BATHYMETRIC DATA MESSAGES

FILE FORMAT DESCRIPTION

0014932_REV_A

EdgeTech
4 Little Brook Road
West Wareham, MA 02576

Tel: (508) 291-0057
Fax: (508) 291-2491
www.EdgeTech.com

ABOUT THIS DOCUMENT

This document is to be used in conjunction with the **Bathymetric Data Messages** and formats described in the EdgeTech-supplied header file, **BathymetricMessages.h**. These messages are C format header files and are presented in *APPENDIX B: BATHYMETRIC FILE HEADER* of this document. In addition to these bathymetric messages, the side scan data format and other messages are described in the header file **JSFdefs.h** and the **JSF File Format Description** (PN 0004824).

CUSTOMER SERVICE

Customer service personnel at EdgeTech are always eager to hear from users of our products. Your feedback is welcome, and is a valuable source of information which we use to continually improve these products. Therefore we encourage you to contact EdgeTech Customer Service to offer any suggestions or to request technical support:

E-mail: service@edgetech.com

Mail: 4 Little Brook Road
West Wareham, MA 02576

Telephone: (508) 291-0057

Facsimile: (508) 291-2491

24-Hour Emergency Technical Support Line: (508) 942-8043

For more information please go to www.EdgeTech.com.

COMPANY BACKGROUND

EdgeTech (formerly EG&G Marine Instruments) traces its history in underwater data acquisition, and processing back to 1966. EdgeTech has designed, developed and manufactured products, instruments, and systems for the acquisition of underwater data, including marine, estuarine, and coastal applications for over 45 years.

The company has responded to the needs of the scientific, Naval and offshore communities by providing equipment, such as sub-bottom profilers, side scan sonar, acoustic releases, USBL positioning systems, and bathymetric systems that have become standards in the industry.

EdgeTech has also consistently anticipated and responded to future needs through an active research and development program. Current efforts are focused on the application of cutting edge CHIRP and acoustic technology.

TABLE OF CONTENTS

SECTION 1: OVERVIEW

The **Bathymetric Data Messages** consist of five 300X messages used for the time stamped bathymetric data and supporting information such as position, motion, heading, etc. (see header file in *APPENDIX B: BATHYMETRIC FILE HEADER*). The time stamped bathymetric data can be in one of two forms: (1) interferometric or (2) pseudo multibeam (binned/decimated).

This document describes the 300X family **Bathymetric Data Messages** and provides a definition of the essential calculations used by the bathymetry system. The following sections provide a description for each of the 5 **Bathymetric Data Messages** sent from EdgeTech's Discover Bathymetric Software.

SECTION 2: BathymetricDataMessageType

Message ID: 3000

BathymetricDataMessageType is a source for bathymetry data.

This is the primary message sent from the Bathymetry System. For each ping, there is one message for the port side, and one for the starboard side. This message contains the **time_delay**, **angle** and **amplitude** of each assumed seafloor echo. Multiple messages of this type are interspersed throughout the data file or data stream. This message consists of a header, followed by a number of bathymetric samples (**numberOfSamples**), one corresponding to each sounding point.

2.1 Header Description

Use these data to derive the RAW **x** and **z** data samples prior to any motion correction. The header contains information static to each ping as follows:

DATA FIELD	BYTES	UNITS	USAGE
Time Since 1/1/1970	0-3	s	Essential
Nanosecond Supplement to Time	4-7	ns	Essential
Ping Number	8-11		Essential
Number of BathymetricSampleType Entries	12-13		Essential
Channel	14		
Algorithm Type	15		
Number of Pulses	16		
Pulse Phase	17		
Pulse Length	18-19	ms	
Transmit Pulse Amplitude	20-23		
Chirp Start Frequency	24-27	Hz	
Chirp End Frequency	28-31	Hz	
Mixer Frequency	32-35	Hz	
Sample Rate	36-39	Hz	
Offset to First Sample	40-43	ns	Essential
Time Delay Uncertainty	44-47	s	Essential
Time Scale Factor	48-51	s	Essential
Time Scale Accuracy	52-55	percent	Essential
Angle Scale Factor	56-59	degrees	Essential
Reserved	60-63		
Time to First Bottom Return	64-67	ns	Essential
Format Revision Level	68		Essential
Binning Flag	69		Essential

DATA FIELD	BYTES	UNITS	USAGE
TVG	70	dB/100m	
Reserved	71		
Span	72-75	m/deg	
Reserved	76-79		

Table 2-1: Header Description

NOTE 1: *As of 5/12/2014 bytes 18-19, 44-47, 60-63, and 70 have been modified.*

NOTE 2: *Nonessential entries can be safely disregarded.*

2.2 Important Data Fields

The following data fields and their explanations are important for understanding the bathymetric format.

2.2.1 Time Delay Uncertainty Estimate

The **Time Delay Uncertainty Estimate** (bytes 44-47) is the potential acoustic uncertainty of the true delay to each detected echo. This field is used to compute the range uncertainty, in meters, for each sample in the data packet.

2.2.2 Format Revision Level

The **Format Revision Level** (byte 68) may have a value between 0 and 4. The revision number affects the interpretation of the data fields listed above, as well as the N x 4 sample sets that follow and includes the additional information for binned data. Revisions 0 through 2 only provide information for raw interferometry data only, whereas Revision 3 and 4 supports the raw interferometry and pseudo multibeam data formats.

The latest **Format Revision Level** is described in sub-section *2.3*, whereas the older formats have been provided in the *APPENDIX A: LEGACY FILE FORMATS* for completeness.

2.2.3 Binning Flag

The **Binning Flag** (byte 69) specifies the type of binning output and may have a value between 0 and 2. A value of 0 indicates that no binning has been carried out and the data output is pure interferometric, a value of 1 indicates the data have been decimated based on a user defined equi-distant across track bin size to produce multibeam-like data, and a value of 2 indicates that the data have been decimated based on a user defined equi-angular beam size to produce an alternate form of multibeam-like data.

When this binning process is carried out, the data are filtered (or cleaned of outliers) as much as possible prior to binning so that each local estimate is not corrupted by surface or wake artifacts. A median estimate, as opposed to an average, is also used to reduce the effects of outliers on the local estimates.

2.2.4 Span

Span (bytes 72-75) describes the total across track extent of binned data per side that will be returned. The port and starboard record spans are half that of the complete span, and should be added to get the final data set (i.e. if **Span** = 250, then the final data set should have 500). This parameter can be specified in meters or in degrees and depends on the binning type selected in byte 69.

The correlation between **Binning** and **Span** is defined below.

If **Binning** = 0, then **Span** =	Maximum processing range defined in the bathymetric processing parameters in meters.
If **Binning** = 1, then **Span** =	No. of bins x bin size (in meters).
If **Binning** = 2, then **Span** =	No. of beams x beam size (in degrees).

Therefore, the final data set would be computed as **Span** x 2;

2.3 Format Revision Level = 4 (latest)

Following the **BathymetricDataMessageType** header (or 3000 message header) are N sample sets (N being derived from bytes 12 – 13, appropriately labeled **numberOfSamples**) with each having the 16-bit integer fields described below. The third set of 16 bits have been broken down into the amplitude and angle uncertainty fields, whereas the last 16 bits have been broken down into the flag, SNR, and quality components.

time_delay	16 bits, unsigned
angle	16 bits, signed
amplitude	8 bits, unsigned
angle uncertainty	8 bits, unsigned
flag	8 bits, unsigned
SNR	5 bits, unsigned
quality	3 bits, unsigned

CAUTION!

Use the angle as a signed value! Please refer to sub-section 2.3.1.4 for further information.

2.3.1 Essential Calculations / Definitions

The following calculations and definitions are essential to understanding and using the bathymetric message format:

2.3.1.1 Echo Time

The Echo Time is the total time it takes to receive an echo from the seafloor, in seconds:

$$EchoTime = \left(\frac{timeToFirstSample}{1 \times 10^9} + (time\_delay \times timeScaleFactor)\right) \quad \text{(Eq.1)}$$

2.3.1.2 Slant Range to Echo

The Slant Range to Echo is the slant range to each sample, in meters:

$$Slant\ Range = \left(\frac{soundVelocity}{2}\right) \times EchoTime \quad \text{(Eq.2)}$$

where *soundVelocity* must be in meters per second and acquired from one of the other supporting bathymetric messages in the 300X family.

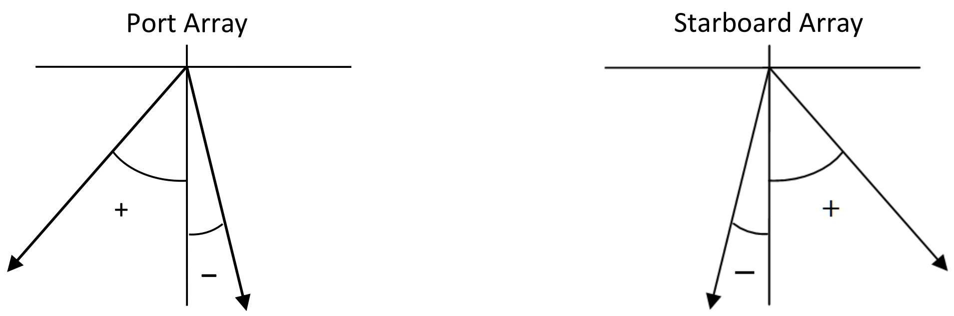
2.3.1.3 Range Uncertainty Estimate

The Range Uncertainty Estimate is the potential acoustic uncertainty of the range for each sample in the data packet, in meters.

$$\sigma_R = \left(\frac{soundVelocity}{2}\right) \times timeDelayUncertainty \qquad \text{(Eq.3)}$$

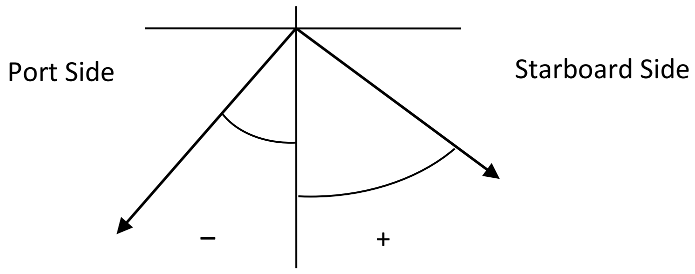
2.3.1.4 Angle from Nadir for Each Sample

Angles, reported in degrees, are measured positive from Nadir and increase outwards from both port and starboard. If data for a channel, such as port side, falls behind Nadir, then these correspond to negative angles.



When both port and starboard data sets are computed and rendered together in the same frame of reference, negative angles are plotted on the port side and positive angles are plotted on the starboard side and as defined by the equation below. {*Channel* = 0 for port, 1 for stbd}:

$$AngleFromNadir = (-1)^{(Channel+1)} \times angle \times angleScale\ Factor \qquad \text{(Eq. 4)}$$



2.3.1.5 Amplitude

Amplitude is a fundamental attribute that is used to trim invalid data points from the final data set and primarily excludes weak echoes, such as the water column or very weak backscatter, based on some minimum threshold. Typically good seafloor echoes are above 25 dB to 30 dB depending on bottom type. Values less than 20 dB are typically not bottom echoes.

The data points measured by the system can have an amplitude value between 0 – 127.5dB and is reported in 0.5dB increments.

2.3.1.6 Angle Uncertainty Estimate

The Angle Uncertainty Estimate can vary between 0 and 5.1 degrees and is reported in 0.02 degree increments. Any angle uncertainty larger than 5.1 degrees is clamped to 5.1 degrees.

2.3.1.7 Flag Interpretation

These flags are used for data cleaning and if set to 1 indicate the data which have been deemed as invalid points by the processing algorithm. A description of these flags is listed below.

Bit 0: *Outlier Removal Flag* – if set the processing algorithm deems these data points as having excessive deviations from the norm.

Bit 1: *Water Column Flag* – if set the processing algorithm deems these data points as water column and are not used in determining the seafloor estimates.

Bit 2: *Amplitude* – if set the processing algorithm deems these data points as invalid based on the calculated threshold. This fundamental attribute is used primarily to exclude weak echo points, such as the water column returns and very weak backscatter amplitudes.

Bit 3: *Quality* – if set the processing algorithm deems these data points as invalid based on angle uncertainty. This filter is used to eliminate points whose phase differences are greater than some specified tolerance.

Bit 4: *SNR* – if set the processing algorithm deems these data points as invalid based on the SNR. This filter is very useful in trimming data points where the angle estimation quality is low due to noise and multipath effects.

Bit 5: *Null Content Binned Data* – if set the binned data contains null content and should be excluded from all processing (i.e. if the total across track extent is too large for the depth, then some bins may be empty and will be deemed as null). This bit is <u>only</u> valid when the data are binned.

Bit 6 - 7: *Reserved*

2.3.1.8 SNR

SNR is a very useful statistic which is used to trim invalid data points where the angle noise is high due to multipath effects. When estimating the primary (largest amplitude) angle of arrival the process also returns an estimate of the power in this angle. This power is compared to the total power in the signal at that instant and so a Signal (primary echo) to Noise (noise plus all multipath echoes) can be estimated. In practice the true noise component of this is very small over the useful range of the bathymetry data and the "Noise" component is almost all due to multipath interference echoes.

SNR values greater than 20dB are excellent in terms of angle estimation quality while less than 10dB are quite poor. Useful thresholds are between 10-20 dB, depending on the desire to have maximum swath (more noise) or lower noise and narrower swath widths.

This metric is described by a 5 bit value that ranges between 0 and 31dB in 1dB increments. Any value higher than 31dB will be limited to 31dB.

2.3.1.9 Quality Factor

Quality Factor is a metric used to identify how well the interstave phase measurements agree. The array used to determine the angles has 10 x ½ wavelength spaced elements. This allows the estimation of 9 interstave phase estimates. In the most ideal case (no errors, no noise, no interfering multipath) these would all agree. In practice this is not the case and interstave phases may either agree quite well (+/- 5 – 10 degrees) or not at all (up to +/- 90 deg). This metric is described by a 3 bit value and has been broken down into 8 discrete numbers associated with each:

0 – quality < 50%
1 – 50% <= quality < 60%
2 – 60% <= quality < 70%
3 – 70% <= quality < 75%
4 – 75% <= quality < 80%
5 – 80% <= quality < 85%
6 – 85% <= quality < 90%
7 – 90% <= quality

For most cases, any data with a Quality Factor less than 50% should be discarded. High quality data is considered to be anything above 70-80%. The quality factor can be set quite high (90%) in most cases, especially when the sea floor is very flat.

> **NOTE:** *Even when the data is very good in the SNR sense, the Quality Factor can be lower than one would expect, i.e. when the seafloor has a slope that approximates equidistance from the array (such as the beginnings of a sloping bank). In this case, the angles are hard to estimate as the angles present in a range bin can be spread over a wide span. In the limit where the echoes are completely normal to the surface over a wide span of angles, very low Quality Factors can result.*

2.3.1.10 Nadir Depth (or Depth below Sounder)

The system incorporates a single beam echo sounder mode with a 15 degree beam width. This depth is reported for each ping and is independent of any interferometric calculations or principles. The value is specified in meters and is defined as:

$$DepthBelowSounder = \left(\frac{soundVelocity}{2}\right) \times timeToFirstBottomreturn \qquad \text{(Eq.5)}$$

where *soundVelocity* must be in meters per second and acquired from one of the other supporting bathymetric messages in the 300X family.

This parameter is used by the system as a *Water Column Filter* to flag all data points whose slant range lies between 0 and the (*Depth below Sounder* – 1m). This metric is effective at trimming water column data but can also trim data from close up targets (or targets located at less slant range than the bottom) such as dock walls, sharp banks, etc., so caution should be taken when surveying next to steep slopes.

2.3.1.11 Seafloor Samples

The seafloor samples are computed as raw *X* and *Z* values, reported in meters, and are calculated as:

$$x = SlantRange \times \sin(AngleFromNadir) \qquad \text{(Eq.6)}$$

$$z = SlantRange \times \cos(AngleFromNadir) \qquad \text{(Eq.7)}$$

Once these data points are rendered, a median filter is run over the data to exclude/flag data points whose *Z* values deviate significantly from the local median value. This filter, known as the *Outlier Removal Filter*, is effective when equidistant across track binning is to be used. However, it is <u>not</u> used when equiangle binning is implemented as it removes small vertical targets.

2.3.1.12 Binning

Currently there are two types of binning schemes the processor can utilize:

The first scheme uses equidistant sections and does a good job at plotting most seafloor topographies. The user must specify the number of bins, bin size (in meters), and total swath (in meters) the user wants to achieve. This relationship is defined below.

$$TotalSwath = NumberOfBins \times BinSize \quad \text{(Eq.8)}$$

This method, however, cannot plot vertical objects very well, as it inherently collapses the data within the bin.

The second scheme uses equiangular sections and is very effective at plotting vertical objects. The user can only specify how many beams and a beam size, as the total field of view is set to a constant 200 deg. This relationship is described below.

$$200^{\circ} = NumberOfBeams \times BeamSize \quad \text{(Eq.9)}$$

This method, however, does not provide an evenly distributed data set like the equidistant method described above. Instead it provides fewer soundings as the angle increases from nadir, much like a traditional MBES.

SECTION 3: AttitudeMessageType

Message ID: 3001

AttitudeMessageType is a source for roll, pitch, heave, and heading data. Roll, pitch, and heading are specified in degrees, whereas heave is given in meters. The convention is: roll port up is positive, pitch bow up is positive, heave down is positive, and yaw to starboard is positive.

This 3001 message has a field for *Heading, Heave, Pitch,* and *Roll*. *Yaw* is not used. Some or all of these fields may be valid (set to 1) depending on which type(s) of sensor is (are) used.

The validity of each field is indicated in the *Data Valid Flags* fields and it is imperative that this is used to correctly parse the fields. Which fields are populated depend on the device used in conjunction with the sonar system supplying the data.

Since each message has a single, unique timestamp, only the data supplied from the incorporated device in one input string (or message) will be in this message. Unused fields will be flagged as such (set to 0) in the *Data Valid Flags*.

The interpretation of the fields in this message is self-explanatory.

The time stamp accuracy of this message with respect to the sonar ping emission time will be approximately 1 *ms* at 80% and 2 *ms* at 100% of the samples.

3.1 Example 1

Data Source	=	TSS CMS
Data Supplied	=	Roll, Pitch, and Heave
Unused fields	=	Yaw and Heading

Data Valid Flags:

Bit 0: heading	=	0
Bit 1: heave	=	1
Bit 2: pitch	=	1
Bit 3: roll	=	1
Bit 4: yaw	=	0

3.2 Example 2

Data Source	=	OCTANS
Data Supplied	=	Roll, Pitch, Heave, and Heading
Unused Fields	=	Yaw

Data Valid Flags:

Bit 0: heading	=	1
Bit 1: heave	=	1
Bit 2: pitch	=	1
Bit 3: roll	=	1
Bit 4: yaw	=	0

SECTION 4: PressureMessageType

Message ID: 3002

PressureMessageType is a source for sound velocity (in meters per second), and possibly depth (in meters).

Only on those platforms that are deployed subsea will have valid depth data as provided by the platform's depth sensor (such as ROV, ROTV and AUV applications).

The only fields which are valid in this structure are the time stamp and the sound velocity (always), and potentially depth.

This is the sound velocity at the sonar head and must be used when calculating *Slant Range* in Message ID 3000, or **BathymetricDataMessageType**, above.

The time stamp accuracy is 20 *ms* or better.

The validity of each field is determined as in *SECTION 3: AttitudeMessageType*. Unused fields will be flagged as such (set to 0) in the *Data Valid Flags*.

SECTION 5: AltitudeMessageType

Message ID: 3003

AltitudeMessageType is a source for altitude (in meters), speed (in knots), and heading (in degrees).

This *altitude* parameter is reported for each ping and is the value computed from the *Depth Below Sounder* (*Eq.5*). This field will always be valid and should be added to the depth field (if available) from Message ID 3002, or **PressureMessageType**, in order to calculate the total water depth.

For *speed* and *heading*, the *Data Valid Flags* should be tested to determine if these fields are usable (set to 1). The validity of these fields depends on what devices are connected to the sonar system. For example, this message would contain *heading* if there is a device connected to the sonar which supplies heading only (such as a gyroscope).

Again, the validity of each field is determined as in *SECTION 3: AttitudeMessageType*. Unused fields will be flagged as such (set to 0) in the *Data Valid Flags*.

SECTION 6: PositionMessageType

Message ID: 3004

PositionMessageType is a source for position (lat/long), heading, speed, and antenna altitude. Latitude, longitude, and heading are given in degrees, speed in knots, and antenna height in meters.

The *Data Valid Flags* must be tested for the presence or absence of data in each field. If a GPS device is connected and it is a dual antenna system supplying *heading*, then this *heading* field will also be valid (set to 1).

Only *latitude*, *longitude*, *heading*, and *speed* are likely to be valid in this message. There is the potential for antenna height to be populated as well, but is not always present. Therefore, the validity of each field is determined as in *SECTION 3: AttitudeMessageType*. Unused fields will be flagged as such (set to 0) in the *Data Valid Flags*.

APPENDIX A: LEGACY FILE FORMATS

There are several revisions of the **Bathymetric Data Messages** that includes the header description and essential calculations for bathymetric data (**Message ID**: 3000). The revision number affects the header description, interpretation of the N x 4 sample sets, and whether or not the additional information for binned data is present. Revisions 0 through 2 only provide information for raw interferometry data only, whereas Revision 3 and 4 support the raw interferometry and pseudo multibeam data formats. The legacy file formats (0-3) that exist are listed here for completeness.

A.1 Format Revision Level = 0

A description of the header as well as how to interpret the N sample sets (N being derived from bytes 12-13 appropriately labeled **numberOfSamples** in the header file) for this file format revision (Revision Level = 0) are provided below.

A.1.1 Header Description

Use these data to derive the RAW **x** and **z** data samples prior to any motion correction. The header for File Format Revision level 0 contains information static to each ping as follows:

DATA FIELD	BYTES	UNITS	USAGE
Time Since 1/1/1970	0-3	s	Essential
Nanosecond Supplement to Time	4-7	ns	Essential
Ping Number	8-11		Essential
Number of BathymetricSampleType Entries	12-13		Essential
Channel	14		
Algorithm Type	15		
Number of Pulses	16		
Pulse Phase	17		
Pulse ID	18-19		
Transmit Pulse Amplitude	20-23		
Chirp Start Frequency	24-27	Hz	
Chirp End Frequency	28-31	Hz	
Mixer Frequency	32-35	Hz	
Sample Rate	36-39	Hz	
Offset to First Sample	40-43	ns	Essential
Start Time Accuracy	44-47	s	Essential
Time Scale Factor	48-51	s	Essential
Time Scale Accuracy	52-55	percent	Essential
Angle Scale Factor	56-59	degrees	Essential

DATA FIELD	BYTES	UNITS	USAGE
Amplitude Scale Factor	60-63	counts	
Time to First Bottom Return	64-67	ns	Essential
Format Revision Level	68		Essential
Reserved	69-79		

Table 6-1: Header Description

NOTE 1: *This header information is valid for all JSF files created prior to May 2013.*

NOTE 2: *Nonessential entries can be safely disregarded.*

In addition, the data that follows the Header contains N sample sets (N being derived from bytes 12-13 appropriately labeled *numberofSamples* in the header file), each having the four 16 bit integer fields listed below:

time_delay	16 bits, unsigned
angle	16 bits, signed
amplitude	16 bits, unsigned
quality	16 bits, unsigned

CAUTION!

Use the angle as a signed value! Please refer to sub-section 2.3.1.4 for further information.

A.1.2 Essential Calculations/Definitions

The original file format also included the essential calculations and definitions provided in Equations 10 through 17. These are described in sub-sections *A.1.2.1* and *A.1.2.7*.

A.1.2.1 Echo Time

The Echo Time is the total time it takes to receive an echo from the seafloor, in seconds:

$$EchoTime = \left(\frac{timeToFirstSample}{1 \times 10^9} + (time\_delay \times timeScaleFactor) \right) \quad \text{(Eq.10)}$$

A.1.2.2 Slant Range to Echo

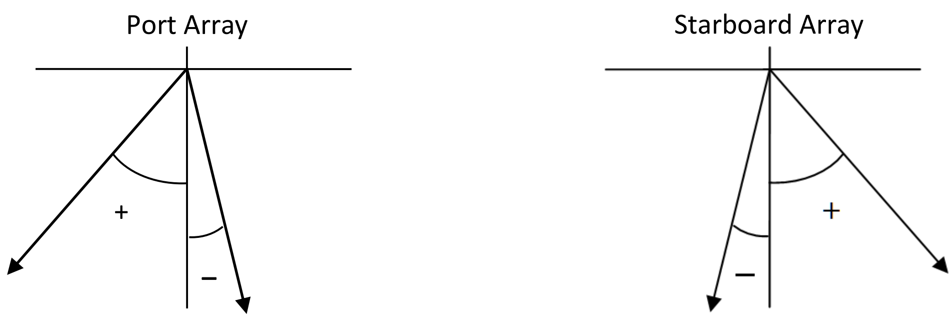
The slant range to echo is the slant range to each sample, in meters:

$$Slant\ Range = \left(\frac{soundVelocity}{2} \right) \times EchoTime \quad \text{(Eq.11)}$$

where *soundVelocity* must be in meters per second and acquired from one of the other supporting bathymetric messages in the 300X family.

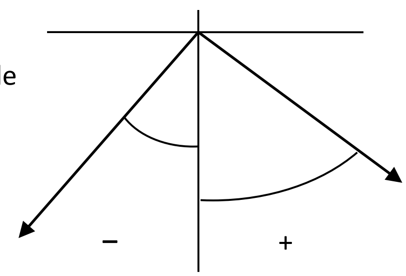
A.1.2.3 Angle from Nadir for Each Sample

Angles, reported in degrees, are measured positive from Nadir and increase outwards from both port and starboard. If data for a channel, such as port side, falls behind Nadir, then these correspond to negative angles.



When both port and starboard data sets are computed and rendered together in the same frame of reference, negative angles are plotted on the port side and positive angles are plotted on the starboard side and as defined by the equation below. {*Channel* = 0 for port, 1 for stbd}:

$$AngleFromNadir = (-1)^{(Channel+1)} \times angle \times angleScale\ Factor \qquad \text{(Eq. 12)}$$



A.1.2.4 Amplitude

The data points measured by the system can have an amplitude value between 0 – 2^{15} and is computed as:

$$Amplitude = amplitude \times amplitudeScaleFactor \qquad \text{(Eq.13)}$$

> **NOTE:** *In Revision 4, the amplitude field is now reported in dB and no longer needs to be calculated as described by Equation 13.*

A.1.2.5 Quality Factor

The Quality Factor is reported within a range from 0-100% and is calculated as in Equation 14.

$$QF = \left(\frac{quality}{2^{16}}\right) \times 100 \qquad \text{(Eq.14)}$$

For most cases, any data with a Quality Factor less than 50% should be discarded. High quality data is considered to be anything above 70-80%. The quality factor can be set quite high (90%) in most cases, especially when the sea floor is very flat.

> **NOTE:** *This field has been modified several times throughout the revisions and must be treated differently depending on the file format revision being used.*

A.1.2.6 Nadir Depth (or Depth below Sounder)

The system incorporates a single beam echo sounder mode with a 15 degree beam width. This depth is reported for each ping and is independent of any interferometric calculations or principles. The value is specified in meters and is defined as:

$$DepthBelowSounder = \left(\frac{soundVelocity}{2}\right) \times timeToFirstBottomreturn \qquad \text{(Eq.15)}$$

where *soundVelocity* must be in meters per second and acquired from one of the other supporting bathymetric messages in the 300X family.

A.1.2.7 Seafloor Samples

The seafloor samples are computed as raw *X* and *Z* values, reported in meters, and are calculated as:

$$x = SlantRange \times sin(AngleFromNadir) \qquad \text{(Eq.16)}$$

$$z = SlantRange \times cos(AngleFromNadir) \qquad \text{(Eq.17)}$$

A.2 Format Revision Level = 1

Format Revision Level 1 is similar to Format Revision Level 0 accept that the last 16 bits have been broken down into the flag, and quality components as shown below.

time_delay	16 bits, unsigned
angle	16 bits, signed
amplitude	16 bits, unsigned
flag	8 bits, unsigned
quality	8 bits, unsigned

CAUTION!

Use the angle as a signed value! Please refer to sub-section 2.3.1.4 for further information.

A.2.1 Flag Interpretation

The introduction of the flag bits allowed the processor to use a filtering scheme for the interferometric data. These flags, if set to 1, indicate the data points which have been deemed as "not part of valid seafloor returns" by the processing algorithm, but are included here for completeness.

Bit 0:	*Water Column Flag* – if set, the processing algorithm deems these data as water column and are not used.
Bit 1:	*Outlier Removal Flag* – if set, the processing algorithm deems these data points as not valid seafloor returns.
Bit 2 – 7:	*Reserved.*

A.2.2 Other Essential Calculations/Definitions

All other essential calculations are identical to the ones provided in sub-section *0*.

A.3 Format Revision Level = 2

Format Revision Level 2 is similar to Format Revision Level 1 accept that the last 16 bits have been broken down into the flag, SNR, and quality components as shown below.

time_delay	16 bits, unsigned
angle	16 bits, **signed**
amplitude	16 bits, unsigned
flag	8 bits, unsigned
SNR	5 bits, unsigned
quality	3 bits, unsigned

CAUTION!

Use the angle as a signed value! Please refer to sub-section 2.3.1.4 for further information.

There were also some new additions to the flag interpretation and a new definition provided for the Quality and SNR fields. These new essential calculations are provided below.

A.3.1 Flag Interpretation

These flags, if set to 1; indicate the data which have been deemed as invalid points by the processing algorithm. A description of these flags is listed below.

Bit 0: *Outlier Removal Flag* – if set, the processing algorithm deems these data points as not valid seafloor returns.

Bit 1: *Water Column Flag* – if set, the processing algorithm deems these data as water column and are not used.

Bit 2: *Amplitude* – if set, the processing algorithm deems data as invalid based on some user defined threshold. This fundamental attribute is used primarily to exclude weak echo points such as the water column and very weak backscatter amplitudes.

Bit 3: *Quality* – if set, the processing algorithm deems data as invalid based on some user defined parameter. This filter is used to eliminate points whose phase differences are greater than some specified tolerance.

Bit 4: *SNR* – if set, the processing algorithm deems data as invalid based on some user defined value. This filter is very useful in trimming data points where the angle noise is high due to noise and multipath effects.

Bit 5 – 7: *Reserved*

A.3.2 Quality Factor

The quality factor is now defined as a three bit value and has been broken up into eight discrete numbers as described below.

0 – quality < 50%
1 – 50% <= quality < 60%
2 – 60% <= quality < 70%
3 – 70% <= quality < 75%
4 – 75% <= quality < 80%
5 – 80% <= quality < 85%
6 – 85% <= quality < 90%
7 – 90% <= quality

A.3.3 SNR

SNR is a 5 bit value that ranges between 0 and 31 dB in 1dB increments. Any value below 0 will be clamped to 0dB and any value higher than 31dB will be clamped to 31dB.

A.3.4 Other Essential Calculations/Definitions

All other essential calculations are identical to the ones provided in sub-section *0*.

A.4 Format Revision Level = 3

Format Revision Level 3 is similar to Format Revision Level 2 accept with the addition of the processors capability to output pseudo-multibeam (or binned/decimated) data. The header file and *Flag Interpretation* fields were modified to include the necessary information for this new output

A.4.1 Header Description

Use these data to derive the RAW **x** and **z** data samples prior to any motion correction. The header contains information static to each ping as follows:

DATA FIELD	BYTES	UNITS	USAGE
Time Since 1/1/1970	0-3	s	Essential
Nanosecond Supplement to Time	4-7	ns	Essential
Ping Number	8-11		Essential
Number of BathymetricSampleType Entries	12-13		Essential
Channel	14		
Algorithm Type	15		
Number of Pulses	16		
Pulse Phase	17		
Pulse ID	18-19		
Transmit Pulse Amplitude	20-23		
Chirp Start Frequency	24-27	Hz	
Chirp End Frequency	28-31	Hz	
Mixer Frequency	32-35	Hz	
Sample Rate	36-39	Hz	
Offset to First Sample	40-43	ns	Essential
Start Time Accuracy	44-47	s	Essential
Time Scale Factor	48-51	s	Essential
Time Scale Accuracy	52-55	percent	Essential
Angle Scale Factor	56-59	degrees	Essential
Amplitude Scale Factor	60-63	counts	Essential
Time to First Bottom Return	64-67	ns	Essential
Format Revision Level	68		Essential
Binning Flag	69		Essential
Reserved	70-71		
Span	72-75	m/deg	
Reserved	76-79		

Table 6-2: Header Description

> **NOTE 1:** *Bytes 69 and 72-75 have been modified. This header is valid for JSF files created between May 2013 and May 2014.*
>
> **NOTE 2:** *Nonessential entries can be safely disregarded*

A.4.2 Flag Interpretation

These flags, if set to 1; indicate the data which have been deemed as invalid by the processing algorithm. A description of these flags is listed below.

Bit 0: *Outlier Removal Flag* – if set, the processing algorithm deems these data points as not valid seafloor returns.

Bit 1: *Water Column Flag* – if set, the processing algorithm deems these data as water column and are not used.

Bit 2: *Amplitude* – if set, the processing algorithm deems data as invalid based on some user defined threshold. This fundamental attribute is used primarily to exclude weak echo points such as the water column and very weak backscatter amplitudes.

Bit 3: *Quality* – if set, the processing algorithm deems data as invalid based on some user defined parameter. This filter is used to eliminate points whose phase differences are greater than some specified tolerance.

Bit 4: *SNR* – if set, the processing algorithm deems data as invalid based on some user defined value. This filter is very useful in trimming data points where the angle noise is high due to noise and multipath effects.

Bit 5: *Null Content Binned Data* – if set, the binned data contains null content and should be excluded from all processing (i.e. if the total across track extent is too large for the depth, then some bins may be empty and will be deemed as null). This bit is only valid if data is binned.

Bit 6 - 7: *Reserved*

A.4.3 Other Essential Calculations/Definitions

The Quality and SNR fields are identical to the definitions provided in sub-sections *A.3.2* and *A.3.3* and all other essential calculations are identical to the ones provided in sub-section *0*.

APPENDIX B: BATHYMETRIC FILE HEADER

```
/* ------------------------------------------------------------------------- */
/* BathymetricMessages.h                                                     */
/*                                                                           */
/* ------------------------------------------------------------------------- */
/*                                                                           */
/* (c) Copyright 2009 - 2013, 2014,  EdgeTech                                */
/*                                                                           */
/* This file contains proprietary information, and trade secrets of          */
/* EdgeTech, and may not be disclosed or reproduced without the prior        */
/* written consent of EdgeTech.                                              */
/*                                                                           */
/* EdgeTech is not responsible for the consequences of the use or misuse     */
/* of this software, even if they result from defects in it.                 */
/*                                                                           */
/* Bathymetric Specific messages and structures                              */
/* ------------------------------------------------------------------------- */

#ifndef __BATHYMETRIC_MESSAGES_H__
#define __BATHYMETRIC_MESSAGES_H__


/* ------------------------------------------------------------------------ */

#include "systemNamespace.h"
#include "PublicMessageOffsets.h"

/* ------------------------------------------------------------------------ */

#pragma pack(push, bathyMessages)

#pragma pack(4)


/* ------------------------------------------------------------------------ */
/* Message Types                                                            */
/* ------------------------------------------------------------------------ */
/* sonarMessage field indicates the type of data to follow.                 */
/* ------------------------------------------------------------------------ */

typedef enum
{
  /* ---------------------------------------------------------------------- */
  /* Bathymetric Control Messages                                           */
  /* ---------------------------------------------------------------------- */
```

```
/* Hypack Bathymetric Parameters (HypackControlMessageType)             */
BATHYMETRIC_HYPACK_MESSAGE_CONTROL = MESSAGE_OFFSET_BATHYMETRIC_CONTROL,

/* Local Bathymetric parameters (BathymetricParameterMessageType)       */
BATHYMETRIC_MESSAGE_LOCAL_CONTROL,

/* Public Bathymetric Parameters (BathymetricControlMessageType)        */
BATHYMETRIC_MESSAGE_CONTROL,

/* Topside verification (BathymetricVerificationMessageType)            */
/* Discover will ignore all topside commands until verification has     */
/* completed successfully.                                              */
/*                                                                      */
/* Verification Procedure:                                              */
/* 1) Topside sends BATHYMETRIC_MESSAGE_TOPSIDE_VERIFICATION get message*/
/*    (COMMAND_GET) - This requests a seed from Discover.               */
/* 2) Discover sends BATHYMETRIC_MESSAGE_TOPSIDE_VERIFICATION reply     */
/*    message (COMMAND_REPY) with the seed value.                       */
/* 3) Topside sends BATHYMETRIC_MESSAGE_TOPSIDE_VERIFICATION set message*/
/*    (COMMAND_SET) with the authorization value derived from the seed  */
/*    value.                                                            */
/* 4) Discover will now accept topside commands.                        */
/*                                                                      */
/* Verification is cleared on command socket disconnection.  Thus it    */
/* must be performed every time the command socket connects prior to    */
/* sending commands.                                                    */
BATHYMETRIC_MESSAGE_TOPSIDE_VERIFICATION,

/* Local sensor configuration (BathymetricAuxSensorControlMessageType)  */
BATHYMETRIC_MESSAGE_AUXILLARY_SENSOR_CONTROL,


/* -------------------------------------------------------------------- */
/* Bathymetric Data Messages                                            */
/* -------------------------------------------------------------------- */

/* Bathymetric Data (BathymetricDataMessageType)                        */
BATHYMETRIC_MESSAGE_DATA = MESSAGE_OFFSET_BATHYMETRIC_DATA,

/* Attitude (AttitudeMessageType)                                       */
BATHYMETRIC_MESSAGE_ATTITUDE,

/* Pressure (PressureMessageType)                                       */
BATHYMETRIC_MESSAGE_PRESSURE,

/* Altitude (AltitudeMessageType)                                       */
BATHYMETRIC_MESSAGE_ALTITUDE,

/* Position (PositionMessageType)                                       */
```

```
  BATHYMETRIC_MESSAGE_POSITION,

  /* -------------------------------------------------------------------- */
  /* Bathymetric Status Messages                                          */
  /* -------------------------------------------------------------------- */

  /* Processing statistics (BathymetricProcessingStatusMessageType)       */
  BATHYMETRIC_MESSAGE_PROCESSING_STATUS = MESSAGE_OFFSET_BATHYMETRIC_STATUS,

  /* Situation statistics (BathymetricSituationStatusMessageType)         */
  BATHYMETRIC_MESSAGE_SITUATION_STATUS,

  /* Miscellaneous status values (BathymetricMiscStatusMessageType)       */
  BATHYMETRIC_MESSAGE_MISC_STATUS,

}  BathymetricMessageType;


/* ---------------------------------------------------------------------- */
/* Message Structures                                                     */
/* ---------------------------------------------------------------------- */

#define BATHYMETRIC_DATA_MESSAGE_TYPE_VERSION      (4)


/* ---------------------------------------------------------------------- */
/* Bathymetric Data sample structure                                      */
/* - BathymetricDataMessageType version 1                                 */
/* - bathymetric processor versions 1.07 - 1.13                          */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /* time delay of this sample from start time in time scale factor units */
  /* Combine with the time offset to the first sample (timeToFirstSample) */
  /* to get the total time delay.                                         */
  UInt16 timeDelay;

  /* angle in angle scale factor units from nadir of sensor               */
  Int16 angle;

  /*  linear amplitude in amplitude scale factor units                    */
  UInt16 amplitude;

  /* quality flags                                                        */
  /* 0 - clear, 1 - set                                                   */
  /* Bit  0: Would be excluded by outlier removal filter                  */
  /* Bit  1: Would be excluded by water column filter                     */
  /* Bit  2: Would be excluded by amplitude filter                        */
```

```
  /* Bit  3: Would be excluded by quality filter                              */
  /* Bit  4: Would be excluded by SNR filter                                  */
  Byte qualityFlags;

  /* quality as defined by algorithm type                                     */
  Byte quality;

}  BathymetricSampleType_Version_1;


/* ---------------------------------------------------------------------- */
/* Bathymetric Data sample structure                                        */
/* - BathymetricDataMessageType versions 2 - 3                             */
/* - bathymetric processor versions 1.14 - 63.0.1.108                      */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /* time delay of this sample from start time in time scale factor units */
  /* Combine with the time offset to the first sample (timeToFirstSample) */
  /* to get the total time delay.                                           */
  UInt16 timeDelay;

  /* angle in angle scale factor units from nadir of sensor                 */
  Int16 angle;

  /*  linear amplitude in amplitude scale factor units                      */
  UInt16 amplitude;

  /* quality flags                                                          */
  /* 0 - clear, 1 - set                                                     */
  /* Bit  0: Would be excluded by outlier removal filter                    */
  /* Bit  1: Would be excluded by water column filter                       */
  /* Bit  2: Would be excluded by amplitude filter                          */
  /* Bit  3: Would be excluded by quality filter                            */
  /* Bit  4: Would be excluded by SNR filter                                */
  /* Bit  5: Null content binned data               - added in version 3 */
  /*         Null content data should not be processed.                     */
  Byte qualityFlags;

  /* quality and SNR values defined as follows:                             */
  /* Bits 0 - 4: SNR value in dB 0 - 31+                                    */
  /* Values below 0 are clamped at 0.  Values over 31 are clamped at 31.  */
  /*                                                                        */
  /* Bits 5 - 7: Quality factor:  0 -        quality < 50%                 */
  /*                              1 - 50% <= quality < 60%                 */
  /*                              2 - 60% <= quality < 70%                 */
  /*                              3 - 70% <= quality < 75%                 */
  /*                              4 - 75% <= quality < 80%                 */
```

```
  /*                             5 - 80% <= quality < 85%                 */
  /*                             6 - 85% <= quality < 90%                 */
  /*                             7 - 90% <= quality                       */
  Byte quality;

}  BathymetricSampleType_Versions_2_And_3;


/* --------------------------------------------------------------------- */
/* Bathymetric Data sample structure                                     */
/* - BathymetricDataMessageType versions 4                               */
/* - bathymetric processor versions 64.0.1.101 - Present                 */
/* --------------------------------------------------------------------- */

typedef struct
{
  /* time delay of this sample from start time in time scale factor units */
  /* Combine with the time offset to the first sample (timeToFirstSample) */
  /* to get the total time delay.                                         */
  UInt16 timeDelay;

  /* angle in angle scale factor units from nadir of sensor               */
  Int16 angle;

  /* amplitude in 0.5 dB increments                                       */
  Byte amplitude;

  /* angle uncertainty in 0.02 degree increments                          */
  Byte angleUncertainty;

  /* quality flags                                                        */
  /* 0 - clear, 1 - set                                                   */
  /* Bit  0: Would be excluded by outlier removal filter                  */
  /* Bit  1: Would be excluded by water column filter                     */
  /* Bit  2: Would be excluded by amplitude filter                        */
  /* Bit  3: Would be excluded by quality filter                          */
  /* Bit  4: Would be excluded by SNR filter                              */
  /* Bit  5: Null content binned data                                     */
  /*         Null content data should not be processed.                   */
  Byte qualityFlags;

  /* quality and SNR values defined as follows:                           */
  /* Bits 0 - 4: SNR value in dB 0 - 31+                                  */
  /* Values below 0 are clamped at 0.  Values over 31 are clamped at 31.  */
  /*                                                                      */
  /* Bits 5 - 7: Quality factor:  0 -        quality < 50%                */
  /*                             1 - 50% <= quality < 60%                 */
  /*                             2 - 60% <= quality < 70%                 */
  /*                             3 - 70% <= quality < 75%                 */
```

```
  /*                              4 - 75% <= quality < 80%                */
  /*                              5 - 80% <= quality < 85%                */
  /*                              6 - 85% <= quality < 90%                */
  /*                              7 - 90% <= quality                      */
  Byte quality;

}  BathymetricSampleType;


/* ------------------------------------------------------------------------ */
/* Bathymetric Data message structure                                       */
/* - BathymetricDataMessageType versions 1 - 3                              */
/* - bathymetric processor versions 1.00 - 63.0.1.108                       */
/* ------------------------------------------------------------------------ */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)            */
  UInt32 secondsSince1970;

  /*   4 -   7 : Nanosecond supplement to time                            */
  UInt32 nanoseconds;

  /*   8 -  11 : Ping Number                                              */
  UInt32 pingNumber;

  /*  12 -  13 : Number of BathymetricSampleType entries after the header */
  UInt16 numberOfSamples;

  /*  14 -  14 : Channel : 0 - port, 1 - starboard                        */
  Byte   channel;

  /*  15 -  15 : Algorithm type (0..N)                                    */
  Byte   algorithmType;

  /*  16 -  16 : Number of pulses, 1 - single pulse, > 1 - multi-pulse    */
  Byte   numberOfPulses;

  /*  17 -  17 : Pulse phase (0 <= x < numberOfPulses)                    */
  Byte   pulsePhase;

  /*  18 -  19 : Pulse ID                                                 */
  UInt16 pulseID;

  /*  20 -  23 : Transmitted pulse amplitude : (0.0 - 1.0)                */
  /* 0 = no power (listen only), 1.0 - maximum power                      */
  Single pulseAmplitude;

  /* The center frequency is (startFrequency + endFrequency) / 2 and is   */
```

```
/* different for each mutipulse pulse phase.                                  */

/*  24 -  27 : Chirp start frequency (Hz)                                     */
Single startFrequency;

/*  28 -  31 : Chirp end frequency (Hz)                                       */
Single endFrequency;

/*  32 -  35 : Mixer frequency (Hz)                                           */
/* The frequency used to mix the acoustic returns to complex (I & Q)         */
/* basebanded data, normally near the center frequency.                      */
Single mixerFrequency;

/*  36 -  39 : Sample rate (Hz)                                               */
/* The A/D Converter output sample rate.  This is Not the sample rate of*/
/* the bathymetric data set which is determined by the time delays.          */
Single sampleRate;

/*  40 -  43 : Offset to first sample (nanoseconds)                           */
UInt32 timeToFirstSample;

/*  44 -  47 : Start time accuracy (seconds)                                  */
/* Maximum error of timeToFirstSample value.                                  */
/* This is the upper bound of potential jitter in the transmit/recieve       */
/* process.                                                                   */
Single startTimeAccuracy;

/*  48 -  51 : Time scale factor (seconds)                                    */
Single timeScaleFactor;

/*  52 -  55 : Time scale accuracy (percent)                                  */
/* Maximum error in timeDelay value in each BathymetricSampleType entry.*/
Single timeScaleAccuracy;

/*  56 -  59 : Angle scale factor (degrees)                                   */
Single angleScaleFactor;

/*  60 -  63 : Amplitude scale factor (counts)                                */
Single amplitudeScaleFactor;

/*  64 -  67 : Offset to first bottom return (nanoseconds)                    */
/* Set to zero if bottom not located.                                         */
/* This is normally the two way time to the bottom and can be used to        */
/* derive the sea floor depth in much the same way as a single beam echo*/
/* sounder.                                                                   */
UInt32 timeToFirstBottomReturn;

/*  68 -  68 : Version of this message                                        */
/*             0 - bathymetric processor versions 1.00 - 1.06                 */
```

```
  /*             1 - bathymetric processor versions 1.07 - 1.13         */
  /*             2 - bathymetric processor versions 1.14 - 63.9.1.102     */
  /*             3 - bathymetric processor versions 63.0.1.103-63.0.1.108 */
  /*               - discover versions 33.0.1.104 - 34.0.1.101            */
  /* current version is defined in BATHYMETRIC_DATA_MESSAGE_TYPE_VERSION  */
  Byte version;

  /*  69 -  69 : Output Binning                                           */
  /*             0 - all points                                           */
  /*             1 - equi-distance binning                                */
  /*             2 - equi-angle binning                                   */
  Byte binning;

  /*  70 -  71 : Reserved for future use                                  */
  Byte reservedA[2];

  /*  72 -  75 : Span of binned data in this record, in meters or degrees */
  /* depending on the binning type.  The total span is twice this amount  */
  /* If output binning = 0, maximum processing range.                     */
  /* If output binning = 1, # of bins x the bin size. (meters)            */
  /* If output binning = 2, # of beams x the beam width. (degrees)        */
  Single totalSpan;

  /*  76 -  79 : Reserved for future use                                  */
  UInt32 reservedB;

  /* -------------------------------------------------------------------- */
  /* Data area begins here                                                */
  /* -------------------------------------------------------------------- */
  /* Data begins at byte 80 having numberOfSamples entries of             */
  /* BathymetricSampleType                                                */
  /*                                                                      */
  /* BathymetricSampleType  data[numberOfSamples];                        */
  /* -------------------------------------------------------------------- */

}  BathymetricDataMessageType_Versions_1_Thru_3;


/* ---------------------------------------------------------------------- */
/* Bathymetric Data message structure                                     */
/* - BathymetricDataMessageType version 4                                 */
/* - bathymetric processor versions 64.0.1.102 - present                  */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)            */
  UInt32 secondsSince1970;
```

```
/*   4 -   7 : Nanosecond supplement to time                           */
UInt32 nanoseconds;

/*   8 -  11 : Ping Number                                             */
UInt32 pingNumber;

/*  12 -  13 : Number of BathymetricSampleType entries after the header */
UInt16 numberOfSamples;

/*  14 -  14 : Channel : 0 - port, 1 - starboard                       */
Byte   channel;

/*  15 -  15 : Algorithm type (0..N)                                   */
Byte   algorithmType;

/*  16 -  16 : Number of pulses, 1 - single pulse, > 1 - multi-pulse    */
Byte   numberOfPulses;

/*  17 -  17 : Pulse phase (0 <= x < numberOfPulses)                   */
Byte   pulsePhase;

/*  18 -  19 : Pulse Length in microseconds                            */
UInt16 pulseLength;

/*  20 -  23 : Transmitted pulse amplitude : (0.0 - 1.0)               */
/* 0 = no power (listen only), 1.0 - maximum power                     */
Single pulseAmplitude;

/* The center frequency is (startFrequency + endFrequency) / 2 and is   */
/* different for each mutipulse pulse phase.                           */

/*  24 -  27 : Chirp start frequency (Hz)                              */
Single startFrequency;

/*  28 -  31 : Chirp end frequency (Hz)                                */
Single endFrequency;

/*  32 -  35 : Mixer frequency (Hz)                                    */
/* The frequency used to mix the acoustic returns to complex (I & Q)    */
/* basebanded data, normally near the center frequency.                */
Single mixerFrequency;

/*  36 -  39 : Sample rate (Hz)                                        */
/* The A/D Converter output sample rate.  This is Not the sample rate of*/
/* the bathymetric data set which is determined by the time delays.     */
Single sampleRate;

/*  40 -  43 : Offset to first sample (nanoseconds)                    */
UInt32 timeToFirstSample;
```

```
/*  44 -  47 : Time delay uncertainty (seconds)                          */
/* Maximum time delay error.                                             */
/* This is the potential acoustic uncertainty of the true delay to each */
/* detected echo.                                                        */
/* This time delay uncertainty should be used to to determine the range */
/* uncertainty for each sample in the packet                            */
Single timeDelayUncertainty;

/*  48 -  51 : Time scale factor (seconds)                               */
Single timeScaleFactor;

/*  52 -  55 : Time scale accuracy (percent)                             */
/* Maximum error in timeDelay value in each BathymetricSampleType entry.*/
Single timeScaleAccuracy;

/*  56 -  59 : Angle scale factor (degrees)                              */
Single angleScaleFactor;

/*  60 -  63 : Reserved for future use                                   */
UInt32 Reserved_01;

/*  64 -  67 : Offset to first bottom return (nanoseconds)               */
/* Set to zero if bottom not located.                                    */
/* This is normally the two way time to the bottom and can be used to   */
/* derive the sea floor depth in much the same way as a single beam echo*/
/* sounder.                                                              */
UInt32 timeToFirstBottomReturn;

/*  68 -  68 : Version of this message                                   */
/*             0 - bathymetric processor versions 1.00 - 1.06            */
/*             1 - bathymetric processor versions 1.07 - 1.13            */
/*             2 - bathymetric processor versions 1.14 - 63.9.1.102      */
/*             3 - bathymetric processor versions 63.0.1.103-63.0.1.108 */
/*               - discover versions 33.0.1.104 - 34.0.1.101             */
/*             4 - bathymetric processor versions 64.0.1.102 - present  */
/*               - discover versions 34.0.1.102 - present                */
/* current version is defined in BATHYMETRIC_DATA_MESSAGE_TYPE_VERSION  */
Byte version;

/*  69 -  69 : Output Binning                                            */
/*             0 - all points                                            */
/*             1 - equi-distance binning                                 */
/*             2 - equi-angle binning                                    */
Byte binning;

/*  70 -  70 : applied TVG in dB per 100 meters                          */
Byte appliedTVG;
```

```
  /*  71 -  71 : Reserved for future use                                   */
  Byte reservedA;

  /*  72 -  75 : Span of binned data in this record, in meters or degrees */
  /* depending on the binning type.  The total span is twice this amount  */
  /* If output binning = 0, maximum processing range.                     */
  /* If output binning = 1, # of bins x the bin size. (meters)            */
  /* If output binning = 2, # of beams x the beam width. (degrees)        */
  Single totalSpan;

  /*  76 -  79 : Reserved for future use                                   */
  UInt32 reservedB;

  /* ------------------------------------------------------------------ */
  /* Data area begins here                                              */
  /* ------------------------------------------------------------------ */
  /* Data begins at byte 80 having numberOfSamples entries of           */
  /* BathymetricSampleType                                              */
  /*                                                                    */
  /* BathymetricSampleType  data[numberOfSamples];                      */
  /* ------------------------------------------------------------------ */

}  BathymetricDataMessageType;


/* --------------------------------------------------------------------- */
/* Attitude message structure                                            */
/* --------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)           */
  UInt32 secondsSince1970;

  /*   4 -   7 : Nanosecond supplement to time                           */
  UInt32 nanoseconds;

  /*   8 -  11 : Data valid flags:                                       */
  /* 0 - clear, 1 - set                                                  */
  /* Bit  0: heading                                                     */
  /* Bit  1: heave                                                       */
  /* Bit  2: pitch                                                       */
  /* Bit  3: roll                                                        */
  /* Bit  4: yaw                                                         */
  UInt32 flags;

  /*  12 -  15 : Heading (Degrees) (0.0 - 360.0)                         */
  Single heading;
```

```
  /*  16 -  19 : Heave (meters)  - down is positive                         */
  Single heave;

  /*  20 -  23 : Pitch (degrees) - bow up is positive                       */
  Single pitch;

  /*  24 -  27 : Roll (degrees)  - port up is positive                      */
  Single roll;

  /*  28 -  31 : Yaw (degrees)  - starboard is positive                     */
  Single yaw;

}  AttitudeMessageType;


/* ---------------------------------------------------------------------- */
/* Pressure Message structure                                             */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)            */
  UInt32 secondsSince1970;

  /*   4 -   7 : Nanosecond supplement to time                            */
  UInt32 nanoseconds;

  /*   8 -  11 : Data valid flags:                                        */
  /* 0 - clear, 1 - set                                                   */
  /* Bit 0: pressure                                                      */
  /* Bit 1: water temperature                                             */
  /* Bit 2: salt PPM                                                      */
  /* Bit 3: conductivity                                                  */
  /* Bit 4: sound velocity                                                */
  /* Bit 5: depth                                                         */
  UInt32 flags;

  /*  12 -  15 : Absolute Pressure (PSI)                                  */
  Single pressure;

  /*  16 -  19 : Water temperature (Degrees C)                            */
  Single waterTemperature;

  /*  20 -  23 : Salinity (Parts Per Million)                             */
  Single saltPPM;

  /*  24 -  27 : Conductivity (micro-Siemens per cm)                      */
  Single conductivity;
```

```
  /*  28 -  31 : Velocity of Sound (meters per second)                     */
  Single soundVelocity;

  /*  32 -  35 : Depth (meters)                                             */
  Single depth;

} PressureMessageType;


/* ------------------------------------------------------------------------ */
/* Altitude Message structure                                               */
/* ------------------------------------------------------------------------ */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)              */
  UInt32 secondsSince1970;

  /*   4 -   7 : Nanosecond supplement to time                              */
  UInt32 nanoseconds;

  /*   8 -  11 : Data valid flags:                                          */
  /* 0 - clear, 1 - set                                                     */
  /* Bit 0: altitude                                                        */
  /* Bit 1: speed                                                           */
  /* Bit 2: heading                                                         */
  UInt32 flags;

  /*  12 -  15 : Altitude (meters)                                          */
  Single altitude;

  /*  16 -  19 : Speed (Knots)                                              */
  Single speed;

  /*  20 -  23 : Heading (Degrees) (0.0 - 360.0)                            */
  Single heading;

}  AltitudeMessageType;


/* ------------------------------------------------------------------------ */
/* Position Message structure                                               */
/* ------------------------------------------------------------------------ */

typedef struct
{
  /*   0 -   3 : Time since 1/1/1970 in seconds (time() value)              */
  UInt32 secondsSince1970;
```

```
  /*   4 -   7 : Nanosecond supplement to time                                */
  UInt32 nanoseconds;

  /*   8 -   9 : Data valid flags:                                            */
  /* 0 - clear, 1 - set                                                       */
  /* Bit 0: UTM Zone                                                          */
  /* Bit 1: easting                                                           */
  /* Bit 2: northing                                                          */
  /* Bit 3: latitude                                                          */
  /* Bit 4: longitude                                                         */
  /* Bit 5: speed                                                             */
  /* Bit 6: heading                                                           */
  /* Bit 7: altitude                                                          */
  UInt16 flags;

  /*  10 -  11 : UTM Zone                                                     */
  UInt16 UTMZone;

  /*  12 -  19 : Easting (meters)                                             */
  Double easting;

  /*  20 -  27 : Northing (meters)                                            */
  Double northing;

  /*  28 -  35 : latitude (degrees) north is positive                         */
  Double latitude;

  /*  36 -  43 : longitude (degrees) east is positive                         */
  Double longitude;

  /*  44 -  47 : Speed (Knots)                                                */
  Single speed;

  /*  48 -  51 : Heading (Degrees) (0.0 - 360.0)                              */
  Single heading;

  /*  52 -  55 : antennae altitude (meters)                                   */
  Single altitude;

}  PositionMessageType;


/* ---------------------------------------------------------------------- */
/* Hypack control message structure                                       */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : minimum altitude in meters               [-10.0 - 100.0] */
```

```
  Single minimumDepth;

  /*   4 -   7 : maximum altitude in meters                  [0.0 - 350.0] */
  Single maximumDepth;

  /*   8 -  11 : port offset limit in meters                 [0.0 - 350.0] */
  Single portOffsetLimit;

  /*  12 -  15 : starboard offset limit in meters            [0.0 - 350.0] */
  Single starboardOffsetLimit;

  /*  16 -  19 : port angle limit in degrees                 [ Not Used  ] */
  Single portAngleLimit;

  /*  20 -  23 : starboard angle limit in degrees            [ Not Used  ] */
  Single starboardAngleLimit;

  /*  24 -  27 : port sonar head vertical offset in meters  [0.0 - 100.0] */
  Single portSonarHeadVerticalOffset;

  /*  28 -  31 : starboard sonar head vert offset in meters [0.0 - 100.0] */
  Single starboardSonarHeadVerticalOffset;

  /*  32 - 127 : Reserved                                                */
  /* Reserved for future use                                             */
  UInt32 Reserved[24];

} HypackControlMessageType;


/* ---------------------------------------------------------------------- */
/* Bathymetric control message structure                                  */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : minimum altitude in meters                [-10.0 - 100.0] */
  Single minimumDepth;

  /*   4 -   7 : maximum altitude in meters                  [0.0 - 500.0] */
  Single maximumDepth;

  /*   8 -  11 : maximum swath in meters                    [10.0 - 600.0] */
  Single maximumSwath;

  /*  12 -  15 : port sonar head vertical offset in meters  [0.0 - 100.0] */
  Single portSonarHeadVerticalOffset;

  /*  16 -  19 : starboard sonar head vert offset in meters [0.0 - 100.0] */
```

```
  Single starboardSonarHeadVerticalOffset;

  /*  20 - 127 : Reserved                                          */
  /* Reserved for future use                                       */
  UInt32 Reserved[27];

} BathymetricControlMessageType;


/* ---------------------------------------------------------------------- */
/* Bathymetric verification message structure                             */
/* ---------------------------------------------------------------------- */

typedef struct
{
  /*   0 -   3 : Topside ID                                         */
  UInt32 topsideID;

  /*   4 -   7 : Seed/Authorization value                           */
  UInt32 value;

}  BathymetricVerificationMessageType;


/* ---------------------------------------------------------------------- */

#pragma pack(pop, bathyMessages)


/* ---------------------------------------------------------------------- */

#endif  /* __BATHYMETRIC_MESSAGES_H__ */


/* ---------------------------------------------------------------------- */
/*                    end BathymetricMessages.h                           */
/* ---------------------------------------------------------------------- */
```