RESON SeaBat™ 7k DATA FORMAT

INTERFACE CONTROL DOCUMENT

FOR

RESON 6046

Version 0.50

(Preliminary — Subject to Change)

Volume III

Approval

Notice

The information contained in this document is subject to change without notice. RESON makes no warranty of any kind with regard to this material, including (but not limited to) the implied warranties of merchant liability and fitness for a particular purpose. RESON shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Function	Name	Version	Date	Signature
Program Manager	Steve Weymann	1.00		

Protocol Version History

Protocol Version (DRF and NF)	ICD Version
1	0.01 to 0.31
2	0.32 to 0.47
3	0.48 +

Table of Contents

1. PURPOSE

This document is supplementary to reference [1], in which the general 7k format definitions are defined, and describes those 7k data records relevant to the RESON 6046 Payload Controller.

2. REFERENCES

[1] RESON SeaBat™ 7k Data Format: Volume I, RESON Inc, 2002.

3. RECORD TYPE DEFINITIONS

The following table summarizes the record type identifiers that specifically pertain to the RESON 6046 Payload Controller.

RECORD TYPE	DESCRIPTION
3000	EdgeTech FS-DW side scan sonar data record
3001	EdgeTech FS-DW sub-bottom profile data record
3100	Framed Bluefin data frame record
11000	Payload Controller — Command
11001	Payload Controller — Command Acknowledge
11002	Payload Controller — Alarm and Status
11200 – 11299	Reserved: Payload Controller — Sensor QC records

RECORD TYPE #3000 `EdgeTech FS-DW side scan data record'

DESCRIPTION:

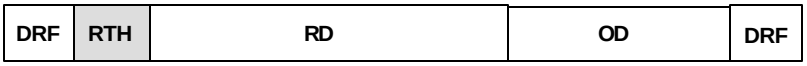
EdgeTech FS-DW side scan sonar data format.

The device id field of the DRF is used to distinguish between different EdgeTech subsystems, such as low and high frequency side scan sonars.

Each side scan channel imagery channel immediately follows the RTH and is in-term prefixed with its own CHANINFO structure. Port channel appears first followed by starboard.

The received EdgeTech data headers are appended (verbatim) to the end of the imagery data as optional data; sequenced port then starboard. The optional data Identifier field of the DRF determines the format of the EdgeTech header appended — 1 for sidescan data type[1]. Note: the EdgeTech weighting factor is not applied to the data but is present in the appended EdgeTech header.

DATA DEFINITION:



RECORD TYPE HEADER:

NAME	SIZE	DESCRIPTION
Millisecond time stamp	u32	Relative millisecond timer value.
Ping number	u32	Ping number as received from the EdgeTech subsystem.
Number of channels	u32	Number of imagery channels to follow (typically 2).
Total Bytes of channel data to follow	u32	Total bytes of channel data (and headers) to follow RTH (including optional data).
Data format	u32	Format of data: 0 – Envelope 1 – I and Q (complex)

Following the RTH is the concatenated imagery data each prefixed with a CHANNEL INFO header as follows:

[1] — Refer to appendix A for details of the EdgeTech data fields for the side scan sonar.

CHANNEL INFO

NAME	SIZE	DESCRIPTION
Channel number	u8	Channel number: 0 to Number of channels –1.
Channel type	u8	0 - port 1 - starboard
Data type	u8	0 - slant range 1 - ground range
Polarity	u8	0 – bipolar, 1 - unipolar
Bytes per sample	u8	Bytes per sample of the imagery.
Reserved 1	3 * u8	Reserved for future use.
Number of samples	u32	Number of samples in this channel.
Start time	u32	Start of first sample in microseconds relative to the ping time stamp.
Sample interval	u32	Data sample interval in microseconds.
Range	f32	Slant range or ground range in meters and depends on the data type field above.
Voltage (FSD)	f32	Analogue maximum amplitude. Should be –1 if not used.
Name	16 * i8	Channel name as a zero terminated character array.
Reserved 2	20 * u8	Padding and reserved fields.

RECORD TYPE #3001 `EdgeTech FS-DW Subbottom profiler data record'

DESCRIPTION:

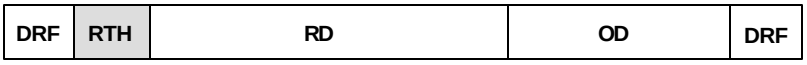
EdgeTech FS-DW sub-bottom profiler data record.

Channel imagery immediately follows the channel data and is prefixed with a CHANINFO structure.

The received EdgeTech SEG-Y header is appended (verbatim) to the end of the imagery data as optional data. The optional data identifier field of the DRF determines the format of the EdgeTech header appended: 2 for SEG-Y data type[2].

Note:

- SBP data will typically be complex (representing I & Q per sample) thus 4 bytes per sample.
- The EdgeTech weighting factor is not applied to the data but is present in the appended EdgeTech header (refer to appendix A).



RECORD TYPE HEADER:

NAME	SIZE	DESCRIPTION
Millisecond time stamp	u32	Relative millisecond timer value.
Ping number	u32	Ping number as received from the EdgeTech subsystem.
Number of channels	u32	Number of imagery channels to follow (typically 1).
Total Bytes of channel data to follow	u32	Total bytes of channel data (and headers) to follow this RTH including optional data.
Data format	u32	Sample format: 0 – Envelope 1 – I and Q (complex)

[2] — Refer to appendix A for details of the EdgeTech SEG-Y data format.

CHANNEL INFO

NAME	SIZE	DESCRIPTION
Channel number	u8	Channel number 0 to Number of channels –1.
Channel type	u8	0 – port 1 – starboard
Data type	u8	0 – slant range 1 – ground range
Polarity	u8	0 – bipolar 1 – unipolar
Bytes per sample	u8	Bytes per sample of imagery.
Reserved 1	3 * u8	Reserved for future use.
Number of samples	u32	Number of samples in this channel.
Start time	u32	Start of first sample in micro seconds relative to the ping time stamp.
Sample interval	u32	Data sample interval in microseconds.
Range	f32	Slant range or ground range in meters and depends on the data type field above.
Voltage (FSD)	f32	Analog maximum amplitude. Should be –1 if not used.
Name	16 * i8	Channel name as a zero terminated character array.
Reserved 2	20 * u8	Padding and reserved fields.

RECORD TYPE # 3100 — Framed Bluefin Data Frame Record

DESCRIPTION:

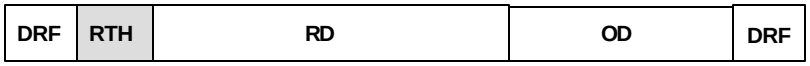
This is a general container for various Bluefin data frames — each identified by the data format field of the RTH as defined below.

DATA DEFINITION:

The embedded data consists of a RTH, followed by a variable number of Bluefin defined data frames. For high record rates, multiple frames will be concatenated into a single record; the number will be rate dependent and typically represent 1 second of data.

For example, the navigation frames are output at nominal rate of 25 Hz — thus 25 frames representing 1 second of data will typically be contained by a single record.

Refer to appendix B for a definition of the record data frames.



RECORD TYPE HEADER

NAME	SIZE	DESCRIPTION
Time stamp of first frame	u32	Millisecond time stamp of first frame in record
Number of frames	u32	Number of frames embedded in this record.
Frame size	u32	Embedded frame size in bytes
Data format	u32	Data type identifier 0 – Navigation data 1 – Environment data
Reserved	16 * u8	Reserved.

RECORD TYPE # 11000 'Payload Controller Command'

DESCRIPTION:

This record type is used to send commands to the Payload Controller (PLC) or specified subordinate sensors for specific command and control purposes. Commands are routed using the appropriate sensor index identifiers.

Subordinate sensor commands and optional PLC command fields use an embedded ASCII variable length command data field; parameters are comma-delimited fields. Floating point parameters shall use an ASCII "." to delimit the fractional portion of a number. Messages are case sensitive.

Command and message definitions are provided in appendix F of this document.

ASCII COMMAND MESSAGE SYNTAX

<COMMAND>[[,<PARAMETER 1>][,<PARAMETER_2>]… [,<PARAMETER_N>]]

DATA DEFINITION:

DRF	RTH	RD	OD	DRF

NAME	SIZE	DESCRIPTION
Sensor index	u32	Identifies the target for this command: -1 – PLC only >= 0 – Specified sensor identifier.
Command ID	u32	Command identifier (see table C.1).
Action	u32	Specifies whether the command is setting or retrieving state information: 0 – Unspecified 1 – Set 2 – Get (typically solicits a response) Note: Typically for a retrieve operation the optional ASCII COMMAND DATA field is not required.
Command bytes to follow	u32	Number of bytes in the command to follow and may be zero.
ASCII COMMAND DATA	Dynamic	Optional field specifying PLC or sensor ASCII command.

Note: the sensor index field is a relative index used to identify the target of a specific command. The index is assigned by the PLC and will be a unique sequential number for each sensor in the run-list of the sensor suite.

RECORD TYPE # 11001 'Payload Controller Command Acknowledge'

DESCRIPTION:

In verbose mode, the Payload Controller (PLC) will reply to all commands received from its remote host(s) in order to acknowledge command receipt and to indicate its success or failure. Conversely, the PLC will not reply to receive commands in quite mode.

Note: the PLC makes no guarantee of sequencing this message type with a solicited response — connected hosts should be prepared to handle these messages in any order.

The message format is as follows:

DATA DEFINITION:

DRF	RTH	RD	OD	DRF

NAME	SIZE	DESCRIPTION
Command ID	u32	Command ID of the previously received command.
Sensor index	i32	Sensor index identifying the target sensor for the previous command. Values are: -1 – Payload controller. >= 0 – Specified sensor. See also record #11000.
Command status	u32	Last command receive status: 0 – Command rejected 1 – Command accepted 2 – Unknown command.

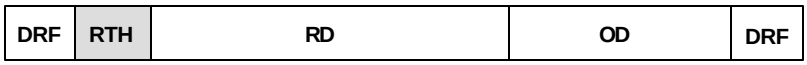
RECORD TYPE # 11002 'Payload Controller Alarm or Status'

DESCRIPTION:

This message will be sent to connected hosts either in response to a solicited enquiry or an unsolicited message due to a significant change in the system state such as an alarm condition.

Appendix C of this document gives the relevant status and alarm message identifiers and formats.

DATA DEFINITION:



NAME	SIZE	DESCRIPTION
Type	i32	Type of message: 0 – Status, 1 – Alarm
Alarm mode	i32	Alarm mode (relevant only to alarms): -1 = Unspecified, 0 = Alarm Clear 1 = Alarm Active
Identifier	i32	Message or alarm identifier (c.f. appendix F).
Message bytes	u32	Bytes to follow in dynamic portion (RD) of this message.

4. APPENDIX A — EDGETECH FS-DW FORMAT HEADERS

The following are the EdgeTech defined channel header structure definitions for the FS-DW series of sonars that detail the channel data formats pertaining to the SEG-Y and sidescan formats (c.f. the format identifier field of record type # 3000 and 3001).

```
/* ------------------------------------------------------------------------ */
/* SidescanDefs.h                                                           */
/* ------------------------------------------------------------------------ */
/*                                                                          */
/* (c) Copyright 1999  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.                */
/*                                                                          */
/* ------------------------------------------------------------------------ */
/*                                                                          */
/* EdgeTech sidescan data format description.                               */
/*                                                                          */
/* ------------------------------------------------------------------------ */

#ifndef __SIDESCANDEFS_H__
#define __SIDESCANDEFS_H__


/* ------------------------------------------------------------------------ */
/* includes                                                                 */
/* ------------------------------------------------------------------------ */

#include "SegyDefs.h"


/* ------------------------------------------------------------------------ */
/* Each record of data has a 80 byte header.  The content of which is       */
/* defined below.                                                           */
/*                                                                          */
/* Unused fields should be set to 0                                         */
/* ------------------------------------------------------------------------ */

typedef struct
{
  /*   0 -   1 : Subsystem (0 .. n)                                        */
  unsigned short int subsystem;

  /*   2 -   3 : Channel Number (0 .. n)                                   */
  unsigned short int channelNum;

  /*   4 -   7 : Ping number (increments with ping)                        */
  unsigned long int pingNum;

  /*   8 -   9 : Packet number (1..n) Each ping starts with packet 1       */
  unsigned short int packetNum;

  /*  10 -  11 : TriggerSource (0 = internal, 1 = external)               */
  unsigned short int trigSource;

  /*  12 -  15 : Samples in this packet                                    */
  unsigned long int samples;

  /*  16 -  19 : Sample interval in ns of stored data                      */
  unsigned long int sampleInterval;

  /*  20 -  23 : starting Depth (window offset) in samples                 */
  unsigned long int startDepth;
```

```
  /*  24 -  25 : -- defined as 2 -N volts for lsb                          */
short int weightingFactor;

/*  26 -  27 : Gain factor of ADC                                         */
unsigned short int ADCGain;

/*  28 -  29 : Maximum absolute value for ADC samples for this packet     */
unsigned short int ADCMax;

/*  30 -  31 : Range Setting (meters X 10)                                */
unsigned short int rangeSetting;

/*  32 -  33 : Unique pulse identifier                                    */
unsigned short int pulseID;

/*  34 -  35 : Mark Number (0 = no mark)                                  */
unsigned short int markNumber;

/*  36 -  37 : Data format                                                */
/*   0 = 1 short  per sample  - envelope data                             */
/*   1 = 2 shorts per sample  - stored as real(1), imag(1),               */
/*   2 = 1 short  per sample  - before matched filter (raw)               */
/*   3 = 1 short  per sample  - real part analytic signal                 */
/*   NOTE: For type = 1, the total number of bytes of data to follow is   */
/*   4 * samples.  For all other types the total bytes is 2 * samples.    */
unsigned short int dataFormat;

/*  38 -  39 : Reserved field to round up to a 32-bit word boundary       */
unsigned short int reserved;


/* -------------------------------------------------------------------- */
/* computer date / time data acquired                                   */
/* -------------------------------------------------------------------- */

/*  40 -  43 : Millieconds today                                          */
unsigned long int millisecondsToday;

/*  44 -  45 : Year                                                       */
short int year;

/*  46 -  47 : Day of year (1 - 366)                                      */
unsigned short int day;

/*  48 -  49 : Hour of day (0 - 23)                                       */
unsigned short int hour;

/*  50 -  51 : Minute (0 - 59)                                            */
unsigned short int minute;

/*  52 -  53 : Second (0 - 59)                                            */
unsigned short int second;


/* -------------------------------------------------------------------- */
/* Auxillary sensor information                                         */
/* -------------------------------------------------------------------- */

/*  54 -  55 : Compass heading (minutes)                                  */
short int heading;

/*  56 -  57 : Pitch (minutes)                                            */
short int pitch;

/*  58 -  59 : Roll (minutes)                                             */
short int roll;

/*  60 -  61 : Heave (centimeters)                                        */
short int heave;

/*  62 -  63 : Yaw (minutes)                                              */
short int yaw;

/*  64 -  67 : Vehicle depth (centimeters)                                */
unsigned long int depth;

/*  68 -  69 : Temperature (degrees Celsius X 10)                         */
```

```
    short int temperature;

  /*  70 -  79 : Reserved for future use                                  */
  char reserved2[10];


  /* ------------------------------------------------------------------- */
  /* Data area begins here                                               */
  /* ------------------------------------------------------------------- */
  /* Data begins at byte 80.                                             */
  /* short int data[];                                                   */
  /* ------------------------------------------------------------------- */

} SidescanHeaderType;


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

#endif  /* end Not __SIDESCANDEFS_H__ */

/* ------------------------------------------------------------------- */
/*                          end SidescanDefs.h                         */
/* ------------------------------------------------------------------- */
```

```
 /* ------------------------------------------------------------------- */
 /* SegyDefs.h                                                          */
/* ------------------------------------------------------------------- */
/*                                                                     */
/* (c) Copyright 1997, 1998, 2000  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.           */
/*                                                                     */
/* ------------------------------------------------------------------- */
/*                                                                     */
/* EdgeTech SEG-Y subbottom data format description.                   */
/*                                                                     */
/* ------------------------------------------------------------------- */

#ifndef __SEGYDEFS_H__
#define __SEGYDEFS_H__


/* ------------------------------------------------------------------- */
/* Values for trigger type field                                       */
/* ------------------------------------------------------------------- */

typedef enum
{
  TRIGGER_INTERNAL,           /* Internal trigger (0)                  */
  TRIGGER_EXTERNAL,           /* External trigger (1)                  */
} TriggerSourceType;


/* ------------------------------------------------------------------- */
/* Each record of data has a 240 byte header, the content of which is  */
/* defined below.  The structure is intended to be compatible with most */
/* implemented SEG-Y formats, as well as the original standard.        */
/*                                                                     */
/* Unused fields should be set to 0                                    */
/*                                                                     */
/* SEG-Y defined fields (** -> Highly Recommended in SEG-Y)            */
/* ------------------------------------------------------------------- */

typedef struct
{
  /* 0-3 : Trace Sequence Number (always 0) **                         */
  long int sequenceNumber;

  /* 4-7 : Starting depth (window offset) in samples.                  */
  unsigned long int startDepth;

  /* 8-11: Ping number (increments with ping) **                       */
  unsigned long int pingNum;

  /* 12-15 : Channel Number (0 .. n) **                                */
  unsigned long int channelNum;

  /* 16-27                                                             */
  short int unused1[6];

  /* 28-29 : ID Code (always 1 => seismic data) **                     */
  short int traceIDCode;

  /* 30-33                                                             */
  short int unused2[2];

  /* 34-35 : DataFormatType                                            */
  /*   0 = 1 short  per sample  - envelope data                        */
  /*   1 = 2 shorts per sample, - stored as real(1), imag(1),          */
  /*   2 = 1 short  per sample  - before matched filter                */
  /*   3 = 1 short  per sample  - real part analytic signal            */
  /*   4 = 1 short  per sample  - pixel data / ceros data              */
  short int dataFormat;

  /* 36-37 : Distance from towfish to antennae in cm                   */
  short int NMEAantennaeR;
```

```
  /* 38-39 : Distance to antennae starboard direction in cm                */
  short int NMEAantennaeO;

/* 40-71 : Reserved for RS232 data - TBD                                   */
char RS232[32];

/* ---------------------------------------------------------------------- */
/* Navigation data :                                                      */
/* If the coorUnits are seconds(2), the x values represent longitude      */
/* and the y values represent latitude.  A positive value designates      */
/* the number of seconds east of Greenwich Meridian or north of the       */
/* equator.                                                               */
/* ---------------------------------------------------------------------- */

/* 72-75 : Meters or Seconds of Arc                                        */
long int sourceCoordX;

/* 76-79 : Meters or Seconds of Arc                                        */
long int sourceCoordY;

/* 80-83 : mm or 10000 * (Minutes of Arc)                                  */
long int groupCoordX;

/* 84-87 : mm or 10000 * (Minutes of Arc)                                  */
long int groupCoordY;

/* 88-89 : Units of coordinates - 1->length (x /y), 2->seconds of arc      */
short int coordUnits;

/* 90-113 : Annotation string                                              */
char annotation[24];

/* 114-115 : Samples in this packet **                                     */
/* Note:  Large sample sizes require multiple packets.                     */
unsigned short int samples;

/* 116-119 : Sample interval in ns of stored data **                       */
unsigned long int sampleInterval;

/* 120-121 : Gain factor of ADC                                            */
unsigned short int ADCGain;

/* 122-123 : user pulse power setting (0 - 100) percent                    */
short int pulsePower;

/* 124-125 : correlated data 1 - No, 2 - Yes                               */
short int correlated;

/* 126-127 : Starting frequency in 10 * Hz                                 */
unsigned short int startFreq;

/* 128-129 : Ending frequency in 10 * Hz                                   */
unsigned short int endFreq;

/* 130-131 : Sweep length in ms                                            */
unsigned short int sweepLength;

/* 132-139                                                                 */
short int unused7[4];

/* 140-141 : alias Frequency (sample frequency / 2)                        */
unsigned short int aliasFreq;

/* 142-143 : Unique pulse identifier                                       */
unsigned short int pulseID;

/* 144-155                                                                 */
short int unused8[6];

/* 156-157 : Year data recorded (CPU time)                                 */
short int year;

/* 158-159 : day                                                           */
short int day;

/* 160-161 : hour                                                          */
short int hour;
```

```
  /* 162-163 : minute                                                    */
  short int minute;

/* 164-165 : second                                                     */
short int second;

/* 166-167 : Always 3 (other not specified by standard)                 */
short int timeBasis;

/* 168-169 :  weighting factor for block floating point expansion       */
/*            -- defined as 2 -N volts for lsb                          */
short int weightingFactor;

/* 170-171 :                                                            */
short int unused9;

/* -------------------------------------------------------------------- */
/* From pitch/roll/temp/heading sensor                                  */
/* -------------------------------------------------------------------- */

/* 172-173 : Compass heading (100 * degrees) -180.00 to 180.00 degrees  */
short int heading;

/* 174-175 : Pitch                                                      */
short int pitch;

/* 176-177 : Roll                                                       */
short int roll;

/* 178-179 : Temperature (10 * degrees C)                               */
short int temperature;

/* -------------------------------------------------------------------- */
/* User defined area from 180-239                                       */
/* -------------------------------------------------------------------- */

/* 180-181 : Heave compensation offset (samples)                        */
short int heaveCompensation;

/* 182-183 : TriggerSource (0 = internal, 1 = external)                 */
short int trigSource;

/* 184-185 : Mark Number (0 = no mark)                                  */
unsigned short int markNumber;

/* 186-187 : Hour                                                       */
short int NMEAHour;

/* 188-189 : Minutes                                                    */
short int NMEAMinutes;

/* 190-191 : Seconds                                                    */
short int NMEASeconds;

/* 192-193 : Course                                                     */
short int NMEACourse;

/* 194-195 : Speed                                                      */
short int NMEASpeed;

/* 196-197 : Day                                                        */
short int NMEADay;

/* 198-199 : Year                                                       */
short int NMEAYear;

/* 200-203 : Millieconds today                                          */
unsigned long int millisecondsToday;

/* 204-205 : Maximum absolute value for ADC samples for this packet     */
unsigned short int ADCMax;

/* 206-207 : System constant in tenths of a dB                          */
short int calConst;

/* 208-209 : Vehicle ID                                                 */
short int vehicleID;
```

```
    /* 210-215 : Software version number                                  */
    char softwareVersion[6];

  /* Following items are not in X-Star                                    */

  /* 216-219 : Initial spherical correction factor (useful for multiping /*/
  /* deep application) * 100                                            */
  long int sphericalCorrection;

  /* 220-221 : Packet number (1 - N) (Each ping starts with packet 1)     */
  unsigned short int packetNum;

  /* 222-223 : A/D decimation before FFT                                  */
  short int ADCDecimation;

  /* 224-225 : Decimation factor after FFT                              */
  short int decimation;

  /* 226-239                                                            */
  short int unuseda[7];

  /* ------------------------------------------------------------------ */
  /* Data area begins here                                              */
  /* ------------------------------------------------------------------ */
  /* Data begins at byte 240, has this.samples points in it             */
  /* short int data[];                                                  */
  /* ------------------------------------------------------------------ */
} SegyDataType;


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

#endif  /* end Not __SEGYDEFS_H__ */

/* ---------------------------------------------------------------------- */
/*                            end SegyDefs.h                              */
/* ---------------------------------------------------------------------- */
```

5. APPENDIX B — BLUEFIN DATA FRAME DEFINITIONS

The following Bluefin defined data message types are relevant to record number 3100. Each message type below is follows the network / data frame specified below.

Network/Data Frame:

NAME	SIZE	DESCRIPTION
Packet Size	u32	Size in bytes of this packet including the header and appended data.
Version	u16	Version of this frame (e.g.: 1, 2, etc.)
Offset	u16	Offset, in bytes, to the start of data from the start of this packet.
Data Type	u32	Data Description. 1 = Navigation message 2 = Environment message
Data Size	u32	Size of data, in bytes.
Time Stamp	u8*10	7K Time Format, UTC.
Checksum	u32	CRC32 checksum for all bytes in record.
Reserved	u16	Reserved.
Data	Dynamic	Start of data section with either a partial or one or more records.

Navigation Message:

NAME	SIZE	DESCRIPTION
Quality Metrics	u32	NA
Latitude	f64	Radians
Longitude	f64	Radians
Speed	f32	Meters / Second
Depth	f64	Meters
Altitude	f64	Meters
Roll	f32	Radians
Pitch	f32	Radians
Yaw	f32	Radians
Northing Rate	f32	Meters / Second
Easting Rate	f32	Meters / Second
Depth Rate	f32	Meters / Second
Altitude Rate	f32	Meters / Second
Roll Rate	f32	Radians / Second
Pitch Rate	f32	Radians / Second
Yaw Rate	f32	Radians / Second
Position Time	f64	Seconds
Atitude Time	f64	Seconds

Navigation Message Quality Metrics:

NAME	BIT NUMBER	DESCRIPTION
Latitude	00 (LSB)	0 – Invalid, 1 – Valid
Longitude	01	0 – Invalid, 1 – Valid
Speed	02	0 – Invalid, 1 – Valid
Depth	03	0 – Invalid, 1 – Valid
Altitude	04	0 – Invalid, 1 – Valid
Roll	05	0 – Invalid, 1 – Valid
Pitch	06	0 – Invalid, 1 – Valid
Yaw	07	0 – Invalid, 1 – Valid
Northing Rate	08	0 – Invalid, 1 – Valid
Easting Rate	09	0 – Invalid, 1 – Valid
Depth Rate	10	0 – Invalid, 1 – Valid
Altitude Rate	11	0 – Invalid, 1 – Valid
Roll Rate	12	0 – Invalid, 1 – Valid
Pitch Rate	13	0 – Invalid, 1 – Valid
Yaw Rate	14	0 – Invalid, 1 – Valid
Position Time	15	0 – Invalid, 1 – Valid
Attitude Time	16	0 – Invalid, 1 – Valid
Undefined	17 – 31	NA

Environment Message:

NAME	SIZE	DESCRIPTION
Quality metrics	u32	See bit flag definition below.
Sound velocity	f32	Sound speed in m/s.
Reserved	u8 * 88	Padding and reserved space.

Environment Message Quality Metrics:

NAME	BIT NUMBER	DESCRIPTION
Sound velocity	0 (LSB)	0 – Invalid, 1 – Valid
Undefined	1 to 31	NA

Note: the term Yaw used in this appendix is typically synonymous with heading.

6. APPENDIX C — RESON PLC COMMANDS AND MESSAGES

Table C.1 pertains to record type 11000 (the RESON Payload Controller remote command set).

Tables C.2 and C.3 pertain to record type 11002 — the RESON Payload Controller's status and alarm messages respectively. The message type is determined by the "Type" field of the RTH. The tabulated ID field corresponds to the "Identifier" field of the RTH.

Table C.4 specifies the generic command formats relevant to ID 0 of table C.1. Note that although a sensor may receive any of these commands, only those commands relevant to its functional capabilities need be enacted.

Note also that the format specifier field given in the following tables use a pseudo C programming language printf style format and are advisory only.

Table C.1 — RESON PLC Remote Commands.

Command	ID	Format	Description
Generic	0	%s	<command> - generic command as listed in table C.4 below.
Logging	1	%d	<flag> - logging enable (? 0) disable (0).
Switch	2	%s	<name> - changes the logging file automatically ("Auto") based on size or to the specified name otherwise.
Path	3	%s	Sets the logged data path to the specified folder.
Version	4	-	Solicits a PLC version message.
Load	5	-	Load the PLC run list from stored 6046.sav file.
Save	6	-	Save current run list to the 6046.sav file.
Subscribe	7	%d, %lu	<sensor id, record type> - allows a client to subscribe to a particular 7k data message for a given sensor id on a per client basis.
Unsubscribe	8	%d, %lu	<sensor id, record type> - withdraws subscription to a 7k data message for the specified sensor and for the given client.
Alarm	9	%d, %d	<alarm id, action> - disable (0) or enable (1) a given alarm on a per client basis.
Status	10	%d	<sensor id> - solicits a subsequent status message.
Verbose	11	%d	<flag> - disables (0) or enables (1) verbose mode. Disabled is synonymous with quiet mode.
Reset	12	%d	<sensor id> - resets the given sensor back to its default state. This command is reserved for future use.
Shutdown	13	%d	<shutdown type> - issues a shutdown request to the PLC (any) and optionally the OS (1) on which the PLC is running. When 1, shutdown will also be issued to subordinate sensors where they are capable of doing so.
Modules	14	-	Solicits the list of currently supported sensor modules.

Table C.1 — RESON PLC Remote Commands (continued)

Run list	15	-	Solicits the list of sensors that are currently loaded.
Add	16	%d, %d, %d, %s	<module number, subsystem id, port number, sensor name> - adds a sensor to the run list and executes the sensor's startup procedure.
Remove	17	%d	Removes a sensor from the run list and executes the sensor's shutdown procedure.
Report logging status	18	-	Solicits PLC's logging status message (file details, table C.2, id 3).
Reserved	19	-	Reserved for future use.
Time sync	20	%04u/%02u/%02u,%02u:%02u:%8.5f	<year, month, day, hour, minute, seconds> - grossly disciplines the PLC's clock to the specified time.
Health	21	-	Solicits a health message as defined by identifier 6 of table C.2.
Auto start	22	%lu	<mode> - disables (0) or enables (1) logging and active sensors on start up.
Auto health check	23	%d, %lu	<enable, period> - enables (1) or disables (0) the internal PLC auto health check of subordinate sensors at the specified period (ms).
Route Messages	24	%d	<sensor id> - sets the sensor for which non-PLC based 7k messages will be routed when sent to the PLC by a remote client.
Report Alarms	25	-	Reports active alarms to the requesting client. Individual alarms are reported as successive 7k records (c.f. record # 11002 of [1]).
Logging setup	26	%lu, %lu or none	<file size, overlap> - used to either set or retrieve the PLC logging setup. When setting, the maximum file size and record overlap between successive log files are specified. Retrieval solicits the logging setup status record and does not require any supplied parameters.
Reserved	27	-	Reserved for RESON internal use.
Time message enable	28	%lu	Enables (1) or disables (0) the emission of an approximate one-minute time message (record #7400; ICD vol. I) to all connected hosts. Note: this is a global PLC function affecting all connected hosts.

Table C.2 — RESON PLC Status Messages

Status	ID	Format	Description
Version	0	%0.4f	<version> - PLC version, interpreted as X.YY.ZZ where X, Y and Z respectively represent the major, minor and maintenance version numbers.
Modules	1	Modules: %d N x (, Module = %d, %s, %.4lf	<N, N x (module #, file name, version)> - Number of modules (N) and details of each module supported; includes the module number, name and version number for each. Notes: • The format specifier is not printf compliant and is advisory only. • The Version field is to be interpreted per version format of "Version" status message (ID 0), above.
Run list	2	Sensors: %d N x (,Sensor = %d, %d, %d, %d, %d, %d, %s)	<N, N x (sensor index, sensor id, sensor type, port, media, subsystem id)> - specifies the active run list for the sensor suite. Note: the format specifier is not printf compliant and advisory only.
File details	3	Logging = %d, %s, %.3lf	<enable flag, file name, file size> - specifies the current state of logging (0 – inactive, 1 – active) and, if active, the current fully qualified file name on the PLC system and its size in Mbytes.
Sensor reply	4	PulseFiles: %d, N x (, %s)	<N, N x (pulse name)> - Provides a list of the pulse names. In the case of the EdgeTech FS-DW, the pulse names are the pulse files. Note: the format specifier is not printf compliant and advisory only.
Logging setup	5	LoggingSetup = %lu, %lu, %s	
Health	6	Sensors: %lu N x (, %d, %d)	<number of sensors, N x (sensor id, health flag)> - health status for each sensor is provided. The health flag is either healthy (1) or unhealthy (0).
Shutdown	7	-	The PLC will send this message to all connected clients whenever it is shutting down. Clients may use this message to help maintain an accurate connection status.

In addition to the above identifiers, the following are reserved for future use:

0 to 5,000 — General system status messages

Table C.3 — RESON PLC Alarm Messages

Alarm	ID	Format	Description
All clear	0	-	No system alarms are active at this time.
Disk space critical	1	%s	PLC system's disk is at or below critical remaining available space (usually = 10% of capacity).
Sensor not responding	2	%d, %s	<sensor id, description> - a given sensor has not responded for a predefined period.
Ping dropped	3	%d, %ld, %ld	<sensor id, start ping number, stop ping number> - specified ping number or range of pings have been dropped by the subsystem. Note: -1 if field is invalid. E.g., "0, 200, -1": sensor (id 0) reports that ping 200 was dropped.
Sensor reported failure	4	%d, %d, %d, %s	<sensor id, cause, internal code, description> - an internal sensor error has been detected. An internal code of -1 means that the sensor code is unknown. Causes are: 0 — Unknown or unspecified 1 — Internal error code 2 — Service required 3 — Temperature out of range 4 — Humidity out of range
Sensor unhealthy	5	%d, %s	<sensor id, description> — the specified sensor has failed its health check.
Sensor failed to start	6	%d, %s	<sensor id, description> — the specified sensor has failed to correctly start up.
Sensor failed to shutdown	7	%d, %s	<sensor id, description> — the specified sensor has failed to correctly shut down.
Sensor failed to load data	8	%d, %s	<sensor id, description> — the specified sensor has failed to provide acquired data to PLC for logging.
Sensor failed to set time	9	%d, %s	<sensor id, description> — the specified sensor has failed to set the system time.
Sensor failed to route 7k message	10	%d, %d, %s	<sensor id, client index, description> — the specified client failed to route non-PLC 7k message to the specified sensor.
Sensor command failed	11	%d, %d, %s	<sensor id, client id, description> — the command addressed to the specified sensor by the given client failed to be invoked.
Sensor status retrieval failed	12	%d, %s	<sensor id, description> — the PLC failed to retrieve status information for specified sensor.
Reset failed	13	%d, %s	<sensor id, description> — the PLC failed to reset the specified sensor.
Invalid command	14	%d, %d, %s	<client index, command id, description> — the specified command from the given client was invalid thus ignored.
RESON Hasp key missing	15	"%s"	This alarm will be raised whenever the PLC has detected that no valid RESON HASP key is present on the system — the PLC will terminate sometime thereafter

In addition to the above identifiers, the following are reserved for future use:

0	to 5,000	— General alarm messages
10,000	to 11,000	— QC-based alarm messages

Table C.4 — RESON PLC Generic Command Formats

Command	Description
“Range = %.2f”	Sets the range of a sensor in meters.
“Pulse = %s”	Sets a specified pulse by name; nomenclature is specific to the sensor or subsystem.
“Ping = %d”	Enables or disables ping transmission.
“TxPower = %.2f”	Sets the transmission power level as a percentage of full power.
“RxGain = %d”	Sets a specified gain setting; values are dimensionless.
“Duration = %.2f”	Specifies a given pulse repletion rate in milliseconds.
“PulseFiles”	Requests a status reply message indicating the available pulse(s) by name.
“DefaultState”	Requests a sensor be reset to its default state. This command is optional and may be ignored by the sensor.