CODAR
OCEAN SENSORS

SeaSonde Radial Site Release 6
Range Series File Format

Apr 19, 2009 ©Copyright CODAR Ocean Sensors

Range Series files are a collection 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. The Range Series can be later imported by SeaSondeAcquisition or CSPro 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	ei 2009
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 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 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 or other platform, which uses Little-Endian byte ordering, the first four bytes will be 'TFQA'. In which case, you will need to swap the byte order on each value 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.

Data Type Definitions

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

Each subkey contents is inside of {} brackets
Each key data content is indented in order after key.

// Begin File. The first 4bytes should read 'AQFT'

'AQFT' Size32 - This is the first key in the file. All data is inside this key.

```
{
    'HEAD'  Size32
    {
        'sign'  Size32              // File Signature
        UInt32   nFileVersion       // file code    '1.00'
            UInt32      nFileType                   // file type    'AQFT'
            UInt32      nOwner// ownertype    'CDAR'
            UInt32      nUserFlags                  // whatever     0
            Char[64]    szFileName                  // "SeaSondeAcquisition"
            Char[64]    szOwnerName                 // "CODAR Ocean Sensors Ltd"
            Char[64]    szComment                   // whatever
        'mcda'  Size32              // Data Time Stamp
            UInt32      nDateTime                   // MacOS seconds from 1904
        ‘dbrf’ Size32
            Double      Receiver Power loss reference in dB. Adding this should give roughly dBm.
        'cnst'  Size32              // Data Sizes
            SInt32      Number Channels
            SInt32      Number Range Cells
            SInt32      Number Doppler Cells
            SInt32      1 source was I only, 2 source was I&Q
        'swep'  Size32   // Acquired from SeaSondeController App
            SInt32      Samples Per Sync
            Double      Start Freq in Hz
            Double      BandWidth in Hz
            Double      Sweep Rate in Hz
            SInt32      Start Range Bin from orig FFT (zero based)
        'fbin'  Size32              // Type 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 integer (2byte) use 'scal' to adjust
             if 'fix3' then data is of integer (3byte) use 'scal' to adjust
             if 'fix4' then data is of integer (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
    {
        // 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 Posistion 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

    'indx'  Size32
            SInt32      Current RangeSeries index number 0 to (DopplerCells - 1)
        'scal'  Size32  Data Scalar for following ‘afft’ key contents
            Double      Data Scalar for complex real component
            Double      Data Scalar for complex imaginary component
        'afft' Size32  Range Array
          // Array Size is (row, col) or [Channels] by [RangeCells] of
          // Complex real, imag pairs.
        'ifft' Size32  Range Array Negative frequencies. (Optional Key)
            // Contains the image freq of the FFT in reverse order
            // 'afft' rangecell 0 corrisponds to 'ifft' rangecell (RangeCells-1)
          // Array Size is (row,col) [Channels] by [RangeCells] of
          // Complex real,imag pairs
        // Repeat of previous keys for number of DopplerDells
    }
    'END' Size32        // zero size key indicating of range series
}
// End Of File
```