Range Series File Format

114.01.1609.FF
Aug 26, 2016

Range Series File Format

Range Series files are a collection of consecutive time sweeps consisting of received signal power over range. The data is obtained by SeaSondeAcquisition performing a Fast Fourier Transform on the collected Time Series data. Range Series can be imported by SeaSondeAcquisition application to produce cross spectra.

File Name Format

"Rng_XXXX_yyyy_mm_dd_hhmmss.rs"
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
extension is ".rs"

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 key blocks 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 one or 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 or skip 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 or other platform, which uses Little-Endian byte ordering, the first four bytes will be **TFQA**. In this case, you will need to swap the byte order on each field except strings.

If the file has not finished writing or was closed improperly, the first riff key **AQFT** 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.

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 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)0x7FFFFF * scalarReal;
double imag = (double)IntegerImag / (double)0x7FFFFF * 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 **AQFT**
Below represents the file layout as blocks with the <key> <size> and data structure between the {}.

```
AQFT  Size32  – This is the first key in the file. All data is inside this key.
{
   HEAD  Size32
   {
      sign  Size32   – File Signature
      {
         UInt32   nFileVersion    // '1.00'
         UInt32   nFileType       // 'AQFT'
         UInt32   nOwner          // 'CDAR'
         UInt32   nUserFlags      // 0
         Char[64]szFileName       // "SeaSondeAcquisition"
         Char[64]szOwnerName  // "CODAR Ocean Sensors Ltd"
         Char[64]szComment        // whatever
      }
      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 Channels/Antennas
         SInt32   Number Range Cells
         SInt32   Number Sweeps asked to collect in file.
         SInt32   IQ Indicator (reserved)
      }
      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   Start Range Bin from FFT (zero based)
      }
      fbin  Size32   – Type and Format of  data
```

{
Fourcc Type of Data [**cviq**,**dbra**]
if **cviq** then data is complex Voltages I, Q
if **dbra** then data is complex Power dBm, Phase Deg
Fourcc Format Of Range 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 – Range sweeps container
{
The following keys are repeated for each Range Series up to the number of DopplerCells.
The **indx** key will always precede the **afft** key
The data format of **afft** 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 – 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 complex real component
Double Data Scalar for complex imaginary component
}
afft Size32 – Range Array Positive range
{
Array is (row, col) of [Channels] by [RangeCells] of
Complex real, imaginary pairs of **fbin** format type.
}
i**fft Size32** – Range Array Negative range.

```
        {
            Contains the image freq of the FFT in reverse order
            Array is (row, col) of [Channels] by [RangeCells] of
            Complex real, imaginary pairs of fbin format type.
        }
    }
    END Size32 - zero size key indicating of range series
}
```

End Of File

Revision History

First Draft Sep 27, 2016

Copyright and Disclaimer