CODAR
OCEAN SENSORS

Spectrum Analyzer File Format

128.01.1611.FF
Nov 15, 2016

Spectrum Analyzer File Format

Spectrum Analyzer files are a collection consecutive time sweeps consisting of received signal power vs frequency. The data is recorded by SeaSondeAcquisition in the spectrum analyzer mode by collecting a consecutive series of measurements from the receiver.

File Name Format

"SpAn_XXXX_yyyy_mm_dd_hhmmss.sa"

where XXXX = four char code site name
where yyyy = created year ei 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 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 4byte 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 'NAPS'. 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 set (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.

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 Range data is auto scaled to an integer value. The scalars used come from the **scal** keys.

The default format used by **SeaSondeAcquisition** is **flt4** Float which requires no scaling. The values are 4byte IEEE single precision floating point.

If using fixed type **fix4** then:
double real = (double)IntegerReal / (double)0x7FFFFFFF * scalarReal;
double imag = (double)IntegerImag / (double)0x7FFFFFFF * scalarImag;

If using fixed type **fix3** then:
double real = (double)IntegerReal / (double)0x7FFFF * scalarReal;
double imag = (double)IntegerImag / (double)0x7FFFF * scalarImag;

If using fixed type **fix2** then:
double real = (double)IntegerReal / (double)0x7FFF * scalarReal;
double imag = (double)IntegerImag / (double)0x7FFF * scalarImag;

File Contents Layout

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

SPAN Size32 - This is the first key in the file. All data is inside this key.
{
  HEAD Size32
  {
    sign Size32 - File Signature
    {
      UInt32 nFileVersion // '1.10'
      UInt32 nFileType // 'SPAN'
      UInt32 nOwner // 'CDAR'
      UInt32 nUserFlags // 0
      Char[64]szFileName // "SeaSondeAcquisition"
      Char[64]szOwnerName // "CODAR Ocean Sensors Ltd"
      Char[64]szComment // anything
    }
    mcda Size32 - Data Time Stamp
    {
      UInt32 Seconds since 1904
    }
    dbrf Size32
    {
      Double Receiver Power loss reference in dB. Adding this should give roughly dBm.
    }
    cnst Size32 - Data Sizes
    {
      SInt32 Number Sweeps to expect in file
      SInt32 Number Frequencies in each sweep
    }
    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 Number of sweep averaged
      SInt32 Antenna Channel recorded.
    }
    fbin Size32 - Type and Format of data
    {

Fourcc Type of Data
real indicating amplitude only data
Fourcc Format of Data
fix2 data is of int (2byte) use **scal** to adjust
}
}
BODY Size32 – sweeps container
{
The following keys are repeated for each sweep up to the number of sweeps.
indx Size32 – Range Series Index
{
SInt32 index number 0 to (DopplerCells – 1)
}
scal Size32 – Data Scalar for following **afft** and **ifft** key contents
{
Double Data Scalar for amplitude.
}
data Size32 – The sample data for a single sweep
{
Contains an array of Number Frequencies of measured amplitude scaled from dB to SInt16 value.
}
}
END Size32 – zero size key indicating of range series
}

End Of File

Revision History

First Draft Nov 15, 2016

Copyright and Disclaimer