ? false : true); pBuffer++;}
inline BOOL GetBOOL(unsigned char *&pBuffer)
   {BOOL value = GetUChar(pBuffer);
   return (value = 0 ? FALSE : TRUE); pBuffer++;}
```

The type of data samples is one of several types, according to m_sampleType in the header.

```
       switch (m_header.m_sampleType)
       {
       case PingHeader::eFormatPhaseDiff:
              memcpy(m_samples, pBuffer+samplesOffset,
       m_header.m_rxSamples * sizeof(BathySample));
              break;
       case PingHeader::eFormat4Phase:
              {
                     unsigned char* buffPos = pBuffer+samplesOffset;
                     for (unsigned int i = 0; i < m_header.m_rxSamples; i++)
                     {
                            m_4phSamples[i].ReadBuffer(buffPos);
                            buffPos += BathySample4Ph_SIZE;
                     }
              }
              break;
       case PingHeader::eFormatIQ:
              {
                     unsigned char* buffPos = pBuffer+samplesOffset;
                     for (unsigned int i = 0; i < m_header.m_rxSamples; i++)
                     {
                            m_IQsamples[i].ReadBuffer(buffPos);
                            buffPos += BathySampleIQ_SIZE;
                     }
              }
              break;
       }
```

```
void BathySample4Ph::ReadBuffer(unsigned char* pBuffer)
{
       READ_SHORT(pBuffer, m_sampNo);
       READ_SHORT(pBuffer, m_phaseA);
       READ_SHORT(pBuffer, m_phaseB);
       READ_SHORT(pBuffer, m_phaseC);
       READ_SHORT(pBuffer, m_phaseD);
```

```
        READ_SHORT(pBuffer, m_amp);
}

// Read from memory
void BathySampleIQ::ReadBuffer(unsigned char* pBuffer)
{
        READ_SHORT(pBuffer, m_sampNo);
        READ_FLOAT(pBuffer, m_a_r);
        READ_FLOAT(pBuffer, m_a_i);
        READ_FLOAT(pBuffer, m_b_r);
        READ_FLOAT(pBuffer, m_b_i);
        READ_FLOAT(pBuffer, m_c_r);
        READ_FLOAT(pBuffer, m_c_i);
        READ_FLOAT(pBuffer, m_d_r);
        READ_FLOAT(pBuffer, m_d_i);
}
```

Annex A - Previous Document Change Record

The previous change history of this document is as follows.

See the top of this document for recent change history.

Issue[2]	Date	Reason for Change and Issue
0.1		Original
2.0		Changes to the format to include timestamps on strings, as well as some minor typographical corrections
3.0	21 May 1998	Adds the file header record
3.2	18 August 1998	Adds the xyza format
3.3	25 August 1998	Additions to xyza format
3.4	26 August 1998	Config and processed data file information added
3.5	06 April 1999	cXYZAPoint corrected in line with released code
3.6	27 April 1999	Note added that file header may be absent
3.7	29 April 1999	Array sizes defined
3.8	05 May 1999	Clarification of easting and northing
3.9	07 May 1999	Name changed of samp counts in txer data
3.91	03 June 1999	File header length corrected
4.0	19 December 2003	Changed to SEA SWATHplus format
4.1	18 February 2004	Clarified some sign conventions
4.2	21 October 2005	Added phase offset notes
5 A	July 2006	Formal SEA format
5 B	November 2006	Version 3 software data format added
5 C	November 2006	Aux ports documented. Other data block types identified.
5 D	December 2006	TCP/IP and Com Port Command and Status command formats clarified.
5 E	February 2007	Added Parsed Data format
5 F	June 2007	Corrections to interpretation notes for angle in the parsed data format
5 G	July 2007	SXI filtering information
5 K	December 2007	Parsed Data Filter information updated
5L	March 2008	Format of MRUT_DATA clarified
5M	May 2008	CMS_CMD modified to add new filter settings
5N	September 2009	Updates to content of Processed ping data.
6A	January 2010	New format, updated appearance for clarity
6B	January 2010	Corrected error in description of old SXP format. Note on class memory images added.
6C	February 2010	Added extra filter controls in CMS_CMD

[2] Early versions used Submetrix issue format