File formats

This chapter describes the file formats supported by the EK60 Scientific echo sounder.

Topics

Numeric type definition

In order to describe the data type formats, common ”C” structures are used to represent individual data blocks.

Table 7 The size of the various “C” types

char	8-bit integer
WORD	16-bit unsigned integer
short	16-bit integer
Int	32-bit integer
long	32-bit integer
float	32-bit floating point (IEEE 754)
double	64-bit floating point (IEEE 754)
DWORDLONG	64-bit integer

Raw data format

The ***.raw** file may contain one or more of the following datagram types:

- Configuration
- NMEA
- Annotation
- Sample

Every ***.raw** file begins with a configuration telegram. A second configuration datagram within the file is illegal. The data content of the **Configuration** datagram of an already existing file cannot be altered from the EK60. **NMEA**, **Annotation** and **Sample** datagrams constitute the remaining file content. These datagrams are written to the ***.raw** file in the order that they are generated by the EK60.

In the following descriptions, the information after the "//" characters are comments to that line.

Note

Strictly sequential time tags are not guaranteed.

Topics

Data encapsulation

A standard encapsulation scheme is used for all data files. Each datagram is preceded by a 4 byte length tag stating the datagram length in bytes. An identical length tag is appended at the end of the datagram.

Format

```
long Length;
struct DatagramHeader
   {
   long DatagramType;
   struct {
      long LowDateTime;
      long HighDateTime;
   } DateTime;
   };
- -
< datagram content
- -
long Length;
};
```

Description

All datagrams use the same header. The datagram type field identifies the type of datagram. ASCII quadruples are used to ease human interpretation and long term maintenance; three characters identify the datagram type and one character identifies the version of the datagram.

The *DateTime* structure contains a 64-bit integer value stating the number of 100 nanosecond intervals since January 1, 1601. This is the internal "*filetime*" used by the Windows NT operating system. The data part of the datagram contains any number of bytes, and its content is highly datagram dependent.

Common computers fall into two categories:

- Intel based computers write a multibyte number to file starting with the LSB (Least Significant Byte).
- HP, Sun and Motorola do the opposite. They write the MSB (Most Significant Byte) to file first.

The byte order of the length tags and all binary fields within a datagram is always identical to the native byte order of the computer that writes the data file. It is the responsibility of the software that reads the file to perform byte swapping of all multibyte numbers within a datagram if required. Byte swapping is required whenever there is an apparent mismatch between the head and the tail length tags. Hence, the two length tags may be used to identify the byte order of the complete datagram.

The Intel processors allow a multibyte number to be located at any RAM address. However, this may be different on other processors; a short (2 byte) must be located at an even address, a long (4 byte) and a float (4 byte) must be located at addresses that can be divided by four. Hence, the numeric fields within a datagram is specified with this in mind.

Configuration datagram

All character strings are zero terminated.

Format

```
struct ConfigurationDatagram {
   DatagramHeader DgHeader // "CON0"
   ConfigurationHeader ConfigHeader;
   ConfigurationTransducer Transducer[];
   };
struct ConfigurationHeader
{
   char SurveyName[128]; // "Loch Ness"
   char TransectName[128];
   char SounderName[128]; // "ER60"
   char version [30];
   char spare [98];
   long TransducerCount; // 1 to 7
};
struct ConfigurationTransducer {
   char ChannelId[128]; // Channel identification
   long BeamType; // 0 = Single, 1 = Split
```

```
   float Frequency; // [Hz]
   float Gain; // [dB] - See note below
   float EquivalentBeamAngle; // [dB]
   float BeamWidthAlongship; // [degree]
   float BeamWidthAthwartship; // [degree]
   float AngleSensitivityAlongship;
   float AngleSensitivityAthwartship;
   float AngleOffsetAlongship; // [degree]
   float AngleOffsetAthwartship; // [degree]
   float PosX; // future use
   float PosY; // future use
   float PosZ; // future use
   float DirX; // future use
   float DirY; // future use
   float DirZ; // future use
   float PulseLengthTable[5];
      // Available pulse lengths for the channel [s]
   char Spare1[8]; // future use
   float GainTable[5];
      // Gain for each pulse length in the PulseLengthTable [dB]
   char Spare2[8]; // future use
   float SaCorrectionTable[5];
      // Sa correction for each pulse length in the PulseLengthTable [dB]
   char Spare3[8];
   char GPTSoftwareVersion [16];
   char Spare4[28];
   };
```

Note

float Gain: The single gain parameter was used actively in raw data files generated with software version 1.3. This was before *PulseLengthTable*, *GainTable* and *SaCorrectionTable* were introduced in software version 1.4 to enable gain and Sa correction parameters for each pulse duration.

NMEA datagram

This datagram contains the original NMEA 0183 input message line; carriage return, line feed and a terminating zero included.

Format

```
struct TextDatagram{
   DatagramHeader DgHeader; // "NME0"
   char Text[]; // "$GPGLL,5713.213,N......"
   };
```

Description and examples

The size of the datagram depends on the message length.

An example **GLL** NMEA position message line is shown below:

```
$GPGLL,5713.213,N,1041.458,E‹cr›‹lf›
```

The information contained in the **VTG** NMEA telegram is not used by the EK60, and this information is thus only written to the ***.raw** data file.

An example NMEA speed message line is shown below:

```
$HUVTG,245.0,T,245.0,M,4.0,N,2.2,K‹cr›‹lf›
```

Related topics

- *Specification of NMEA telegrams* on page 177

Annotation datagram

The annotation datagram contains comment text that you have entered ("dangerous wreck"). The text string is zero terminated. The size of the complete datagram depends on the annotation length. The maximum annotation string is 80 characters.

Format

```
struct TextDatagram {
   DatagramHeader DgHeader; // "TAG0"
   char Text[]; // "Dangerous wreck"
   };
```

Sample datagram

The sample datagram contains sample data from just one transducer channel. It can contain power sample data (Mode = 0), or it can contain both power and angle sample data (Mode = 1).

Format

```
struct SampleDatagram
   {
   DatagramHeaderDgHeader; // "RAW0"
   short Channel; // Channel number
   short Mode; // Datatype
   float TransducerDepth; // [m]
   float Frequency; // [Hz]
   float TransmitPower; // [W]
   float PulseLength; // [s]
   float BandWidth; // [Hz]
   float SampleInterval; // [s]
   float SoundVelocity; // [m/s]
   float AbsorptionCoefficient; // [dB/m]
   float Heave; // [m]
   float Tx Roll; // [deg]
   float Tx Pitch; // [deg]
   float Temperature; // [C]
   short Spare 1
   Short Spare 2
   float Rx Roll [Deg]
   float Rx Pitch [Deg]
   long Offset; // First sample
   long Count; // Number of samples
   short Power[]; // Compressed format - See Remark 1!
   short Angle[]; // See Remark 2 below!
   };
```

The sample data datagram can contain more than 32 768 sample points.

Remarks

1 **Power**: The power data contained in the sample datagram is compressed. In order to restore the correct value(s), you must decompress the value according to the equation below.

$$y = x \frac{10 \log 2}{256}$$

where:

- **x** = power value derived from the datagram
- **y** = converted value (in dB)

2 **Angle**: The fore-and-aft (alongship) and athwartship electrical angles are output as one 16-bit word. The alongship angle is the most significant byte while the athwartship angle is the least significant byte. Angle data is expressed in 2's complement format, and the resolution is given in steps of 180/128 electrical degrees per unit. Positive numbers denotes the fore and starboard directions.