CODAR
OCEAN SENSORS

Time Series File Format

118.01.1609.FF
Sep 30, 2016

Time Series File Format

Time Series files are a collection consecutive time sweeps consisting of received signal power over time. The data is obtained by SeaSondeAcquisition by collecting a consecutive series of measurements from the receiver. The TimeSeries can later be imported by SeaSondeAcquisition to re-examine and further processing into range and cross spectra data.

File Name Format

"Lvl_XXXX_yyyy_mm_dd_hhmmss.ts"
where XXXX = four char code site name
where yyyy = created year i.e. 2016
where mm = created month 01 to 12
where dd = created day 01 to 31
where hh = created hour 00 to 23
where mm = created minute 00 to 59
where ss = created second 00 to 59

File Contents

Format is binary file with a resource indexed file format. The file is composed of keyed blocks of binary data where each block starts with a 4byte character type code followed by a 4 byte long data size of how much data follows.

Big-Endian Byte ordering (MSB first)
IEEE floats & doubles
Two's complement integer values

The file is compose of multiple keys where each key consists of:

- A 4 byte character key type code
- A 4 byte integer of key data size (can be zero)
- Followed by the key data, which is the data size length of bytes.

By convention, Keys with all CAPITALS have subkeys, meaning that the key's data is made up of more keys. When you read a subkey you should read the data in the key as more RIFF keys.

A key may have no data (zero size), in which case the key will contain only the type code and the zero value key size.

When Reading

If you do not recognize the key you should usually skip over it by doing a dummy read of the key's data size.
Do not expect the keys to be in order unless implicitly stated.
Keys can be repeated as needed describing new or changed information.

If you read this file on an Intel Platform or other which uses Little Endian byte ordering the first four bytes will be **VLQA**. In which case, you will need to swap the byte order on each value.

If the file has not finished writing or was closed improperly, the first riff key **AQLV** will contain an invalid data size of 0xFFFFFFFF or ((UInt32)-1). You can then decide if you want to continue reading the partial file or skip it.

SeaSondeAcquisition writes the number of sweeps it's configured for (normally 512) of continuous time sweeps of data. If the sweep information changes or the next sample sweep is not continuous, the file will contain less than this number of sweeps.

When reading and processing consecutive files, you must verify that they are also consecutive in time. When **SeaSondeAcquisition** has to restart the data collection, it will close the file prematurely causing it to have less sweeps than the doppler count.

Data Field type Definitions.

These definitions are a guide to the data structures within file.

Fourcc	4 bytes four character code (example 'xxxx')
Char	1 byte char
Char[64]	64 bytes, string, zero terminated
Char[ ]	[ ]bytes from key data size, zero terminated string
SInt8	1 byte Signed integer -128 to +127 (2s complement)
UInt8	1 byte Unsigned integer 0 to 255
SInt16	2 byte Signed integer -32768 to 32767 (2s complement)
UInt16	2 byte Unsigned integer 0 to 65535
SInt24	3 byte Signed integer (2s Complement)
SInt32	4 byte Signed integer -2Giga to +2Giga (2s complement)
UInt32	4 byte Unsigned integer 0 to 4 Giga
Float	4 byte IEEE single precision floating point
Double	8 byte IEEE double precision floating point
Size32	4 byte Unsigned integer 0 to 4 Gigabytes (tells how much data follows key)

Data Scaling

If the **fbin** key data type is of **fix4**, **fix3**, or **fix2** then the Analog data is auto scaled to an integer value. The scalars used come from the **scal** keys.

The default format used by **SeaSondeAcquisition** is fix2.

If using fixed type **fix4** then:
double IValue = (double)IntegerIValue / (double)0x7FFFFFFF * scalarOne;
double QValue = (double)IntegerQValue / (double)0x7FFFFFFF * scalarTwo;

If using fixed type **fix3** then:
double IValue = (double)IntegerIValue / (double)0x7FFFF * scalarOne;
double QValue = (double)IntegerQValue / (double)0x7FFFF * scalarTwo;

If using fixed type **fix2** then:
double IValue = (double)IntegerIValue / (double)0x7FFF * scalarOne;
double QValue = (double)IntegerQValue / (double)0x7FFF * scalarTwo;

File Contents Layout

The first 4 bytes should read **AQLV**
Below represents the file layout as blocks with the <key> <size> and data structure between the {}.

AQLV Size32 – This is the first key in the file. All data is inside this key.

```
{
   HEAD  Size32
   {
      sign  Size32    – File Signature
      {
         UInt32   nFileVersion     // '2.00'
         UInt32   nFileType        // 'ALVL'
         UInt32   nOwner           // 'CDAR'
         UInt32   nUserFlags       // 0
         Char[64]szFileName        // “SeaSondeAcquisition Time Series"
         Char[64]szOwnerName       // "CODAR Ocean Sensors Ltd"
         Char[64]szComment         // whatever
      }
      mcda  Size32  – Data Time Stamp
      {
         UInt32   Seconds since 1904
      }
      cnst  Size32        – Data Sizes
      {
         SInt32   Number Channels/Antennas
         SInt32   Number Sweeps asked to Record.
         SInt32   Number Samples Per Sweep.
         SInt32   IQ indicator.1 otherwise 2 if sample data is IQ
      }
      swep  Size32  – Receiver settings
      {
         SInt32   Samples Per Sweep
         Double   Start Freq in Hz
         Double   Bandwidth in Hz (negative if down sweep)
         Double   Sweep Rate in Hz
         SInt32   Offset (reserved)
      }

      fbin  Size32    – Type and Format of  data
      {
         Fourcc   Type of Data
                  cviq data is complex Voltages I, Q
         Fourcc   Format Of Analog Array complex Values
```

if **fix2** then data is of int (2byte) use **scal** to adjust
if **fix3** then data is of int (3byte) use **scal** to adjust
if **fix4** then data is of int (4byte) use **scal** to adjust
if **flt4** then data is of IEEE (4byte) floating point
if **flt8** then data is of IEEE (8byte) floating point

}

}

BODY Size32 - Analog sweeps container

{

The following keys are repeated for each Times Series analog sweep up to the number of number of sweep asked to record.

The **indx** key will always precede the **alvl** key

The data format of **alvl** is determined by previous **fbin** key

rtag Size32 - Repeater Position Tag (Optional Key)

{

UInt32 Bearing to Repeater degrees

}

gps1 Size32 - GPS Tag (Optional Key)

{

Double Latitude in Radians

Double Longitude in Radians

Double Altitude in Meters

SInt32 TimeStamp. Seconds from 1904

}

indx Size32 - Time Series Index

{

SInt32 index number 0 to (Number Sweeps - 1)

}

scal Size32 - Data Scalar for following **all** key contents

{

Double Data Scalar for complex real component

Double Data Scalar for complex imaginary component

}

alvl Size32 - Analog sweep Array

{

Array is (row,col) of [Channels] by [Samples per Sweep] of

I sample followed by **Q** sample pairs of **fbin** format type.

}

END Size32 - zero size key indicating end of time series

}

End Of File

Revision History

First Draft Sep 30, 2016

Copyright and Disclaimer