Delta File Format Information JEOL USA, Inc.

October 2010

This document describes the physical data layout of Delta NMR Data Files version 1.2.

This document may be used to translate Delta NMR data into other formats. This document is NOT sufficient to translate other formats into Delta NMR format. There are various semantic and alignment issues which are not discussed here.

The data file is divided into these sections, in this order:

- Header Section
- History Section
- List Section
- Parameter Section
- Data Section
- Context Section
- Annotation Section

The History, Context, and Annotation sections are beyond the scope of this document and are not discussed.

The data file begins with the Header at file offset 0. All fields in the header are always Big Endian format. The offsets for the other sections are stored in the Header. The History and List sections are always stored in big endian format. The Parameter, Data, Context, and Annotation sections may be stored in either big or little endian formats, according to the Endian header field.

Header Section

Each line begins with the name of a header field, followed by its offset from the beginning of the file, its size in bytes, and its type. Detailed descriptions of each field follow.

The file supports up to 8 dimensions, so each field which is listed "per axis" is an array of 8 elements. All comments that refer to Name[n] are refering to the nth element of the array (starting from 1) corresponding to axis n.

Field Name	Offset	Size	Type
File_Identifier	0	8	String
Endian	8	1	Enum
Major_Version	9	1	Unsigned
Minor_Version	10	2	Unsigned
Data_Dimension_Number	12	1	Unsigned
Data_Dimension_Exist	13	1	1-Bit Boolean per axis
Data_Type	14	2/8	Enum (2-Bits)
Data_Format	14	6/8	Enum (6-Bits)
Instrument	15	1	Enum
Translate	16	8	1-Byte Unsigned per axis
Data_Axis_Type	24	8	1-Byte Enum per axis

Data_Units	32	16	2-Byte Unit Structure per axis
Title	48	124	String
Data_Axis_Ranged	172	4	4-Bit Enum per axis
Data_Points	176	32	4-Byte Unsigned per axis
Data_Offset_Start	208	32	4-Byte Unsigned per axis
Data_Offset_Stop	240	32	4-Byte Unsigned per axis
Data_Axis_Start	272	64	Double per axis
Data_Axis_Stop	336	64	Double per axis
Creation_Time	400	4	Time Structure
Revision_Time	404	4	Time Structure
Node_Name	408	16	String
Site	424	128	String
Author	552	128	String
Comment	680	128	String
Data_Axis_Titles	808	256	32-Byte String per axis
Base_Freq	1064	64	Double per Axis
Zero_Point	1128	64	Double per Axis
Reversed	1192	8	1-Bit Boolean per axis
reserved	1200	3	
Annotation_Ok	1203	1/8	1-Bit boolean
reserved	1203	7/8	rest of byte
History_Used	1204	4	Unsigned
History_Length	1208	4	Unsigned
Param_Start	1212	4	Unsigned
Param_Length	1216	4	Unsigned
List_Start	1220	32	4-Byte Unsigned per axis
List_Length	1252	32	4-Byte Unsigned per axis
Data_Start	1284	4	Unsigned
Data_Length	1288	8	Unsigned
Context_Start	1296	8	Unsigned
Context_Length	1304	4	Unsigned
Annote_Start	1308	8	Unsigned
Annote_Length	1316	4	Unsigned
Total_Size	1320	8	Unsigned
Unit_Location	1328	8	1-Byte Unsigned per axis
Extended_Units	1336	24	2 12-Byte Unit Structures

File_Identifier offset 0 size 8 type String

This is used as a file type identifier.
The possible values and their meanings:

"JEOL.NMR" Normal Delta NMR file
"RMN.LOEJ" File was not properly closed,
data may be lost or inconsistent

Endian offset 8 size 1 type Enum

The byte order of the Parameter, Data, Context, and Annotation sections.
Does not apply to Header, History, or List sections, which are always
big endian.

0 = Big Endian
1 = Little Endian

Major_Version offset 9 size 1 type Unsigned

The major version number of this file format.
This value must be 1.

Minor_Version offset 10 size 2 type Unsigned

```
    The minor version number of this file format.
    This value must be 2.

Data_Dimension_Number  offset 12    size 1    type Unsigned
    The number of dimensions of data in this file.
    Valid range is 1 to 8.

Data_Dimension_Exist   offset 13    size 1    type 1-Bit Boolean per axis
    An array of Booleans indicating which dimensions exist.
    bits 7 to 0 correspond to axes x, y, z, a, b, c, d, e
    NOTE: These do not refer to the physical layout of the file, only
          to the displayed layout. (i.e. if only bits 6,5 are set, this
          refers to a 2-D file displayed in the y and z dimensions, the
          files format may be greater that 2-D)

Data_Type              offset 14    size 2/8  type Enum
    The data_type will most often be 0 (64Bits) for all data files.
    Newer systems may choose 1 (32Bits) for data areas larger than 0.5GB
    0 = 64Bit Float
    1 = 32Bit Float
    2 = Reserved
    3 = Reserved

Data_Format            offset 14    size 6/8  type Enum
    The physical layout of data in the file.  See Data Section below for
    information on what each of these means for the data layout.
    1 = One_D
    2 = Two_D
    3 = Three_D
    4 = Four_D
    5 = Five_D
    6 = Six_D
    7 = Seven_D
    8 = Eight_D
    9 - 11 are not for NMR data formats
    12 = Small_Two_D
    13 = Small_Three_D
    14 = Small_Four_D

Instrument             offset 15    size 1    type Enum
    The type of instrument/software this data was originally obtained from.
    0  = NONE
    1  = GSX
    2  = ALPHA
    3  = ECLIPSE
    4  = MASS_SPEC
    5  = COMPILER
    6  = OTHER_NMR
    7  = UNKNOWN
    8  = GEMINI
    9  = UNITY
    10 = ASPECT
    11 = UX
    12 = FELIX
    13 = LAMBDA
    14 = GE_1280
    15 = GE_OMEGA
```

```
    16 = CHEMAGNETICS
    17 = CDFF
    18 = GALACTIC
    19 = TRIAD
    20 = GENERIC_NMR
    21 = GAMMA
    22 = JCAMP_DX
    23 = AMX
    24 = DMX
    25 = ECA
    26 = ALICE
    27 = NMR_PIPE
    28 = SIMPSON

Translate              offset 16    size 8    type 1-Byte Unsigneds
    An array of 8 values that translates from the display dimensions in
    Data_Dimension_Exist to the internal dimensions of the file.  (i.e.
    values of {3,1,2,4,5,6,7,8} means the axis displayed in X is stored
    in axis 3, the y axis is stored in axis 1, the z axis in axis 2).
    A raw unprocessed file should always have values of {1,2,3,4,5,6,7,8}
    Valid range for each element is 1 to 8.

IMPORTANT: All the following fields which are arrays of 8 elements refer
           to the internal layout of the axes NOT the displayed axes.
           Internal layouts always use contiguous dimensions starting with
           axis 1.

Data_Axis_Type         offset 24    size 8    type 1-Byte Enum per axis
    An array of 8 enumerations.  Each element indicates the type of data
    for that axis.  These values interact with Data_Format to determine
    data layout in the Data Section.
    0 = None
        Axis is not used.
    1 = Real
        Axis has real data only, no imaginary.
    2 = TPPI
    3 = Complex
        Axis has complex data.
    4 = Real_Complex
        Axis should be accessed as complex when it is the major axis,
        accessed as real otherwise.  This is only valid when all axes in
        use have this setting.
    5 = Envelope
      Behaves the same way as a Real_Complex dimension but the data
      has different meaning.  Instead of being treated as real and
      imaginary parts of a complex number, the data should be treated
      as minimum and maximum parts of a projection.  This is used
      for the data that results from an envelope projection.

Data_Units             offset 32    size 16    type 2-Byte Unit per axis
    An array of 8 SI units.  Each element is the SI unit to be applied to
    the ruler for that axis.
    The format of each unit:
        Prefix    offset 0 bits 7..4    4-Bit Signed Enum
            The SI prefix.
            -8 = Yotta
            -7 = Zetta
```

```
    -6 = Exa
    -5 = Pecta
    -4 = Tera
    -3 = Giga
    -2 = Mega
    -1 = Kilo
     0 = None
     1 = Milli
     2 = Micro
     3 = Nano
     4 = Pico
     5 = Femto
     6 = Atto
     7 = Zepto

Power     offset 0 bits 3..0    4-Bit Integer
    The power of the unit.  This is signed, so -1 is 1/unit. 0 is
    only valid if unit is None.

Base      offset 1              1-Byte Enum
    The SI base unit (with some extensions).
     0 = None
     1 = Abundance
     2 = Ampere
     3 = Candela
     4 = Celsius
     5 = Coulomb
     6 = Degree
     7 = Electronvolt
     8 = Farad
     9 = Sievert
    10 = Gram
    11 = Gray
    12 = Henry
    13 = Hertz
    14 = Kelvin
    15 = Joule
    16 = Liter
    17 = Lumen
    18 = Lux
    19 = Meter
    20 = Mole
    21 = Newton
    22 = Ohm
    23 = Pascal
    24 = Percent
    25 = Point
    26 = Ppm
    27 = Radian
    28 = Second
    29 = Siemens
    30 = Steradian
    31 = Tesla
    32 = Volt
    33 = Watt
    34 = Weber
    35 = Decibel
```

```
            36 = Dalton
            37 = Thompson
            38 = Ugeneric    <-- Treated as None, but never displayed
            39 = LPercent    <-- Treated as percent for display, but
different for comparison
            40 = PPT         <-- Parts per trillion (Private, do not use)
            41 = PPB         <-- Parts per billion (Private, do not use)
            42 = Index

Title                  offset 48   size 124    type String
    The title to be displayed.  The string is null terminated except when
    all bytes are used.  This is in UTF-8 encoding for newer files, but
    older files may contain ASCII.BEL encoding for non Latin characters.
    This is true for all Strings in the file.

Data_Axis_Ranged      offset 172    size 4    type 4-Bit Enum per axis
    An array of 8 enumerations.  Axis 1 is the high nibble of byte 0, axis
    2 is the low nibble of byte 0, etc.
    0 = Ranged
        The ruler for the axis ranges from Data_Axis_Start[n] to
        Data_Axis_Stop[n] with a step function of
            (Data_Axis_Stop[n] - Data_Axis_Start[n]) /
            (Data_Offset_Stop[n] - Data_Offset_Start[n])
    1 = Listed (deprecated)
        The ruler for the axis is a list of doubles stored in the
        List Section.  Values in the ruler may be anything.
    2 = Sparse
        The ruler for the axis is a list of doubles stored in the
        List Section.  Values in the rulers must be strictly monotonically
        increasing or decreasing.
    3 = Listed
        The ruler for the axis is a list of doubles stored in the
        List Section.  Values in the rulers do not fit definition of Sparse.

Data_Points           offset 176    size 32    type 4-Byte Unsigned per axis
    An array of 8 unsigned integers.  Each element indicates how many data
    points are STORED for each axis.  Some of this range may not be valid.
    Data_Offset_Start[n] and Data_Offset_Stop[n] indicate the valid range
    of data.  A value of 1 indicates the axis is not used.
    Valid values for this depend on the Data_Format.
    e.g.  if Data_Points[1] = 512  Data_Offset_Start[1] = 5
          Data_Offset_Stop[1] = 500, then one vector of data has
          512 data points but the first 5 points and last 11 points
          are not valid data.

Data_Offset_Start     offset 208    size 32    type 4-Byte Unsigned per axis
    An array of 8 unsigned integers.  Each element indicates the offset
    where valid data begins for that axis.
    The valid range is 0 to Data_Offset_Stop[n].

Data_Offset_Stop      offset 240    size 32    type 4-Byte Unsigned per axis
    An array of 8 unsigned integers.  Each element indicates the offset
    where valid data ends for that axis.
    The valid range is Data_Offset_Start[n] to Data_points[n] - 1.

Data_Axis_Start       offset 272    size 64    type Double per axis
    An array of 8 doubles.  Each element is the first value of the
```

```
    ruler (corresponding to Data_Offset_Start[n]).

Data_Axis_Stop        offset 336    size 64    type Double per axis
    An array of 8 doubles.  Each element is the last value of the
    ruler (corresponding to Data_Offset_Stop[n]).

Creation_Time         offset 400    size 4    type Time Structure
    The creation time of the original data file.
    Uses JEOL universal time format.
    The Time Structure contains:

    Year              offset 0 bits 31..25    7-Bit Unsigned
        The Year is offset by 1990 (value 0 is the year 1990).

    Month             offset 0 bits 24..21    4-Bit Unsigned
        Month is 1 to 12.

    Day               offset 0 bits 20..16    5-Bit Unsigned
        Day is 1 to 31.

    Day_Fraction      offset 2                2-Byte Unsigned
        Day Fraction is the 1/65535 part of the day.  Multiply this value by
        86400/65535 (=1.318379) to get seconds since midnight.

    NOTE: Because the fields do not fall on byte boundaries, you can not
          just swap the bytes to read this on a Little Endian machine.

Revision_Time         offset 404    size 4    type Time Structure
    The time of last change to the data portion of the file.
    Uses JEOL universal time format (see description in Creation_Time).

Node_Name             offset 408    size 16    type String
    The name of the computer on which the file was collected / converted.
    The string is null terminated except when all bytes are used.

Site                  offset 424   size 128    type String
    The physical site where the file was collected / converted.  The string
    is null terminated except when all bytes are used.

Author                offset 552   size 128    type String
    The author / owner of the file.  The string is null terminated except
    when all bytes are used.

Comment               offset 680   size 128    type String
    The comment to be display.  The string is null terminated except when
    all bytes are used.

Data_Axis_Titles      offset 808   size 256    type 32-Byte String per axis
    An array of 8 32-Byte strings.  Each element is the title to be
    displayed for that axis.  The string is null terminated except when
    all bytes are used.

Base_Freq            offset 1064    size 64    type Double per axis
    An array of 8 doubles.  Each element is the base spectrometer frequency
    for that axis in Megahertz.  This value is used for hertz <--> ppm unit
    conversion.
```

```
Zero_Point           offset 1128    size 64    type Double per axis
    An array of 8 doubles.  Each element stores the location of 0 on the
    ruler when in hertz.  It is stored as a fraction of the total vector
    length with a bias of 0.5.  i.e. A value of 0 means that ruler value 0
    is exactly halfway between the endpoints, a value of -0.5 means that
    ruler value 0 is exactly on the first data point.

Reversed             offset 1192    size 8    type 1-Bit Boolean per axis
    An array 8 booleans.  Each element indicates whether the data was
    collected in a reverse method (True = NType, False = PType). This
    should only ever be set on data which has not been Fourier transformed.

Annotation_Ok          offset 1203    size 1/8  type 1-Bit Boolean
    True if the annotation database has been verified.

History_Used           offset 1204    size 4    type Unsigned
    The length of the History Section (the amount of space
    that is actually used).
    The History Section starts immediately following the Header at
    offset 1360.

History_Length       offset 1208    size 4    type Unsigned
    Total amount of space allocated for the History Section.
    The History Section starts immediately following the Header at
    offset 1360.

Param_Start          offset 1212    size 4    type Unsigned
    The offset from the beginning of the file where the Parameter Section
    is stored.

Param_Length         offset 1216    size 4    type Unsigned
    The length of the Parameter Section.
    The value is 0 if no Parameter Section is present.

List_Start           offset 1220    size 32    type 4-Byte Unsigned per axis
    An array of 8 unsigned integers.  Each element is the offset from the
    beginning of the file where the ruler list for that axis is stored
    (if it exists).  The list is an array of doubles which must have the
    same number of elements as Data_Points[n].  Data_Offset_Start[n] and
    Data_Offset_Stop[n] apply just as they do for a data vector.


List_Length          offset 1252    size 32    type 4-Byte Unsigned per axis
    An array of 8 unsigned integers.  Each element is the length in bytes
    of the ruler list for that axis.  This value should either be 0 if no
    list is present, or 8 * Data_Points[n].

Data_Start           offset 1284    size 4    type Unsigned
    The offset from the beginning of the file where the Data Section
    is stored.

Data_Length          offset 1288    size 8    type Unsigned
    The length of the Data Section.
    The value is 0 if no Data Section is present.

Context_Start        offset 1296    size 8    type Unsigned
    The offset from the beginning of the file where the Context Section
```

```
    is stored.

Context_Length       offset 1304    size 4    type Unsigned
    The length of the Context Section.
    The value is 0 if no Context Section is present.

Annote_Start         offset 1308    size 8    type Unsigned
    The offset from the beginning of the file where the Annotation Section
    is stored.

Annote_Length        offset 1316    size 4    type Unsigned
    The length of the Annotation Section.
    The value is 0 if no Annotation Section is present.

Total_Size           offset 1320    size 8    type Unsigned
    The total length of the file.
    This value must be less than or equal to the actual length of the file.

Unit_Location        offset 1328    size 8    type 1-Byte Unsigned per axis
    An array of 8 values, one for each axis.  If non-zero indicates which of
    the 2 Compound_Units to use for this axis.  Only 2 axes can have
    compound units.

Compound_Units       offset 1336    size 24    type 12-Byte Compound Unit
Structure
    An array of 2 12-byte unit structures.
    Each Compound Unit consists of:

    Unit_Scaler      offset  0      size  2    Integer
        The units are multiplied by 10**Unit_Scaler if this value is not 0.
        i.e. Megahertz can be stored as either unit:megahertz,scaler:0 or
        as unit:hertz,scaler:6.  Whenever possible this value should be 0.

    Units            offset  4      size 10    5 2-Byte Unit Structures
        An array of 5 Structures.  Each element is a Unit structure.  See
        header field Data_Unit for the definition of this structure.  There
        are 5 of these so that compound units may be expressed.

Parameter Section
-----------------


The Parameter Section starts at the offset specified by the Parameter_Start
header field.  The Parameter Section contains the settings of the
spectrometer and the experiment that were used to collect the data.
This section is optional.

There is a small header on this section which consists of these fields.
The offset is relative to the start of the Parameter Section.

Field Name    Offset  Size    Type
------------  ------  ----    ----
Parameter_Size     0     4    Unsigned
Low_Index          4     4    Unsigned
High_Index         8     4    Unsigned
Total_Size        12     4    Unsigned
```

```
Parameter_Size   offset  0    size 4    type Unsigned
    The size in bytes of one parameter.
    The value should be 64.

Low_Index        offset  4    size 4    type Unsigned
    The low array index.
    The value should be 0.

High_Index       offset  8    size 4    type Unsigned
    The high array index.  The number of parameters is High_Index + 1.

Total_Size       offset 12    size 4    type Unsigned
    The total size of all parameters, not including this header.
    The value should be (High_Index + 1) * 64
```

Immediately following the parameter header at offset 16 is an array of parameters.

Each parameter consists of these fields. The offset is relative to the start of each parameter.

```
Field Name  Offset  Size    Type
----------  ------  ----    ----
Class            0     4    Class Structure
Unit_Scaler      4     2    Integer
Units            6    10    5 2-Byte Unit Structures
Value           16    16    Union
Value_Type      32     4    Enumeration
Name            36    28    String

Class          offset  0    size  4    type Class Structure
    This is an internal structure used for spectrometer control.  Not
    documented here.

Unit_Scaler    offset  4    size  2    type Integer
    The units are multiplied by 10**Unit_Scaler if this value is not 0.
    i.e. Megahertz can be stored as either unit:megahertz,scaler:0 or
    as unit:hertz,scaler:6.  Whenever possible this value should be 0.

Units          offset  6    size 10    type 5 2-Byte Unit Structures
    An array of 5 Structures.  Each element is a Unit structure.  See
    header field Data_Unit for the definition of this structure.  There
    are 5 of these so that compound units may be expressed.

Value          offset 16    size 16    type Union
    The type of this field and the exact size depends on the value of
    Value_Type.  These are the definitions for each Value_Type value.
    The byte order of the value is determined by the Endian header field.
    String
        A 16-Byte string which is padded with spaces.
    Integer
        A 4-Byte integer.
    Float
        An 8-Byte double.
    Complex
        2 8-Byte doubles, the real value first (offset 16) followed by the
```

imaginary value (offset 24).
Infinity
A 4-Byte enumeration.
1 = Negative_Infinity
2 = Minus_One
3 = Zero
4 = Positive_One
5 = Positive_Infinity

```
Value_Type     offset 32    size  4    type Enumeration
    0 = String
    1 = Integer
    2 = Float
    3 = Complex
    4 = Infinity

Name           offset 36    size 28    type String
    This is the parameter name.  The strings are padded with spaces to the
    full length.
```

There is an optional extended parameter store in the context section which encodes
name value pairs in a streaming format.

Data Section

The exact format of the Data Section is determined by both the Data_Format and Data_Axis_Type header fields. All data is stored as doubles with the Endian header field determining the byte order.

This section is divided into multiple sections, one for each double of a (hyper)complex value. The number of sections is therefore determined by the Data_Axis_Type header field.

In general the number of sections is 2**number_of_complex_dims.

The length of each section is simply all the Data_Points multiplied together times 8 bytes.

The format of each section is determined by the Data_Format header field. In specific the format is described as a row major order nD array of row major order nD arrays of doubles. Each format has a fixed size submatrix of doubles, with the number of those submatrices determined by the number of Data_Points. As a result Data_Points[n] must be a multiple of the edge length of that format's specific submatrix.

The submatrix size for each format is given below.

Data_Format	Submatrix Edge	Total Submatrix Points
One_D	8	8 (8**1)
Two_D	32	1024 (32**2)
Three_D	8	512 (8**3)
Four_D	8	4096 (8**4)
Five_D	4	1024 (4**5)

```
Six_D          4               4096  (4**6)
Seven_D        2               128   (2**7)
Eight_D        2               256   (2**8)
Small_Two_D    4               16    (4**2)
Small_Three_D  4               64    (4**3)
Small_Four_D   4               256   (4**4)

Examples:
    1D Real, 512 points
    Data_Axis_Type[1..8] = [Real,None,None,None,None,None,None,None]
    number of sections : 1
    section format     : vector of 512 doubles

    1D Complex, 512 points
    Data_Axis_Type[1..8] = [Complex,None,None,None,None,None,None,None]
    number of sections : 2 (Real, Imaginary)
    section format     : vector of 512 doubles

    2D Real, 256 x 64 points
    Data_Points[1..8]    = [256,64,1,1,1,1,1,1]
    Data_Axis_Type[1..8] = [Real,Real,None,None,None,None,None,None]
    Data_Format          = Two_D
    number of sections : 1
    section format     : 16 total submatrices laid out 8 x 2
                         each submatrix is 32 x 32 doubles

    2D Complex, 512 x 128 points
    Data_Points[1..8]    = [512,128,1,1,1,1,1,1]
    Data_Axis_Type[1..8] =
[Real_Complex,Real_Complex,None,None,None,None,None,None]
    Data_Format          = Two_D
    number of sections : 2 (Real, Imaginary)
    section format     : 64 total submatrices laid out 16 x 4
                         each submatrix is 32 x 32 doubles

    2D Hyper Complex, 256 x 16 points
    Data_Points[1..8]    = [256,16,1,1,1,1,1,1]
    Data_Axis_Type[1..8] = [Complex,Complex,None,None,None,None,None,None]
    Data_Format          = Small_Two_D
    number of sections : 4 (RealReal, RealImag, ImagReal, ImagImag)
    section format     : 256 total submatrices laid out 64 x 4
                         each submatrix is 4 x 4 doubles

    3D Hyper Complex, 128 x 64 x 8 points
    Data_Points[1..8]    = [128,64,8,1,1,1,1,1]
    Data_Axis_Type[1..8] = [Complex,Complex,Complex,None,None,None,None,None]
    Data_Format          = Three_D
    number of sections : 8 (RRR, RRI, RIR, RII, IRR, IRI, IIR, III)
    section format     : 256 total submatrices laid out 16 x 8 x 1
                         each submatrix is 8 x 8 x 8 doubles

This example shows how the Data_Offset fields work with the others.

    2D Hyper Complex, 500 x 100 points
    Data_Points[1..8]       = [512,128,1,1,1,1,1,1]
    Data_Offset_Start[1..8] = [0,0,0,0,0,0,0,0]
    Data_Offset_Stop[1..8]  = [499,99,0,0,0,0,0,0]
```

```
    Data_Axis_Type[1..8]     = [Complex,Complex,None,None,None,None,None,None]
    Data_Format              = Two_D
    number of sections : 4 (RealReal, RealImag, ImagReal, ImagImag)
    section format     : 256 total submatrices laid out 64 x 4
                         each submatrix is 32 x 32 doubles
```

Retrieval of data using this scheme is probably best shown with example code. Here is the generalized routine for finding a particular data point's offset.

```
struct File_Info
{
    ...
    uchar translate[8];
    uint  offset_start[8];
    uint  offset_stop[8];
    uint  submatrices[8];    /* number of submatrices along each dim */
    uint  submatrix_edge;    /* length of an edge of submatrix */
    unit  submatrix_size;    /* number of points in submatrix */
    ...
};

/*
    Given a file and a position, where position[i] is in the range
    0 to (file_info.offset_stop[i] - file_info.offset_start[i])
    return the offset of that position from the beginning of each data
    subsection.
*/
uint file_offset( struct File_Info file_info, uint position[] )
{
    uint i, pos[8], posi;
    uint sub_off = 0;    /* accumulates submatrix offset */
    uint pnt_off = 0;    /* accumulates point offset within submatrix */

    /* translate to internal dimensions */
    for (i = 0; i < 8; i++)
        pos[file_info.translate[i]] = position[i];

    for (i = 7; i >= 1; i--)
    {
        posi    = pos[i] + file_info.offset_start[i];
        pnt_off = (pnt_off + posi % file_info.submatrix_edge) *
file_info.submatrix_edge;
        sub_off = (sub_off + posi / file_info.submatrix_edge) *
file_info.submatrices[i-1];
    }
    posi    = pos[0] + file_info.offset_start[0];
    pnt_off = pnt_off + posi % file_info.submatrix_edge;
    sub_off = sub_off + posi / file_info.submatrix_edge;
    return sizeof (double) * (sub_off * file_info.submatrix_size + pnt_off);}
```

Digital Filter Phase Correction Information

$$\Theta(\omega) = \frac{\omega\pi}{2} \sum_{i=1}^{orders_1} \frac{orders_{i+1} - 1}{\prod_{j=i}^{orders_1} factors_j} \left(-\frac{1}{2} \le \omega < \frac{1}{2} \right)$$

Orders and factors are lists of space separated integers. So to process digitally filtered data you have to Fourier transform it, apply the above phase, then inverse transform it. The data will then look like a normal FID, with the data decreasing in magnitude from the first point, but there will be an increasing portion of the data near the end of the FID. This region actually corresponds to data at a negative time and care has to be taken when zero filling or applying window functions to treat it properly or artifacts will appear in the spectrum. For example to zero fill the data, the phase shift from the digital filters first has to be reapplied, then the zero filling, and then the phase shift has to be undone.

There is another parameter called digital_filter_status in the data, if it contains a capital P then the phase shift has already been applied to the spectrum and the delayed points are at the end of the FID, a small p indicates that the delay is at the beginning of the FID. You will have to pay attention to this parameter to determine what adjustments are necessary to processing.