The **R**oproc **F**ormat **F**ile

A dedicated file format for vectorial data processing

Patrick ROBERT, CNRS/LPP

Version 2.2, February 2011

First Version 1.0, November 2004

Contents

Chapter 1

INTRODUCTION TO RFF FILES

1.1 HISTORY

The Roproc Format File (RFF) is an ascii files containing scientific data in a dedicated format used by the Roproc data processing procedures [1], [2]. This format is a basic tool allowing communication between different Roproc commands. A primitive version of the RFF format has been defined many years ago to store spatial data issued from magnetospheric spacecraft, such as GEOS-1, GEOS-2, and updated more recently for CLUSTER constellation, CUSP rocket or Double Star Probe. The experiments providing these data were mainly magnetometer, waveform unit with a precise frequency sample, orbit or trajectory measurement.

After the apparition of a general Cluster Exchange Format (CEF) in 2004 [3], in the framework of the Cluster Active Archive [4], a new version of the Roproc_Format_File (V 2.0) has been produced where nomenclature and names of keywords have been changed to be close to those of the CEF format. At this occasion, the RFF has been extended to cover most kind of data, but always keeping a look very simple and easily readable.

The more important advantage of the actual RFF V 2.2 is that into all the Roproc procedures, which process data issued from magnetometer, wave experiment, or trajectory positions, or other kind of resulting data such as spectra, spectrograms, wave polarization, and so on, the input and the output of all the Roproc commands can be RFF files.

The RFF V 2.2 files are also used to process the STAFF-SC data which are provided to the Cluster Active Archive. Then, decommuted waveform (DWF) issued directly for telemetry, calibrated waveform (CWF) and complex spectra (CS) are computed in RFF file and then converted accross a simple filter into the CEF required format.

1.2 CLASSES OF RFF FILES

1.2.1 Class presentation

The RFF file format can be used to store various kind of data; nevertheless, to get a quick look of the idea of its contents, it is useful to define various "classes" of RFF. A class is relative to the general form of the content. For instance, in plasma or spatial experiments, a most common kind of data is a time series of a vector, i.e. a vector indexed by time. This is what one call a *VecTime* class (see 5.1). Sometimes, the vectors measured can depend by another parameter that the time, for instance a scalar value (as a temperature) or a vector value (as a position in space). These cases correspond to the *VecSca* and the *VecVec* class.

In the same idea, one can found a measured matrix rather than a measured vector. This matrix can depend on the time, or on a scalar value, or on a vector. So one can define the corresponding RFF classes *MatTime* class, *MatSca*, and *MatVec*. In the same way, one also encounter very simple data as a scalar value (as a density), depending on the time, of another scalar value, or of a vector. This leads to the *ScaTime*, *ScaSca* and *ScaVec* classes.

In spatial data, one can often encounter that one call most commonly a "waveform data". This is a time series of vectors measurements, regularly sampled with a fixed and knowed frequency. This frequency is a constant value, depending on the hardware, and knowed with a good accuracy. So, there is no need to measure the time of each vector, but only the starting time of a block of vectors. The lengh of this block, i.e. the number of vector contained in a block, depend on the accuracy of the frequency sample. The more this frequency is accurate, the more one can increase the number of vectors into a block, keeping the possibility to rebuild the time of each vector with a good accuracy.

So, a waveform data is viewed into a RFF file as a *MatTime* class : a matrix, or a group of N vectors, indexed by a time value which corresponds to the time measurement of the first line of the matrix, i.e. the first vector. Thus one can define a *WaveForm* class as a logical subclass of the most general *MatTime class*.

In spatial data, one often use that one call a "spectrogram", or "dynamic spectra". This is a time series of spectra, generally deduced by successive Fourier Transform of consecutive windows picked up from waveform data. These series of spectra are generally plotted as an image, where X axis is the time, and Y axis is the frequency ; one also call this a "time-frequency" plot. In this image, each pixel has a constant $\Delta t \cdot \Delta f$ time-frequency resolution, equal to 1 if there is not overlapping from a spectra to another. Note that if the initial waveform is a vector of N components (for instance 3 cartesian components), one can deduce N spectrograms, or N images. Thus a group of 3 spectrograms is not so different that a waveform of a 3-D vector: at each data block corresponds a spectrum, that is to say a frequency series regularly spaced by a δf ecrement, rather than a time series regularly spaced by a δt increment.

So one can define the *Spectrogram* class, as also a logical subclass of *MatTime* class. Each column of the matrix corresponds to the spectrum of the component of the initial waveform, while each line corresponds to a frequency. Each spectrum has a time equal to the time of the corresponding block in the waveform data.

At last, the *Image* class corresponds to a series of images data, depending on the time, or of a scalar parameter, or on a vectorial parameter. It is also a subclass of MatTime, MatSca or MatVec main class.

The table 1.1 given hereafter summarize the various classes used in the RFF files.

1.2.2 Examples

A classic example of *VecTime* RFF class is the CLUSTER/FGM data : it is a time series of the Bx, By, Bz components of the DC-magnetic field vector recorded onboard the 4 CLUSTER spacecraft. Each vector measurement has an ISO date-time index, but the 3-D vectors are not necessarly regularly spaced in time. The *VecTime* class is a very common class, that one can encounter on many field measurement, especially those with not too short time sample; indeed, for high sample frequency, this format become heavy in size, and generally one can then prefer the *WaveForm* class format, used for instance for CLUSTER/STAFF-SC, DSP/STAFF-SC, and other waveform unit experiment. Note that a *WaveForm* RFF file can be easyly converted to *VecTime* file (with only a rise on the file size), and reciprocally a *VecTime* file can be converted in *WaveForm* file only if the time sample of the initial file is constant and knowed with a good accuraccy. This allows, for instance, to compute spectrograms or time-frequency polarization of data such as CLUSTER/FGM, by using the RCL commands initially builded for waveform data.

Table 1.1: *Various classes of RFF files*

ScaTime	series of time indexed scalars
ScaSca	series of scalar indexed scalars
ScaVec	series of vector indexed scalars
VecTime	series of time indexed vectors
VecSca	series of scalar indexed vectors
VecVec	series of vector indexed vectors
MatTime	series of time indexed matrix
MatSca	series of scalar indexed matrix
MatVec	series of vector indexed matrix
WaveForm	subclass of MatTime where Matrix contains vectors regularly spaced in time
Spectrogram	subclass of MatTime where Matrix contains vectors regularly spaced in frequency
Image	subclass of MatTime, MatSca or MatVec where Matrix is viewed as an image

The *Spectrogram* class file is almost always associated with the *WaveForm* data, but it can be also the format of native data, when the spectra are computed onboard the spacecraft. The trajectory data of spacecraft are generally stored in *VecTime* files, but for instance for the 4 CLUSTER spacecraft, it can be usefull to store simultaneously the 4 positions in a *MatTime* file, with a (4,3) matrix.

The *VecVec* class corresponds rarely to measured data, because the index data is not the time, but a vector parameter. But this class can be used to store a field line data, for instance, which can be computed from a magnetic field model at a given time. Indeed, a field line data is a 3-D magnetic field vector versus 3-D position in space. In the same way, the *ScaVec* class can be used to store the temperature or the density of a point in space, always for a fixed time.

Example of various class of RFF files are given in chapter 5.

Chapter 2

RFF FILE STRUCTURE

2.1 GENERALITIES: DATA and METADATA

To have an easily readable data file, the first thing to do is to precede the data themselves by a metadata field, which describes what the data are : the mission name, sometimes also called project name (CLUSTER), the experiment which provided these data (for CLUSTER, for instance, FGM, STAFF, EFW ...), the spacecraft or observatory name (CLUSTER-4, Tango, ground station, etc.), the form of data (scalar, vector, matrix, tensor) and so on. The metadata are also very useful to build standard plots, with correct titles and label axes. So the RFF file structure is roughly split in two parts: the *metadata* and the *data*. File can also contain *comments* to help the readability.

2.2 KEYWORDS

To use this principle, it is necessary to define *keywords*, both for the metadata and the data. Only 4 keywords are used : The `START` and `END` `keywords` define a group of parameters, variables or data. The `PAR` keyword defines a parameter whose value is constant over the given RFF file (for instance the observatory, or the mission name). The `PAR` keyword is followed by a pre-defined list of possible parameters, which can be mandatory or optional (see 2.5). The `VAR` keyword is a user's variable, defined according needs of the user to define single data values (see 2.6).

2.3 DATA GROUPS : START AND END KEYWORD

The RFF file contains at least two data group: metadata and data; these groups can be also split into a group of *mandatory parameters* (mission name, experiment name ...), and *optional parameters* (discipline name, experiment description, etc.). Each group must begin with a `START` keyword and terminate by a `END` keyword. The RFF file itself must begin with a `START` and terminate with a `END` keyword. For instance, one can have a Roproc Format File structure as schown in table 2.1.

The words `START/END` are keywords, while `METADATA`, `DATA`, `MANDATORY_PARAMETERS`, and `OPTIONAL_PARAMETERS` are group names. The description of the metadata and the data parameters is given in chapter 3. Chapter 5 gives examples of RFF files for 2 characteristic experiments of the Cluster mission. Note that keywords names, parameter names and group names are case sensitive (always in upper case).

Table 2.1: *General structure of a RFF file*

```
START ROPROC_FORMAT_FILE
# any comments

     START METADATA
          START MANDATORY_PARAMETERS
          PAR ...
          END MANDATORY PARAMETERS
          START OPTIONAL_PARAMETERS
          PAR ...
          END OPTIONAL_PARAMETERS
     END METADATA

     START DATA
          START CONSTANT_DATA
          PAR ...
          END CONSTANT_DATA
          START INDEXED_DATA
          PAR ...
          END INDEXED_DATA
     END DATA

END ROPROC_FORMAT_FILE
```

2.4 ALLOWED GROUP NAMES

Group names defined by a START keyword can only be taken in the following list:

`START ROPROC_FORMAT_FILE`	: the first and mandatory line of the RFF file
`START METADATA`	: begin description of data by parameters
`START MANDATORY_PARAMETERS`	: begin mandatory parameters of the metadata
`START OPTIONAL_PARAMETERS`	: begin optional parameters of the metadata
`START DATA`	: begin effective data
`START CONSTANT_DATA`	: constant data for a given file, given by VAR
`START INDEXED_DATA`	: the body of the data, varying with the index value

Of course, any `START` group must terminate by a `END` group. First and last line of the RFF file must be a `START ROPROC_FORMAT_FILE` and `END ROPROC_FORMAT_FILE`.

2.5 PAR KEYWORD

All parameters are declared by a `PAR` keyword, and must be followed by the parameter name, the parameter type (into brackets), and the parameter value. The value is constant for a given RFF file. Syntax is:

```
PAR parameter_name  (parameter_type):  parameter_value
```

For instance:

```
PAR MISSION_NAME    (STR): CLUSTER
PAR EXPERIMENT_NAME (STR): FGM
PAR DATA_FORM       (STR): Vector
PAR DATA_DIMENSION  (INT): 3
PAR DATA_TYPE       (STR): FLT
```

The parameter type is given into brackets; allowed values are {STR ; INT ; FLT ; DBL ; CMP ; TXT}. This is an abbreviation for character string, integer, float, double precision, complex. The TXT corresponds to a text, i.e. a series of character string lines, the first line beginning with the character "{" and the last line ending with the character "}". Examples can be found in section 3.5 or in chapter 5. This field being a parameter, it is recommended that the text be short, and limited to a maximum of 20 to 30 lines. As string parameter, the lengh of each line is limited to 128 character.

Note that the type of `DATA_TYPE` parameter is a character string, and the type of the data, defined by the value of the `DATA_TYPE` parameter, is in this example a float number. Parameters are pre-defined, and many of them are mandatory for the computer reading program and a good meaning of the data. White space characters are used as delimiters, and many consecutive blanks can be used to indent the file, or to align the various fields to increase the readability within any text editor. Mandatory parameters are given and detailed in section 3.4.

`PAR` keyword must be present only into a `START / END` group, and only in the `START METADATA / END METADATA` header group.

2.6 VAR KEYWORD

The `VAR` keyword is close of the `PAR` keyword, but is dedicated to the user, who can create new variables, used for instance for auxiliary data, given one time for a group of data. Examples are given in chapter 4 and 5. All variables are declared by a `VAR keyword`, and must be followed by the variable name, the variable type, optionally the variable unit, and the variable value. Syntax is:

```
VAR variable_name  (variable_type) [, u=units] :  variable_value
```

For instance:

```
VAR SPIN_FREQUENCY  (FLT), u=Hz :  0.249516E+00
```

`VAR` keyword must be present only into a `START / END` group, and only in the `START CONSTANT_DATA / END CONSTANT_DATA` header group.

2.7 COMMENTS IN FILES

Each line beginning with a `#` character is a comment, and can be ignored by a computer reading program. These comments are only useful to help the readability of the files under any text editor. In the same way, blank lines can be added, and will be ignored by the reading software. Comments lines or blank lines can appear anywhere in the RFF files, excepting at the beginning or at the end of the file, or inside an indexed block data (see section 4.3). Note that the first line must be `START ROPROC_FORMAT_FILE` and the last line must be `END ROPROC_FORMAT_FILE`.

Chapter 3

METADATA STRUCTURE

3.1 GOAL

The metadata must be a set of parameters that completely describes the data. The data themselves are given in a separate `START/END` group. Metadata are made of a suit of mandatory and optional parameters , defined by a `START/END MANDATORY_PARAMETER` group at the beginning of the file, and optionally followed by a `START/END OPTIONAL_PARAMETER` group (see 2.3).

At the head of the mandatory group, one can find the parameters associated to the file itself: the *file parameters* (file name, version number of the software used to create the RFF file, creation date). Then one finds parameters associated with the *description of the mission* and the experiment (mission name, experiment name, etc.). After that, one finds the parameters associated with the *data description* and the *index description* (see 3.4.5 and 3.4.3). At the end of the group are given the parameters associated with the structure of the *indexed data block*, i.e. the indexed data block format and the number of indexed data blocks.

3.2 DATA FORMAT CONSIDERATIONS

The data themselves can be considered within a mathematical aspect: generally, one measures a quantity which may be scalar (temperature for instance), vector (magnetic field), or a matrix (or 2D image), and why not a tensor of any dimension. This can be inquired by a `DATA_FORM` parameter, and other data description parameters (see hereafter).

The measured (or computed) data vary with a quantity, which can be the time, or the position of the rocket or spacecraft, or another thing that one call an *index*. This index can be itself a scalar (data time epoch in ISO format), a vector (the position of the spacecraft), and why not a matrix or a tensor (but rarely used). So the index has also an `INDEX_FORM` parameter, and other associated description parameters. All parameters are mandatory, for example the dimension of the vector (if the data have a vector data form), the data units, etc.

3.3 METADATA GROUPS

As said above, there are only two allowed groups of metadata: one containing the mandatory parameters, and another one containing optional parameters. Then, the structure of the full metadata group is as below:

```
START METADATA

     START MANDATORY_PARAMETERS
     PAR ...
     END MANDATORY_PARAMETERS

     START OPTIONAL_PARAMETERS
     PAR ...
     END OPTIONAL_PARAMETERS

END METADATA
```

The list of mandatory parameters is fixed, and all must be defined. The list of optional parameters is free. The group of optional parameters can be empty. Some of optional parameters are nevertheless recommandes, see 3.5

3.4 MANDATORY PARAMETERS DESCRIPTION

3.4.1 File parameters

An example is given below:

```
PAR FILE_NAME                (STR): CLU4_STASC_NBR_WFL1_20010923.rff
PAR FILE_CLASS               (STR): WaveForm
PAR FILE_FORMAT_VERSION      (STR): Roproc_Format_File V 2.2
PAR FILE_CREATION_DATE       (STR): 2007-10-13T21:36:34.000Z
```

- FILE_NAME

The original name of the file when it was created.

- FILE_CLASS

One of the allowed classes defined in the table 1.1.

- FILE_FORMAT_VERSION

This is required for the software used to read the RFF file.

- FILE_CREATION_DATE

The creation date is given in ISO format. It is the creation date of the RFF file, not the production date of the original data.

3.4.2 Mission and Experiment parameters

An example is given below:

```
PAR MISSION_NAME             (STR): CLUSTER
PAR OBSERVATORY_NAME         (STR): Tango
PAR OBSERVATORY_NUMBER       (INT): 4
PAR EXPERIMENT_NAME          (STR): STAFF-SC
PAR EXPERIMENT_MODE          (STR): NBR
PAR INSTRUMENT_TYPE          (STR): Search Coils
PAR MEASUREMENT_TYPE         (STR): B AC Magnetic field waveform
```

- MISSION_NAME

A character field, without blank, recommended in upper case.

• OBSERVATORY_NAME

A character field, without blank. Sometime same as MISSION_NAME

• OBSERVATORY_NUMBER

An integer. Can be 0 if not defined.

• EXPERIMENT_NAME

The given experiment can contains sub-systeme names, as STAFF-SC here; for instance the STAFF experiment contains two subsystems: STAFF-SC and STAFF-SA ; STAFF-SC experiment provides waveform data, while the STAFF-SA subsystem provides cross-spectral matrix of E and B fields. The STAFF-SC part is considered as an unique experiment name.

• EXPERIMENT_MODE

The EXPERIMENT_MODE (here NBR mean "Normal Bit Rate") can be set to "None" if the experiment has no different running mode.

• INSTRUMENT_TYPE

Give information about the type of experiment hardware (a few words).

• MEASUREMENT_TYPE

Give information about what is measured (a few words).

3.4.3 Index parameters

The parameters associated with index are given below, with the list of allowed values. These parameters are mandatory to define the form, type and structure of the index of each block data.

For example one can have:

```
PAR INDEX_LABEL              (STR): Time
PAR INDEX_TYPE               (STR): STR
PAR INDEX_UNITS              (STR): ISO_TIME
PAR INDEX_FORMAT             (STR): (a27)
PAR INDEX_FORM               (STR): Scalar
PAR INDEX_DIMENSION          (INT): 1
PAR INDEX_PROPERTIES         (STR): Regularly Spaced
```

• INDEX_LABEL

This is a free character string, rather short, defining the nature of the index, and which can be used to label the X axis of a standard plot.

• INDEX_TYPE

Allowed values are { STR ; INT ; FLT ; DBL ; CMP }. An index can have a character string type (as Date/Time epoch in ISO format), but cannot be encoded in a text format.

• INDEX_UNITS

The units of the given index, whose abbreviation is known, for instance "nT", "mV" or "km". If units is not obvious, the conversion factor to transform it into the SI system should be given in a comment line. The number of units must be equal to the dimension of the INDEX_DIMENSION parameter. If the INDEX_FORM is scalar, as ISO-time, the INDEX_DIMENSION is equal to 1, and the INDEX_UNIT is equal to the character string "ISO_TIME".

• INDEX_FORMAT

This is the format, in FORTRAN or IDL convention, to read the index field in a block data. The format to read an entire block must be a concatenation of the INDEX_FORMAT, INDEX_EXTENSION_FORMAT, and DATA_FORMAT (see hereafter extended index).

• INDEX_FORM

As DATA_FORM, allowed values are { Scalar ; Vector ; Matrix or Image ; Tensor }. The scalar form is used to code the date/time epoch in ISO format, or any else time format using a single value (Julian second, etc.), according INDEX_TYPE parameter (STR, INT etc.). One can also imagine a date/time epoch define as a vector form, of dimension equal to 7, and of integer type, with index(1-7)=(year, month, day, hour, min, sec, millisec).

• INDEX_DIMENSION

If INDEX_FORM is scalar, the INDEX_DIMENSION parameter must be equal to 1. If index form is vector, the INDEX_DIMENSION is a single integer corresponding to the dimension, or number of components, of the vector. If index form is matrix, INDEX_DIMENSION is two integers; first is the number of columns (as the dimension of a simple vector), second is the number of lines. If form is tensor, INDEX_DIMENSION is a series of integer values corresponding to the dimensions of the tensor.

Note that an index has often a scalar or a vector form (date, position), and is rarely a matrix or a tensor.

• INDEX_PROPERTIES

Allowed values are {Regurlarly Spaced, Irregularly Spaced, Roughly Regularly Spaced, Undefined }

This is useful for time indexed data. If, for instance, a vector data series is given with a regular time interval, further tests can be done by the user to check if there is no data gaps. If this time is not constant (vector data irregularly spaced), and unpredictable, no special test can be done. If this time interval is not accurately constant, for instance if the index format is not enough precise, the user can compute an average frequency sample, which can be used to identify data gaps, but this average frequency can be not enough precise to predict with accuracy the time of the next or previous vector data.

3.4.4 Index_Extension parameters

One can wish than the data block could be preceded by a certain number of various data (block quality, status of any experiment, and so on), associated to the index. So, it is possible to define an extension of the index, to expand the header of the data block, which can thus contain anything else more than the only one index. Then the first field is the true index, and the rest of the field is auxiliary data, called "index extension".

For instance, if one have a time depending matrix data as above, and one want to add some auxiliary data to the matrix, one can define :

```
PAR INDEX_EXTENSION_LABEL     (STR): Status ; Phase_angle
PAR INDEX_EXTENSION_TYPE      (STR): STR ; DBL
PAR INDEX_EXTENSION_UNITS     (STR): None ; degree
PAR INDEX_EXTENSION_FORMAT    (STR): (a11,1x,f7.2)
PAR INDEX_EXTENSION_LENGTH    (INT): 19
```

And have an indexed data block as :

```
2001-09-23T00:00:01.353074Z 00000100000  316.29
80f8 8520 800c 0
814a 850c 800d 0
8197 84f4 800a 0
81e4 84d6 8008 0
8231 84b1 8010 0
8275 848e 800f 0
```

In this case, the true index is date time epoch, coded as a string of 27 characters, and the extended index includes a status of the experiment, and the phase angle of the x-axes deduced from the sun pulse signal. This 3 quantities are coded in a single character string field, making up the index extension. In the given example, this field have a lenght of 19 characters.

To decode the values of the quantities present in the index extension, one use the value of the INDEX_EXTENSION_FORMAT parameter, where one can see that the status of the experiment is a string coded with 11 characters, and the phase angle is a float coded in f7.2 format, with a blank between. There is no penalty to extend an index, if the meaning of all auxiliary data words is known, for instance from a comment line. Note that the length of the extension is given by the INDEX_EXTENSION_LENGTH parameter, while the label, type and units of each fiekd of the extended index are given by the corresponding parameters. The number of semi-colon which are the separtors field must be in accordance with the INDEX_EXTENSION_FORMAT value, itself in accordance with INDEXED_EXTENSION_LENGTH value.

- INDEX_EXTENSION_LABEL

This is a free character string, rather short, defining the nature of the index extension. If the index extension contains several fields, each field must be descripted here, separated by semi-colon.

- INDEX_EXTENSION_TYPE

This is the type of the various fields contained in the index extension. If the index extension contains several fields, the type of each field must be defined here, separated by semi-colon.

- INDEX_EXTENSION_UNITS

This is the unit of the various fields contained in the index extension. If the index extension contains several fields, the unit of each field must be defined here, separated by semi-colon.

- INDEX_EXTENSION_FORMAT

This is the format, in FORTRAN or IDL convention, to read the index extension field in a block data. The format required to read an entire block in a single time must be a concatenation of the INDEX_FORMAT, INDEX_EXTENSION_FORMAT, and DATA_FORMAT. During concatenation, a character blank, as ",1x", must be inserted between each of the sub-format above.

• INDEX_EXTENSION_LENGTH

This is the total length, in character, of the extended index. If the user wants to skip the extended index rather to read it, he can replace the INDEX_EXTENSION_FORMAT by a number of blanks corresponding to the INDEX_EXTENSION_LENGTH parameter.

3.4.5 Data parameters

The parameters associated with data are given and described below. Some of them have a limited list of allowed values. These parameters are mandatory to define the form (do not confuse with "format", which is used further), type and properties of the data.
For example one can have:

```
PAR DATA_LABEL                (STR): Bx ; By ; Bz ; Compression Factor
PAR DATA_TYPE                 (STR): INT
PAR DATA_UNITS                (STR): TM_counts ; TM_counts ; TM_counts ; None
PAR DATA_FORMAT               (STR): (24((3(z4,1x),i1),/),(3(z4,1x),i1))
PAR DATA_FORM                 (STR): Matrix
PAR DATA_DIMENSION            (INT): 4 25
PAR DATA_REPRESENTATION       (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM    (STR): SSW6RF
PAR DATA_FILL_VALUE           (INT): -999
```

• DATA_LABEL

It is a free character string parameter which summarizes the nature of the data, for each component. It can be used to label the Y axis of the standard plot. Note that for vector or matrix, each component can have a different label, as in this example. In this case, each label must be separated by a semi-colon ";". The number of labels must be equal to the first dimension of the DATA_DIMENSION parameter.

• DATA_TYPE

Allowed values are {STR ; INT ; FLT ; DBL ; CMP ; TXT} This is an abbreviation for character string, integer, float, double precision, complex and text. The TXT correspond to a text, i.e. a series of character string lines, beginning with the "{" character and ending with the "}" character. This form is rarely used for scientific data, but used for parameters such as EXPERIMENT_DESCRIPTION (see 3.5).

• DATA_UNITS

The units of the given data, whose abbreviation is known, for instance "nT", "mV" or "km". If units is not obvious, the conversion factor to transform it into the SI system should be given in a comment line. The number of units must be equal to the first dimension of the DATA_DIMENSION parameter.

• DATA_FORMAT

This is the format, in FORTRAN or IDL convention, to read the data field in a block data. The format to read an entire block must be a concatenation of the INDEX_FORMAT, INDEX_EXTENSION_FORMAT, and DATA_FORMAT (see hereafter).

• DATA_FORM

It is a mandatory parameter ; allowed values are { Scalar ; Vector ; Matrix or Image ; Tensor }

• DATA_DIMENSION

If DATA_FORM is scalar, the DATA_DIMENSION parameter must be equal to 1. If form is vector, the DATA_DIMENSION is a single integer corresponding to the dimension, or number of components, of the vector. If form is matrix, DATA_DIMENSION is two integers; first is number of lines, second is number of columns. If form is tensor, DATA_DIMENSION is a series of integer values.

• DATA_REPRESENTATION

This parameter is required only for vector or tensor, not for scalar form. Allowed values are {xyz Cartesian, rtp Spherical Polar, rlp Spherical Equatorial, rpz Cylindrical Polar, xy Plane Cartesian, rt Polar Plane}. In case of rtp Spherical option, r is in DATA_UNITS, theta and phi are in radians.

Details of abbreviation are given below:

xyz	Cartesian	x, y, z	Cartesian coordinates
rtp	Spherical Polar	r, θ, φ	θ= polar angle, related to z axis, φ=azimutal angle in xy plane
rlp	Spherical Equatorial	r, λ, φ	λ= latitude, related to xy plane, φ=azimutal angle in xy plane
rpz	Cylindrical Polar	r, θ, z	θ=azimutal angle in xy plane
xy	Plane Cartesian	x, y	Cartesian coordinates
rt	Polar Plane	$r, \theta,$	θ= x,y angle

• DATA_COORDINATE_SYSTEM

This is the coordinate system used to the measurement, also called frame of reference; it can be for instance GSE, GSM, GEO, GEI, SM, etc. If the given system is not well known, it should be described by a comment line.

• DATA_FILL_VALUE

This is an mandatory parameter, even if its value is not used. The fill value is usefull if initial data contain no significant, undefined or missing values. In this case, the value can be set, for convenience, to a constant value, named "fill value" (generally an unexpected value, not physically meaningful, positive or negative). Note that the type of DATA_FILL_VALUE is always STR, even if the data are floating values. This is because this parameter must have a constant type, whatever the data type. For current use, the string value of DATA_FILL_VALUE is converted into the type of the data, in particular for checking if the data value is not a fill value.

3.4.6 Block parameters

An example is given below:

```
PAR BLOCK_NUMBER              (INT): 86321
PAR BLOCK_FIRST_INDEX         (STR): 2001-09-23T00:00:01.353074Z
PAR BLOCK_LAST_INDEX          (STR): 2001-09-23T23:59:51.811447Z
```

These parameters concern the indexed data blocks themselves. They must be coherent with the content of the INDEXED_DATA part of the file.

• BLOCK_NUMBER

This is the number of block data contained in the INDEXED_DATA part of the file.

• BLOCK_FIRST_INDEX

This is the index of the first block encountered in the INDEXED_DATA part of the file.

• BLOCK_LAST_INDEX

This is the index of the last block encountered in the INDEXED_DATA part of the file.

3.5 OPTIONAL PARAMETERS DESCRIPTION

3.5.1 Recommanded optional parameters

Many optional parameters are useful to get a more complete description of the data. Examples hereafter are not mandatory, but could be recommanded.

• TITLE

Useful for a standard plot of the data. Example:

```
PAR TITLE                   (STR): CLUSTER / STAFF-SC / Tango (#4)
```

• SUB_TITLE

Useful also for a standard plot of the data. Example:

```
PAR SUB_TITLE               (STR): TM data in spinning system
```

• DISCIPLINE_NAME

Useful for the field of investigation of the experiment, and keywords for publication research. Example :

```
PAR DISCIPLINE_NAME         (STR): Space and  Magnetospheric Physics
```

• EXPERIMENT_PI_NAME

Useful for ackowledgment for data use, and data expertise. Example :

```
PAR EXPERIMENT_PI_NAME      (STR): Nicole Cornilleau
```

• EXPERIMENT_PI_MAIL

Useful for a contact to get any information on the experiment or on the data. Example :

```
PAR EXPERIMENT_PI_MAIL      (STR): Nicole.Cornilleau@cetp.ipsl.fr
```

• MISSION_DESCRIPTION

A short text to summarize the mission or the project. Eaxample :

```
PAR MISSION_DESCRIPTION     (TXT): {
Cluster is a set of 4 spacecraft, launched in summer 2000.}
```

• EXPERIMENT_DESCRIPTION

A short text to summarize the experiment which provides the data. Example :

```
PAR EXPERIMENT_DESCRIPTION  (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF)
measures the 3 components of the magnetic field up to 4kHz. The STAFF-SC
waveform unit (search coil) produces waveform up to either 10 or 180 Hz,
according to the bit rate. In Normal Bit Rate (NBR), the sample
frequency is 25.0 Hz, while it is 450 Hz in High Bit Rate (HBR). A
quality factor has been added for each xyz component of waveform data.
The data are given in the "SSW6RF" coordinates which means " STAFF
Sensors WEC 6 Reference Frame " where z is close to the spin axis.}
```

• INDEX_DESCRIPTION

A text which can precise anything more on the index of the data which can be not well described by the metadata parameters. Example :

```
PAR INDEX_DESCRIPTION       (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z"; "T" and "Z" separators are required.}
```

• INDEX_EXTENSION_DESCRIPTION

A text which precise the extended index of the data which can be not well described by the metadata parameters. Example :

```
PAR INDEX_EXTENSION_DESCRIP (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contains only a limited series of scalar
data; label, type and units can be different, but in accordance with the
given format. Here, extended index contains the status , which is a word
of 10 caracteres, and the spin phase in double precision.}
```

• DATA_DESCRIPTION

A text which can precise anything more on the data which can be not well described by the metadata parameters.

```
PAR DATA_DESCRIPTION        (TXT): {
This file contains "Matrix" DATA_FORM which are indexed by the time.
So, the file class is WaveForm also called MatTime. This is a temporal
series of data blocks. Each block begins with the INDEX value,
(ISO epoch) and INDEX_EXTENSION values, the rest of the block is a
series of DATA values, over several lines, according to the DATA_FORMAT
value. The number of values per line, or group of lines, is the first
dimension of the matrix, the number of lines, or group of lines, is the
second dimension of the matrix.}
```

• BLOCK_DESCRIPTION

A text which precise the data block composition.

```
PAR BLOCK_DESCRIPTION       (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed by the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}
```

- HISTORY

A text which precises anything that happened to the original file, and the Date-Time epoch when the operation was done. For instance, different operations done on a given file by a Roproc command. Example is given below:

```
PAR HISTORY                 (TXT): {
2010-08-27T04:13:06.000Z : N1_TO_RFF V.20090819
2011-01-21T11:41:58.000Z : GETDATA_CLUSTER_STAFF_SC V.20100409}
```

3.5.2 Other optional parameters

This field is free and can be used for anything required. For instance, at section 4.2.2, we define in the constant data field variables as `VAR TIME_SPAN_FROM and VAR TIME_SPAN_FROM`. It should be possible (and may be better) to place this information as optional parameters, rather than as constant data. Thus, the build-in of a new RFF file requieres a moment of reflection.

Chapter 4

DATA STRUCTURE

4.1 DATA GROUPS

There are only two allowed groups of data: the constant data, and the indexed data. Then, the structure of the full data group is as below:

```
START DATA

     START CONSTANT_DATA
     VAR ...
     END CONSTANT_DATA

     START INDEXED_DATA
     VAR ...
     END INDEXED_DATA

END DATA
```

4.2 CONSTANT DATA GROUP

4.2.1 Constant data inside the current file

The `CONSTANT_DATA` group contains a series of user's variables, for instance :

```
START CONSTANT_DATA

VAR SAMPLE_RATE               (FLT), u=Hz        :  25.000000
VAR VOLT_RANGE                (FLT), u=Volts     :  10.00
VAR TM_RANGE_MIN              (INT), u=TM_counts :  0
VAR TM_RANGE_MAX              (INT), u=TM_counts :  65535

END CONSTANT_DATA
```

Theses values are given only one time, in the corresponding `START/END` group, and then are constant into a given RFF file.

4.2.2 Constant data within a given period

It is possible to give a date of measurement for a group of constant data, for instance :

```
VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME  :  2001-09-23T00:00:00Z
VAR SPIN_PERIOD               (DBL), u=second    :  4.011473
```

```
VAR SPIN_GEI_RIGHT_ASCENSION  (FLT), u=degree    :    74.66
VAR SPIN_GEI_DECLINATION      (FLT), u=degree    :   -67.10
VAR MASS_CENTER_X             (FLT), u=mm        :    769.5
VAR MASS_CENTER_Y             (FLT), u=mm        :     -0.1
VAR MASS_CENTER_Z             (FLT), u=mm        :     -0.1
VAR EULER_ANGLE_FIRST         (FLT), u=degree    :     0.02
VAR EULER_ANGLE_SECOND        (FLT), u=degree    :     0.10
```

This time is generally included into the period covered by the file, but it is not mandatory. One also can have time span variable, indicating what period is covered by the file, having or not data during this period. For instance :

```
VAR TIME_SPAN_FROM            (STR), u=ISO_TIME  : 2009-12-14T12:50:00.000Z
VAR TIME_SPAN_TO              (STR), u=ISO_TIME  : 2009-12-14T14:00:00.000Z
```

4.3 INDEXED DATA GROUP

The INDEXED_DATA group contains the main data themselves. This is a series of indexed data blocks, according data type. For instance, a 3-D vector data type, indexed with a scalar ISO_TIME, will be as following:

```
START INDEXED_DATA

2003-08-17T16:20:00.006Z  -0.259448E+02  -0.863800E+01   0.380560E+01
2003-08-17T16:20:00.021Z  -0.259574E+02  -0.862550E+01   0.375610E+01
. . .

2003-08-17T16:50:05.975Z  -0.295214E+02   0.913700E+00   0.236880E+01
2003-08-17T16:50:05.990Z  -0.295088E+02   0.884800E+00   0.239170E+01

END INDEXED_DATA
```

Another example is a matrix (3,4) data type, with the same scalar ISO_TIME index. The corresponding indexed data block will be as :

```
START INDEXED_DATA

2001-02-18T19:16:00.514987Z
 -0.110823E+00 -0.125375E+00  0.421707E-01
 -0.104275E+00 -0.118793E+00  0.390765E-01
 -0.974686E-01 -0.112858E+00  0.314512E-01
 -0.881602E-01 -0.105190E+00  0.276471E-01

2001-02-18T19:16:01.514987Z
 -0.104275E+00 -0.118793E+00  0.390765E-01
  0.247627E+00  0.183281E+00  0.467864E+00
  0.240096E+00  0.188725E+00  0.511242E+00
  0.231496E+00  0.194711E+00  0.560935E+00

END INDEXED_DATA
```

A field line model, such as the magnetic field vector, indexed by the location in space of the various vector , will has a vector(3) data type and a vector(3) index type, and the indexed data group will be as :

```
START INDEXED_DATA

 0.123  5.257   8.481   -0.110823E+00 -0.125375E+00  0.421707E-01
 0.564  5.347   9.267   -0.104275E+00 -0.118793E+00  0.390765E-01
 0.625  5.780  10.593   -0.974686E-01 -0.112858E+00  0.314512E-01
 0.735  5.912  11.459   -0.881602E-01 -0.105190E+00  0.276471E-01

END INDEXED_DATA
```

According of course to the appropriate `INDEXED_DATA_FORMAT` parameter. Examples of full RFF files are given in section chapter 5.

Chapter 5

EXAMPLES OF RFF FILES

5.1 VECTOR SERIES INDEXED BY TIME

5.1.1 Time indexed vector data : VecTime Class

For instance, a CLUSTER FGM data file contains the B magnetic field vector versus date and time of the measurement. It is an example of vector time series. Vectors are not necessarly regularly spaced in time. The readability of this kind of Roproc_Format_File is very easy, but the size can be heavy for a high number of vectors. Significant parameters are following:

```
PAR FILE_CLASS          (STR): VecTime

PAR DATA_FORM           (STR): Vector
PAR DATA_DIMENSION      (INT): 3
PAR DATA_TYPE           (STR): FLT
PAR DATA_UNITS          (STR): nT
```

As the vector data varies with time, the date-time epoch must be without any ambiguity, so it is encoded in ISO format, as "2004-03-26T11:53:46.000Z" , so one have:

```
PAR INDEX_FORM           (STR): Scalar
PAR INDEX_DIMENSION      (INT): 1
PAR INDEX_TYPE           (STR): STR
PAR INDEX_UNITS          (STR): ISO_TIME
```

Note that in spite of the INDEX_FORM being 'Scalar', the INDEX_DIMENSION keyword is mandatory and must be equal to 1.

The structure of the data is simple, since each indexed data block is a record, or line, corresponds to a time indexed vector measurement, so one have:

```
PAR INDEXED_DATA_FORMAT   (STR): (a24,3e15.6)
```

An example of data block can be :

```
2003-08-17T16:20:00.006Z  -0.259448E+02  -0.863800E+01   0.380560E+01
```

Example given in table 5.1 shows the corresponding CLUSTER/FGM Roproc_Format_File

Table 5.1: *RFF file for pseudo irregular time indexed data*

```
START ROPROC_FORMAT_FILE
#=========================================================================
# The Roproc_Format_File is a general format used mainly for data from
# spatial missions (magnetometer, waveform units, S/C trajectory ...).
# For readability, all lines beginning with character # are comments,
# empty or blank lines are ignored. Others lines are Keywords,
# parameters or data.
# This file contents MetaData (description of the data), Constant Data
# (one value per file) and indexed data (data versus time or other
# INDEX).
# Any group of data begins with a "START" keyword, and stop by a "END".
# MetaData are given as parameters series (one value per parameter).
# Data are described by Metadata parameters.
# Examples of different files are given in the document:
# "Roproc_Format_File Description" (Ref: TBD).
#
# Author: P. Robert, CNRS/CETP
# V 1.0,  1996-2000   (for archive of old mission data)
# V 1.1, October 2001 (for CLUSTER/STAFF-SC NBR & HBR wave data)
# V 1.2, January 2002 (for GEOS wave data)
# V 1.3, August  2003 (for CUSP and any kind of wave data)
# V 1.4, January 2003 (for general titles management)
# V 2.0, March   2004 (for compatibility with Roproc Vector format)
# V 2.1, May     2004 (to be coherent with Cluster Exchange Format)
# Any comment or suggestion: Patrick.Robert@cetp.ipsl.fr
#=========================================================================

START METADATA
#-------------------------------------------------------------------
# two metadata categories: Mandatory Parameters and Optional Parameters
#-------------------------------------------------------------------

START MANDATORY_PARAMETERS

PAR FILE_NAME              (STR): clufgm_bav.rff
PAR FILE_CLASS             (STR): VecTime
PAR FILE_FORMAT_VERSION    (STR): Roproc_Format_File V 2.1
PAR FILE_CREATION_DATE     (STR): 2004-03-26T11:53:46.000Z

PAR MISSION_NAME           (STR): CLUSTER
PAR OBSERVATORY_NAME       (STR): CLUSTER-4
PAR OBSERVATORY_SURNAME    (STR): Tango
PAR EXPERIMENT_NAME        (STR): FGM
PAR EXPERIMENT_MODE        (STR): BAV
PAR INSTRUMENT_TYPE        (STR): Magnetometer
PAR MEASUREMENT_TYPE       (STR): DC Magnetic field

PAR DATA_LABEL             (STR): B
PAR DATA_FORM              (STR): Vector
PAR DATA_DIMENSION         (INT): 3
PAR DATA_TYPE              (STR): FLT
PAR DATA_REPRESENTATION    (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): GSE
PAR DATA_UNITS             (STR): nT
PAR DATA_FILL_VALUE        (FLT): -1.E-20

PAR INDEX_LABEL            (STR): Time
PAR INDEX_FORM             (STR): Scalar
PAR INDEX_TYPE             (STR): String
PAR INDEX_UNITS            (STR): ISO_TIME Date/Time epoch
PAR INDEX_PROPERTIES       (STR): Roughly Regularly Spaced

PAR INDEXED_DATA_FORMAT    (STR): (a24,3e15.6)
PAR INDEXED_DATA_NUMBER    (INT): 59

END MANDATORY_PARAMETERS
```

```
START OPTIONAL_PARAMETERS

PAR SUB_TITLE           (STR): PPD 4 sec. Res.
PAR DISCIPLINE_NAME     (STR): Space physics, Magnetospheric Physics
PAR EXPERIMENT_PI_NAME  (STR): A.Balogh@ic.ac.uk

PAR MISSION_DESCRIPTION    (TXT):
{Cluster is a set of 4 Spacecraft, launched in summer 2000.}

PAR EXPERIMENT_DESCRIPTION (TXT):
{The Flux Gate Magnetometer experiment (FGM) measure the 3 components
of the DC magnetic field vector. The time resolution is varying:
the low time resolution, also called Prime Parameter Data (PPD) is of
the order of the spin period (4 s), while the high resolution is
around 0.04 s.}

PAR DATA_DESCRIPTION  (TXT):
{This file contains "vector" DATA_FORM which are indexed by the time.
So this is a temporal series of data blocks. Each block begin with the
INDEX value, (ISO epoch) the rest of the block is a series of values,
corresponding to the number of component of the data vector. A data
block can be written on a single line, or record, or a group of lines,
according the INDEXED_DATA_FORMAT descriptor.}

PAR INDEX_DESCRIPTION (TXT):
{Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z"; "T" and "Z" separators are required.}

END OPTIONAL_PARAMETERS
END METADATA

#=============================================================================

START DATA
#-----------------------------------------------------------------------------
# two categories: Constant data (one value per file), and Indexed data
# (data versus time or other INDEX)
#-----------------------------------------------------------------------------

START CONSTANT_DATA

VAR SPIN_FREQUENCY     (FLT), u=Hz     :  0.249516E+00
VAR GEI_SPIN_RIGHT_ASC (FLT), u=degree :  0.102590E+03
VAR GEI_SPIN_DECLIN    (FLT), u=degree : -0.638600E+02

END CONSTANT_DATA
#-----------------------------------------------------------------------------

START INDEXED_DATA

2003-08-17T16:20:00.006Z  -0.259448E+02  -0.863800E+01   0.380560E+01
2003-08-17T16:20:00.021Z  -0.259574E+02  -0.862550E+01   0.375610E+01
2003-08-17T16:20:00.036Z  -0.259537E+02  -0.857630E+01   0.373060E+01
2003-08-17T16:20:00.051Z  -0.259100E+02  -0.853730E+01   0.369280E+01
2003-08-17T16:20:00.066Z  -0.259987E+02  -0.858120E+01   0.381320E+01
2003-08-17T16:20:00.081Z  -0.259280E+02  -0.860000E+01   0.373870E+01
2003-08-17T16:20:00.096Z  -0.259411E+02  -0.859420E+01   0.373850E+01
2003-08-17T16:20:00.110Z  -0.259403E+02  -0.860050E+01   0.372860E+01
2003-08-17T16:20:00.125Z  -0.259472E+02  -0.861820E+01   0.373570E+01

. . .   59 time indexed vector data records

2003-08-17T16:50:05.975Z  -0.295214E+02   0.913700E+00   0.236880E+01
2003-08-17T16:50:05.990Z  -0.295088E+02   0.884800E+00   0.239170E+01
2003-08-17T16:50:06.005Z  -0.294824E+02   0.865500E+00   0.240520E+01
END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE
```

5.1.2 Regular time indexed matrix data : WaveForm Class

A CLUSTER STAFF-SC data file contains the B magnetic field vector, in 0.1- 12.5 Hz frequency range, versus date and time.

The only difference between the previous example is the data form: B vector is given by blocks of vectors, and only the beginning of the block is indexed by time, not each vector. The time of a single vector in the block is defined by the time at the beginning of the block, and the frequency sample, which must be known and constant. This is why the frequency sample is given in the CONSTANT_DATA block.

This simply means that the DATA_FORM parameter value is 'Matrix', the INDEX_FORM remaining 'Scalar', and contains the Date/time epoch in ISO_TIME format. The WaveForm class is in fact a more general MatTime class.

```
PAR DATA_FORM          (STR): Matrix
```

In the example below, the block data of the 3D vector have 25 lines, then one have:

```
PAR DATA_DIMENSION     (INT): 4  25
```

The data being time depending, index has the following properties:

```
PAR INDEX_TYPE         (STR): STR
PAR INDEX_UNITS        (STR): ISO_TIME
PAR INDEX_FORMAT       (STR): (a27)
PAR INDEX_FORM         (STR): Scalar
PAR INDEX_DIMENSION    (INT): 1
PAR INDEX_PROPERTIES   (STR): Regularly Spaced
```

Note that if INDEX_PROPERTIES is regularly spaced, the frequency sample is constant, and this frequency should be know by an user's variable, as:

```
VAR FREQUENCY_SAMPLE   (FLT), u=Hz :  0.250000E+00
```

Furthermore, one wish that each data block contains experiment status and phase angle. This can be done by using the notion of index extension (see 3.4.3, as :

```
PAR INDEX_EXTENSION_LABEL     (STR): Status ; Phase_angle
PAR INDEX_EXTENSION_TYPE      (STR): STR ; DBL
PAR INDEX_EXTENSION_UNITS     (STR): None ; degree
PAR INDEX_EXTENSION_FORMAT    (STR): (a11,1x,f7.2)
PAR INDEX_EXTENSION_LENGTH    (INT): 19
```

This advantage of this format is that the ISO_TIME being given every 25 vectors, the encoding data as "matrix form" save space disk by comparison with the previous format, where each vector has an encoded ISO_TIME.

Example of a such file is given hereafter in table 5.2.

Table 5.2: *RFF file for regular time indexed vector data*

```
START ROPROC_FORMAT_FILE

#=====================================================================
     same comments as example 1-a
#=====================================================================

START METADATA


START MANDATORY_PARAMETERS

PAR FILE_NAME                 (STR): working_file.rff
PAR FILE_CLASS                (STR): WaveForm
PAR FILE_FORMAT_VERSION       (STR): Roproc_Format_File V2.2
PAR FILE_CREATION_DATE        (STR): 2011-01-21T11:41:58.000Z

PAR MISSION_NAME              (STR): CLUSTER
PAR OBSERVATORY_NAME          (STR): Salsa
PAR OBSERVATORY_NUMBER        (INT): 2
PAR EXPERIMENT_NAME           (STR): STAFF-SC
PAR EXPERIMENT_MODE           (STR): NBR
PAR INSTRUMENT_TYPE           (STR): Search Coils
PAR MEASUREMENT_TYPE          (STR): AC Magnetic field waveform

PAR INDEX_LABEL               (STR): Time
PAR INDEX_TYPE                (STR): STR
PAR INDEX_UNITS               (STR): ISO_TIME
PAR INDEX_FORMAT              (STR): (a27)
PAR INDEX_FORM                (STR): Scalar
PAR INDEX_DIMENSION           (INT): 1
PAR INDEX_PROPERTIES          (STR): Regularly Spaced

PAR INDEX_EXTENSION_LABEL     (STR): Status ; Phase_angle
PAR INDEX_EXTENSION_TYPE      (STR): STR ; DBL
PAR INDEX_EXTENSION_UNITS     (STR): None ; degree
PAR INDEX_EXTENSION_FORMAT    (STR): (a11,1x,f7.2)
PAR INDEX_EXTENSION_LENGTH    (INT): 19

PAR DATA_LABEL                (STR): Bx ; By ; Bz ; Compression Factor
PAR DATA_TYPE                 (STR): INT
PAR DATA_UNITS                (STR): TM_counts ; TM_counts ; TM_counts ; None
PAR DATA_FORMAT               (STR): (24((3(z4,1x),i1),/),(3(z4,1x),i1))
PAR DATA_FORM                 (STR): Matrix
PAR DATA_DIMENSION            (INT): 4 25
PAR DATA_REPRESENTATION       (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM    (STR): SSW6RF
PAR DATA_FILL_VALUE           (STR): -999

PAR BLOCK_NUMBER              (INT): 4138
PAR BLOCK_FIRST_INDEX         (STR): 2009-12-14T12:49:59.698702Z
PAR BLOCK_LAST_INDEX          (STR): 2009-12-14T13:59:59.636289Z

END MANDATORY_PARAMETERS


START OPTIONAL_PARAMETERS

PAR TITLE                     (STR): CLUSTER / STAFF-SC / Salsa (#2)
PAR SUB_TITLE                 (STR): TM data in spinning system
PAR DISCIPLINE_NAME           (STR): Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME        (STR): Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL        (STR): Nicole.Cornilleau@lpp.polytechnique.fr

PAR MISSION_DESCRIPTION       (TXT): {
Cluster is a set of 4 spacecraft, launched in summer 2000.}

PAR EXPERIMENT_DESCRIPTION    (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF)
measures the 3 components of the magnetic field up to 4kHz. The STAFF-SC
waveform unit (search coil) produces waveform up to either 10 or 180 Hz,
according to the bit rate. In Normal Bit Rate (NBR), the sample
frequency is 25.0 Hz, while it is 450 Hz in High Bit Rate (HBR). A
quality factor has been added for each xyz component of waveform data.
The data are given in the "SSW6RF" coordinates which means " STAFF
Sensors WEC 6 Reference Frame " where z is close to the spin axis.}

PAR INDEX_DESCRIPTION         (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z"; "T" and "Z" separators are required.}
```

```
PAR INDEX_EXTENSION_DESCRIP   (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data; label, type and units can be different, but in accordance with the
given format. Here, extended index contains the status , which is a word
of 10 characters, and the spin phase in double precision.}

PAR DATA_DESCRIPTION          (TXT): {
This file contains "Matrix" DATA_FORM which are indexed by the time.
So, the file class is WaveForm also called MatTime. This is a temporal
series of data blocks. Each block begins with the INDEX value,
(ISO epoch) and INDEX_EXTENSION values, the rest of the block is a
series of DATA values, over several lines, according to the DATA_FORMAT
value. The number of values per line, or group of lines, is the first
dimension of the matrix, the number of lines, or group of lines, is the
second dimension of the matrix.}

PAR BLOCK_DESCRIPTION         (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

PAR HISTORY                   (TXT): {
2010-08-27T04:13:06.000Z : N1_TO_RFF V.20090819
2011-01-21T11:41:58.000Z : GETDATA_CLUSTER_STAFF_SC V.20100409}

END OPTIONAL_PARAMETERS
#------------------------------------------------------------------------------
END METADATA
#------------------------------------------------------------------------------

START DATA

#------------------------------------------------------------------------------
START CONSTANT_DATA

VAR SAMPLE_RATE               (FLT), u=Hz        :  25.0003611
VAR VOLT_RANGE                (FLT), u=Volts     : 10.00
VAR TM_RANGE_MIN              (INT), u=TM_counts : 0
VAR TM_RANGE_MAX              (INT), u=TM_counts : 65535

VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None      : 1.000000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None      : 1.000000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None      : 1.000000

VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME  : 2009-12-12T07:36:40Z
VAR SPIN_PERIOD               (DBL), u=second    : 4.138893
VAR SPIN_GEI_RIGHT_ASCENSION  (FLT), u=degree    : 90.23
VAR SPIN_GEI_DECLINATION      (FLT), u=degree    : -61.66
VAR MASS_CENTER_X             (FLT), u=mm        : 762.0
VAR MASS_CENTER_Y             (FLT), u=mm        : -0.0
VAR MASS_CENTER_Z             (FLT), u=mm        : 0.0
VAR EULER_ANGLE_FIRST         (FLT), u=degree    : -0.01
VAR EULER_ANGLE_SECOND        (FLT), u=degree    : 0.04

VAR TIME_SPAN_FROM            (STR), u=ISO_TIME  : 2009-12-14T12:50:00.000Z
VAR TIME_SPAN_TO              (STR), u=ISO_TIME  : 2009-12-14T14:00:00.000Z

END CONSTANT_DATA
```

```
START INDEXED_DATA

2009-12-14T12:49:59.698702Z 00100100010   66.80
7C4A 7A39 7FC1 0
7BF2 7A79 7FC4 0
7BA0 7AB4 7FC0 0
7B51 7AFB 7FC0 0
7AFE 7B47 7FBF 0
7AB5 7B8E 7FC6 0
7A79 7BE1 7FBD 0
7A3D 7C46 7FBE 0
79FC 7CA0 7FBE 0
79C7 7CF4 7FBF 0
79A0 7D51 7FC5 0
7977 7DB4 7FBB 0
7956 7E1D 7FBF 0
793A 7E7F 7FC3 0
7921 7EEE 7FC7 0
7919 7F57 7FC3 0
7914 7FBD 7FC5 0
7911 802E 7FC5 0
791D 809E 7FC3 0
7925 8100 7FC5 0
7939 8168 7FC8 0
7951 81D5 7FCA 0
7971 823B 7FC8 0
79A0 82A6 7FC9 0
79C8 8309 7FC6 0
2009-12-14T12:50:00.698686Z 00100100010  153.77
79FD 8366 7FC7 0
7A33 83B7 7FCC 0
7A6D 841C 7FCB 0
7AAF 8473 7FCE 0
7AFB 84BF 7FCF 0
7B45 850C 7FCE 0
7B9A 8551 7FD4 0


...


81E4 7F14 7FDD 0
81D6 7EF5 7FDF 0
81C1 7ED9 7FDE 0
81AF 7EBF 7FE3 0
819D 7EA2 7FDD 0
8187 7E8E 7FDA 0
816B 7E7A 7FDD 0
814D 7E5F 7FDB 0

END INDEXED_DATA
#-------------------------------------------------------------------------------
END DATA
#-------------------------------------------------------------------------------
END ROPROC_FORMAT_FILE
```

5.1.3 Spectra as time indexed vector data : Spectrogram Class

One often needs files containing spectrogram data, also called dynamic spectra: this is a series of spectrum values, indexed by time, of an initial waveform vector data. Since we have 3 components, the corresponding data block will be a tri-vector, of for instance 128 complex values over 128 frequencies, if the Fourier transform has been done on 256 time points. Each frequency corresponds to a line of the spectrum, and can be regularly spaced of δF frequency width.

For facilities, rather define a complex vector of dimension 3, we have choosed to define a real vector of dimension 6, 3 for the real part, and 3 for the imaginary part.
So, one will have:

```
PAR DATA_LABEL              (STR): Bxr ; Bxi ; Byr ; Byi ; Bzr ; Bzi
PAR DATA_TYPE               (STR): FLT
PAR DATA_UNITS              (STR): nT ; nT ; nT ; nT ; nT ; nT
PAR DATA_FORMAT             (STR): (127(6(1x,E12.5),/),6(1x,E12.5))
PAR DATA_FORM               (STR): Matrix
PAR DATA_DIMENSION          (INT): 6 128
PAR DATA_REPRESENTATION     (STR): xyz Cartesian

PAR INDEX_LABEL             (STR): Time
PAR INDEX_TYPE              (STR): STR
PAR INDEX_UNITS             (STR): ISO_TIME
PAR INDEX_FORMAT            (STR): (a27)
PAR INDEX_FORM              (STR): Scalar
PAR INDEX_DIMENSION         (INT): 1
PAR INDEX_PROPERTIES        (STR): Regularly Spaced
```

If the spectrogram contains 4263 individual spectra, one will have:

```
PAR BLOCK_NUMBER            (INT): 4263
```

Note that the spectrogram could be not necessarily regularly spaced in time ; after unprocessed data gaps, time could restart at any time, not inevitably at an multiply integer of 256 points in the initial waveform.

In all cases, one should have user's variables, giving the δf frequency interval x (equal to $1/\delta T$, the time duration of the corresponding waveform), and the f_{min} and f_{max} frequency range, and other usefull informations about the spectra itself and the waveform having used to compute it:

```
VAR WAVEFORM_SAMPLE_RATE    (FLT), u=Hz       : 25.0002
VAR WINDOW_LENGTH           (INT), u=None     : 256
VAR TIME_RESOLUTION         (FLT), u=s        : 10.2399
VAR FREQUENCY_RESOLUTION    (FLT), u=Hz       : 0.0976569
VAR FREQUENCY_MIN           (FLT), u=Hz       : 0.
VAR FREQUENCY_MAX           (FLT), u=Hz       : 12.4024
```

One cal also notice the field "history" which give information on how the spectrogram has been computed from the initial level 1 TM waveform:

```
PAR HISTORY                 (TXT): {
2010-11-13T01:55:18.000Z : N1_TO_RFF V.20090819
2010-11-18T17:55:23.000Z : GETDATA_CLUSTER_STAFF_SC V.20091022
2010-11-18T17:57:27.000Z : RESIZE_BLOCK V.20090609
2010-11-18T17:57:27.000Z : waveform_calibration V.20090702
2010-11-18T17:57:46.000Z : change_coordinate_system V.20090604
2010-11-18T17:57:51.000Z : waveform_to_spectrogram V.20090630}
```

Example is given hereafter, in table 5.3

Table 5.3: *RFF file for spectrogram viewed as time indexed vector data*

```
START ROPROC_FORMAT_FILE

START METADATA

START MANDATORY_PARAMETERS

PAR FILE_NAME                 (STR): CLU3_STASC_NBR_CSL2_20091107.rff
PAR FILE_CLASS                (STR): Spectrogram
PAR FILE_FORMAT_VERSION       (STR): Roproc_Format_File V2.2
PAR FILE_CREATION_DATE        (STR): 2010-11-18T17:57:51.000Z

PAR MISSION_NAME              (STR): CLUSTER
PAR OBSERVATORY_NAME          (STR): Samba
PAR OBSERVATORY_NUMBER        (INT): 3
PAR EXPERIMENT_NAME           (STR): STAFF-SC
PAR EXPERIMENT_MODE           (STR): NBR
PAR INSTRUMENT_TYPE           (STR): Search Coils
PAR MEASUREMENT_TYPE          (STR): AC Magnetic field waveform

PAR INDEX_LABEL               (STR): Time
PAR INDEX_TYPE                (STR): STR
PAR INDEX_UNITS               (STR): ISO_TIME
PAR INDEX_FORMAT              (STR): (a27)
PAR INDEX_FORM                (STR): Scalar
PAR INDEX_DIMENSION           (INT): 1
PAR INDEX_PROPERTIES          (STR): Regularly Spaced

PAR INDEX_EXTENSION_LABEL     (STR): Quality factor
PAR INDEX_EXTENSION_TYPE      (STR): INT
PAR INDEX_EXTENSION_UNITS     (STR): None
PAR INDEX_EXTENSION_FORMAT    (STR): (i2)
PAR INDEX_EXTENSION_LENGTH    (INT): 2

PAR DATA_LABEL                (STR): Bx ; By ; Bz
PAR DATA_TYPE                 (STR): FLT
PAR DATA_UNITS                (STR): nT ; nT ; nT
PAR DATA_FORMAT               (STR): (127(6(1x,E12.5),/),6(1x,E12.5))
PAR DATA_FORM                 (STR): Matrix
PAR DATA_DIMENSION            (INT): 6 128
PAR DATA_REPRESENTATION       (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM    (STR): GSE
PAR DATA_FILL_VALUE           (STR): -0.1000E+31

PAR BLOCK_NUMBER              (INT): 4263
PAR BLOCK_FIRST_INDEX         (STR): 2009-11-07T00:00:00.030591Z
PAR BLOCK_LAST_INDEX          (STR): 2009-11-07T23:59:44.069091Z

END MANDATORY_PARAMETERS
```

```
START OPTIONAL_PARAMETERS

PAR TITLE                     (STR): CLUSTER / STAFF-SC / Samba (#3)
PAR SUB_TITLE                 (STR): Step 4: Data in GSE system [nT] without DC
PAR DISCIPLINE_NAME           (STR): Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME        (STR): Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL        (STR): Nicole.Cornilleau@lpp.polytechnique.fr

PAR MISSION_DESCRIPTION       (TXT): {
Cluster is a set of 4 spacecraft, launched in summer 2000.}

PAR EXPERIMENT_DESCRIPTION    (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF)
measures the 3 components of the magnetic field up to 4kHz. The STAFF-SC
waveform unit (search coil) produces waveform up to either 10 or 180 Hz,
according to the bit rate. In Normal Bit Rate (NBR), the sample
frequency is 25.0 Hz, while it is 450 Hz in High Bit Rate (HBR). A
quality factor has been added for each xyz component of waveform data.
The data are given in the "SSW6RF" coordinates which means " STAFF
Sensors WEC 6 Reference Frame " where z is close to the spin axis.}

PAR INDEX_DESCRIPTION         (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z"; "T" and "Z" separators are required.}

PAR INDEX_EXTENSION_DESCRIP   (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data; label, type and units can be different, but in accordance with the
given format. Here, extended index contains the status , which is a word
of 10 characters, and the spin phase in double precision.}

PAR DATA_DESCRIPTION          (TXT): {
This file contains "Matrix" DATA_FORM which are indexed by the time.
So, the file class is WaveForm also called MatTime. This is a temporal
series of data blocks. Each block begins with the INDEX value,
(ISO epoch) and INDEX_EXTENSION values, the rest of the block is a
series of DATA values, over several lines, according to the DATA_FORMAT
value. The number of values per line, or group of lines, is the first
dimension of the matrix, the number of lines, or group of lines, is the
second dimension of the matrix.}

PAR BLOCK_DESCRIPTION         (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

PAR HISTORY                   (TXT): {
2010-11-13T01:55:18.000Z : N1_TO_RFF V.20090819
2010-11-18T17:55:23.000Z : GETDATA_CLUSTER_STAFF_SC V.20091022
2010-11-18T17:57:27.000Z : RESIZE_BLOCK V.20090609
2010-11-18T17:57:27.000Z : waveform_calibration V.20090702
2010-11-18T17:57:46.000Z : change_coordinate_system V.20090604
2010-11-18T17:57:51.000Z : waveform_to_spectrogram V.20090630}

END OPTIONAL_PARAMETERS

END METADATA
```

```
START DATA

START CONSTANT_DATA

VAR VOLT_RANGE                (FLT), u=Volts     : 10.00
VAR TM_RANGE_MIN              (INT), u=TM_counts : 0
VAR TM_RANGE_MAX              (INT), u=TM_counts : 65535

VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None      : 1.000000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None      : 1.000000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None      : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None      : 1.000000

VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME  : 2009-11-05T19:54:09Z
VAR SPIN_PERIOD               (DBL), u=second    : 4.160858
VAR SPIN_GEI_RIGHT_ASCENSION  (FLT), u=degree    : 74.78
VAR SPIN_GEI_DECLINATION      (FLT), u=degree    : -63.64
VAR MASS_CENTER_X             (FLT), u=mm        : 751.0
VAR MASS_CENTER_Y             (FLT), u=mm        : 0.1
VAR MASS_CENTER_Z             (FLT), u=mm        : -0.1
VAR EULER_ANGLE_FIRST         (FLT), u=degree    : -0.09
VAR EULER_ANGLE_SECOND        (FLT), u=degree    : 0.00

VAR TIME_SPAN_FROM            (STR), u=ISO_TIME  : 2009-11-07T00:00:00.000Z
VAR TIME_SPAN_TO              (STR), u=ISO_TIME  : 2009-11-07T23:59:59.999Z

VAR FREQUENCY_FILTER_MIN      (STR), u=Hz        : 0.
VAR FREQUENCY_FILTER_MAX      (STR), u=Hz        : 12.5001
VAR FREQUENCY_CUT_OFF         (STR), u=Hz        : 0.1
VAR FREQUENCY_DETREND         (STR), u=Hz        : 0.5
VAR WAVEFORM_SAMPLE_RATE      (FLT), u=Hz        : 25.0002
VAR WINDOW_LENGTH             (INT), u=None      : 256
VAR TIME_RESOLUTION           (FLT), u=s         : 10.2399
VAR FREQUENCY_RESOLUTION      (FLT), u=Hz        : 0.0976569
VAR FREQUENCY_MIN             (FLT), u=Hz        : 0.
VAR FREQUENCY_MAX             (FLT), u=Hz        : 12.4024

END CONSTANT_DATA

#-------------------------------------------------------------------------------

START INDEXED_DATA

2009-11-07T00:00:00.030591Z  0
  0.62304E-03  0.00000E+00 -0.18052E-02  0.00000E+00  0.83686E-03  0.00000E+00
  0.59885E-02  0.14932E-01  0.77866E-02 -0.21595E-01  0.77286E-03 -0.28893E-02
 -0.10645E-01 -0.81618E-03 -0.43766E-03  0.62741E-02 -0.96669E-02 -0.36955E-02
  0.89155E-02  0.50728E-02  0.90789E-02 -0.90548E-02 -0.74348E-02 -0.55377E-02
  0.59572E-02 -0.89664E-02  0.20762E-02  0.20906E-02  0.56126E-02  0.61827E-02
...
 -0.88745E-04 -0.15233E-03  0.57538E-03 -0.27242E-03 -0.22584E-03 -0.92462E-04
  0.49505E-04 -0.78982E-04 -0.10755E-03 -0.69034E-04  0.28144E-03 -0.22003E-03
 -0.49728E-03  0.15435E-03  0.20047E-03  0.16694E-03  0.12081E-03 -0.13494E-03
2009-11-07T00:00:10.270520Z  0
 -0.30762E-02  0.00000E+00 -0.13022E-03  0.00000E+00 -0.67083E-03  0.00000E+00
  0.22007E-02 -0.11023E-01  0.61526E-02 -0.96347E-02 -0.88746E-03  0.63511E-03
  0.77923E-02 -0.59471E-02 -0.12025E-02 -0.90411E-02  0.48807E-02  0.60068E-02
  0.43176E-02 -0.54447E-02 -0.32732E-02 -0.11135E-02 -0.11029E-02  0.51605E-02
...
  0.15130E-03  0.31279E-03  0.16351E-03 -0.43407E-04 -0.64835E-04  0.25981E-03
  0.12020E-04  0.18591E-03  0.31496E-04  0.17357E-03  0.40379E-04  0.21040E-03
  0.67417E-04 -0.29981E-04 -0.13188E-03 -0.20120E-03 -0.35278E-03  0.24847E-03

END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE
```

5.2 MATRIX SERIES INDEXED BY SCALAR

5.2.1 Spectrograms as component indexed image or MatSca class

A spectrogram data can be considered as a image data, if all data gaps have been correctly managed. In this case, a x-y pixel numbered as (Nx, Ny) image coordinate has a unique correspondence in time and frequency, as:

$$\begin{aligned} t(nx) &= t1 + nx.dt \\ f(ny) &= f1 + ny.df \end{aligned} \tag{5.1}$$

To take again the preceding example, if one have to store 3 Bx, By, Bz spectrograms, each containing 500 spectra of 256 complex values, one will have:

```
PAR DATA_FORM           (STR): Image
PAR DATA_DIMENSION      (INT): 500 256
PAR DATA_TYPE           (STR): CMP

PAR INDEX_LABEL         (STR): Component
PAR INDEX_TYPE          (STR): STR
PAR INDEX_UNITS         (STR): None
PAR INDEX_FORMAT        (STR): (a14)
PAR INDEX_FORM          (STR): Scalar
PAR INDEX_DIMENSION     (INT): 1
PAR INDEX_PROPERTIES    (STR): None
```

As before, to avoid too much long lines, one use a format which splits the long 256 components vector into block of 64 lines, each line corresponding to 4 complex components, that is to say 8 real values. So one have the following parameters :

```
PAR DATA_FORMAT         (STR): (500(64(/,8e10.3),/,))
PAR BLOCK_NUMBER        (INT): 3
```

The "/" character at the end of the data block format allows the reader to visualize the end of each line of the image matrix, before reading the following line. It is not required, but helpful. This RFF file contains only 3 indexed data blocks, but each block is a (500,256) image. Of course, if one have an experiment which provides many channels, one can have more than 3 components or channels.

For spectrogram RFF files, it is necessary to know the $(\delta f, \delta t)$ frequency and time resolution of the image, and the f1 and t1 starting frequency and time. This can be done with the user's variables, as following:

```
VAR TIME_MIN             (STR),          : 2001-02-18T19:10:00.000Z
VAR TIME_MAX             (STR),          : 2001-02-18T20:40:00.000Z
VAR FREQUENCY_MIN        (FLT), u=Hz     :  0.200000E+00
VAR FREQUENCY_MAX        (FLT), u=Hz     :  0.125000E+02
VAR TIME_RESOLUTION      (FLT), u=s      :  0.102400E+02
VAR FREQUENCY_RESOLUTION (FLT), u=Hz     :  0.976562E-01
```

Example is given hereafter, in table 5.4

Table 5.4: *RFF file for spectrograms viewed as component indexed image*

```
START ROPROC_FORMAT_FILE

START METADATA

#--------------------------------------------------------------------

START MANDATORY_PARAMETERS

PAR FILE_NAME              (STR): clustaff_nbr.rff
PAR FILE_CLASS             (STR): Image
PAR FILE_FORMAT_VERSION    (STR): Roproc_Format_File V 2.1
PAR FILE_CREATION_DATE     (STR): 2004-03-26T11:53:46.000Z

PAR MISSION_NAME           (STR): CLUSTER
PAR OBSERVATORY_NAME       (STR): Tango
PAR OBSERVATORY_NUMBER     (STR): 4
PAR EXPERIMENT_NAME        (STR): STAFF-SC
PAR EXPERIMENT_MODE        (STR): NBR
PAR INSTRUMENT_TYPE        (STR): Magnetometer
PAR MEASUREMENT_TYPE       (STR): AC Magnetic field

# one define a series of images indexed by Bxyz various components

PAR INDEX_LABEL            (STR): Component
PAR INDEX_TYPE             (STR): STR
PAR INDEX_UNITS            (STR): None
PAR INDEX_FORMAT           (STR): (a14)
PAR INDEX_FORM             (STR): Scalar
PAR INDEX_DIMENSION        (INT): 1
PAR INDEX_PROPERTIES       (STR): None

PAR INDEX_EXTENSION_LABEL  (STR): None
PAR INDEX_EXTENSION_TYPE   (STR): None
PAR INDEX_EXTENSION_UNITS  (STR): None
PAR INDEX_EXTENSION_FORMAT (STR): None
PAR INDEX_EXTENSION_LENGTH (INT): 0

PAR DATA_LABEL             (STR): B
PAR DATA_TYPE              (STR): CMP
PAR DATA_UNITS             (STR): nT
PAR DATA_FORMAT            (STR): (500(64(/,8e11.3),/,))
PAR DATA_FORM              (STR): Image
PAR DATA_DIMENSION         (INT): 500 256
PAR DATA_REPRESENTATION    (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): GSE
PAR DATA_FILL_VALUE        (FLT): -1.E-20

PAR BLOCK_NUMBER           (INT): 3
PAR BLOCK_FIRST_INDEX      (STR): Bx Spectrogram
PAR BLOCK_LAST_INDEX       (STR): Bz Spectrogram

END MANDATORY_PARAMETERS

#--------------------------------------------------------------------

START OPTIONAL_PARAMETERS

PAR SUB_TITLE              (STR): Same as example 1-b
PAR DISCIPLINE_NAME        (STR): Same as example 1-b
PAR EXPERIMENT_PI_NAME     (STR): Same as example 1-b
PAR MISSION_DESCRIPTION    (TXT): Same as example 1-b
PAR EXPERIMENT_DESCRIPTION (TXT): Same as example 1-b

PAR DATA_DESCRIPTION  (TXT):
{This file contains "Image" DATA_FORM which are indexed by the label of
the selected channel. So this is a series of data blocks. Each block begin
with the INDEX value (a string label channel), the rest of the block is a
series of values, over several lines, according BLOCK_DATA_FORMAT value.
The number of values per line, or group of lines, correspond to the Y
pixels, the number of lines, or group of lines, correspond to the X
pixels.}
```

```
 PAR INDEX_DESCRIPTION (TXT):
 {Label of the selected channel is a character string, as "Bx". There is
 so many image that different channels, or index values, in file.}

 END OPTIONAL_PARAMETERS
 END METADATA

#-------------------------------------------------------------------

 START DATA

#-------------------------------------------------------------------

 START CONSTANT_DATA

 # image are spectrograms, so one define time range and frequency range,
 # as time and frequency resolution

 VAR TIME_MIN             (STR),          : 2001-02-18T19:10:00.000Z
 VAR TIME_MAX             (STR),          : 2001-02-18T20:40:00.000Z
 VAR FREQUENCY_MIN        (FLT), u=Hz     :  0.200000E+00
 VAR FREQUENCY_MAX        (FLT), u=Hz     :  0.125000E+02
 VAR TIME_RESOLUTION      (FLT), u=s      :  0.102400E+02
 VAR FREQUENCY_RESOLUTION (FLT), u=Hz     :  0.976562E-01

 # other variables for plots

 VAR SPIN_FREQUENCY       (FLT), u=Hz     :  0.249516E+00
 VAR GEI_SPIN_RIGHT_ASC   (FLT), u=degree :  0.102590E+03
 VAR GEI_SPIN_DECLIN      (FLT), u=degree : -0.638600E+02

 END CONSTANT_DATA

 #-------------------------------------------------------------------

 START INDEXED_DATA

 Bx Spectrogram
  -0.110E+00 -0.125E+00  0.421E-01 -0.104E+00 -0.118E+00  0.390E-01 -0.974E-01 -0.112E+00
   0.314E-01 -0.881E-01 -0.105E+00  0.276E-01 -0.788E-01 -0.983E-01  0.190E-01 -0.705E-01
  -0.902E-01  0.144E-01 -0.601E-01 -0.823E-01  0.520E-02 -0.104E+00 -0.118E+00  0.390E-01
    . . .       Data block of 64 lines of 4 complex values (i.e. 8 real values
 for real and imaginary parts),
       i.e. 256 complex vector components encoded as 512 real values for the first line
       of the image matrix
    . . .   500 data blocks  for the image matrix

 By Spectrogram
  -0.902E-01  0.144E-01 -0.601E-01 -0.823E-01  0.520E-02 -0.104E+00 -0.118E+00  0.390E-01
  -0.110E+00 -0.125E+00  0.421E-01 -0.104E+00 -0.118E+00  0.390E-01 -0.974E-01 -0.112E+00
   0.314E-01 -0.881E-01 -0.105E+00  0.276E-01 -0.788E-01 -0.983E-01  0.190E-01 -0.705E-01
    . . .   500 data blocks  for the image matrix

 Bz Spectrogram
  -0.788E-01 -0.983E-01  0.190E-01 -0.705E-01 -0.902E-01  0.144E-01 -0.601E-01 -0.823E-01
   0.520E-02 -0.104E+00 -0.118E+00  0.390E-01 -0.125E+00  0.421E-01 -0.104E+00 -0.118E+00
  -0.110E+00 -0.125E+00  0.421E-01 -0.104E+00 -0.601E-01 -0.823E-01  0.520E-02 -0.104E+00
    . . .   500 data blocks  for the image matrix

 END INDEXED_DATA
 END DATA
 END ROPROC_FORMAT_FILE
```

5.3 SERIES INDEXED BY VECTOR

5.3.1 Scalar data indexed by vector

An example of this kind of data file is the altitude of the spacecraft or rocket, from the ground of the earth (with a given ellipsoid reference), versus latitude and longitude of the point. In this example, the data is a unique and single scalar value (altitude), while the index is a vector type (2 components: latitude and longitude).

5.3.2 Vector data indexed by vector

This can be the case of a magnetic field line model: the data is a series of vectors (the magnetic field), while the index is also a vector (the point of space where is computed the magnetic filed vector).
So one can easily imagine a RFF file containing any scalar, vector, matrix or tensor data, versus an index which can be also a scalar, vector, matrix, even tensor.

Bibliography

[1] Patrick ROBERT. Les procédures Roproc pour le traitement des données de CLUSTER/STAFF-SC, FGM et Orbite. Technical report, Centre d'Etude des Environnements Terrestre et Planétaires, 10-12 Av. de l'Europe, 78140 Velizy, France, september 2003.

[2] Patrick ROBERT. Roproc short documentation, Version 4.3. Technical report, Laboratoire de Physique des Plasmas, Ecole Polytechnique, route de Saclay, 91128 Palaiseau Cedex, January 2011.

[3] A. Allen, S.J. Schwartz, C.C. Harvey, C. Perry, C. Huc, and P. Robert. CSDS Archive Task Group, "Cluster Exchange Format, Data File Syntax", DS-QMW-TN-0010,Issue 2 Rev. 0, Issue 2 Rev. 0. Technical report, ESA, april 2004.

[4] A. Allen, S.J. Schwartz, C.C. Harvey, C. Perry, C. Huc, and P. Robert. Archive task group, "metadata dictionary", draft report. Technical report, ESA, May 2004.

NOTES

write here your personnal notes.