Keywords for SAXS Data in EDF Files

EDF_DataFormatVersion = 2.30

Peter Boesecke

1. General Properties (EDF)

EDF files contain one or more sets of data blocks consisting of an ASCII (characters 1-127) readable header and binary data. The header consists of keyword-value pairs. The header contains ASCII text and is therefore easily readable. Nevertheless a dictionary is required to make full use of its information. It is not suggested to change the header manually with a text editor because it might corrupt the data file.

Short Description

The header must start by a line feed '\n' and an opening curly brace '{'and must end after a closing curly braze '}' followed by a line feed '\n'. Files with similar but other start and stop patterns are not considered in this document.

Keywords are not case sensitive. A keyword is always followed by an equal sign that separates it from its value. Each value is terminated by a semicolon ';'.Values are trimmed, i.e. leading and trailing white space is removed. A single leading and a single trailing double prime (") is removed.

Important: An ASCII-zero('\0') is NEVER allowed inside a header, because it is used as an emergency stop if the header end is missing. Curly-braces and semicolons are not allowed inside a header and therefore also neither in keywords nor in values because they are used as control characters. Escape sequences starting with a backslash must be used instead ("\(" for "{", "\)" for "}", "\:" for ";" "\\" for "\" and "\l" for a new line (lf or cr/lf depending on the operating system), see section Backslash escape sequences).

For convenience, the header length is limited to multiples of 512 bytes (including start and end pattern). A cr/lf is inserted after each semicolon following each keyword-value pair since it allows the header section to be viewed with standard system utilities on a very wide range of operating systems.. cr´s and lf´s are generally ignored in values but they should not be used inside numbers.

Keywords

The maximum significant length of a header value is 64 (MaxKeyLen) characters. Keywords are not case sensitive and white spaces are removed.

Types of Header Values

The maximum significant length of a header value is 512 characters (MaxValLen). String values should not exceed 256 characters (counting also backslash escape sequences (String Value) and final zero('\0') in the output string). Too long header values are truncated.

String Value

A string value is a sequence of ASCII characters that does not contain curly braces or a semicolon. These characters are expressed by backslash escape sequences.

Backslash escape sequences

A single '\' at the end of a string is ignored.

Tab. 1: Transformation of a string value to a string.

string value		string
'\"l'	=>	line_feed
'\"r'	=>	'\r'
'\"n'	=>	'\n'
'\"s'	=>	' '
'\"t'	=>	'\t'
'\"v'	=>	'\v'
'\"f'	=>	'\f'
'\"('	=>	'{'
'\")'	=>	'}'
'\":'	=>	';'
'\r'	skipped	-
'\n'	skipped	-
'\";'	end of the string	-
'\'<any other character>	=>	<any other character>

Tab. 2: Transformation of a string to a string value

'\r'\n'	=>	'\"l'
'\n'	=>	'\"l'
'{'	=>	'\"('
'}'	=>	'\")'
';'	=>	'\":'
'\'	=>	'\"\'

Long Integer Value

Decimal number with or without a sign (+|-), example +1234567890. Currently only long integer values are used.

Double Float Value

A decimal number with or without a sign (+|-), with a point ('.') as decimal separator and an 'e' as decimal exponent separator, example: -1.234e+2 for -123.4. Currently only double float values are used.

Time Value

Time values should be written in the following subset of the ISO time notation (YYYY = year, MM: month, DD: day, hh: hour (0-24), mm: minute (0-60), ss: seconds (0-61), ssssss: microseconds ([] optional)):

(1) YYYY-MM-DD hh:mm:ss[.ssssss]

Example: "1998-01-02 12:34:56.000000" for 02-Jan-1998 12:34:56.

Information about ISO 8601 time notation can be found at http://www.cl.cam.ac.uk/~mgk25/iso-time.html.

2. Data Format Specific (EDF)

All data format specific keywords are preceded by "EDF_" and are listed at the top of a header section. "EDF_" keywords that are not listed at the top of a header section are ignored. For convenience some of them are written in a specific order. Data format specific keywords are only defined in EDF_DataFormatVersion >= 2.00. For all other files default values are used. If the file starts with "\n{\r\nEDF_" it is a version >= 2 file. EDF_DataFormatVersion 1 is assumed if the file starts with "\n{".

Header Start Pattern

The start pattern of a header is "\n{" (Version >=2: "\n{\r\nEDF_". It can be used as a magic pattern to identify the file type).

Header End Pattern

The end pattern of a header is "}\n" (Version >=2: "\r\n}\n")

General Block Header Keywords (EDF)

Optional, a general header is only defined in Version>=2. The start pattern of a general header header is "\n{\r\nEDF_DataFormatVersion ". It can be used as a magic pattern to identify a file with general header).

V2.1: The general block header starts with data format specific keywords (starting with "EDF_") and can be followed by application specific keywords (any keyword not starting with "EDF_"). All application specific keywords in the general block header are used as default values. They are used if the keyword is not given in the data block header.

EDF_DataFormatVersion

Description

Obligatory the first keyword in a general header. It describes the version of the data file format

Value

N.mm, where N is a positive number and mm are two numeric characters, e.g. 2.10

Default

1.00

EDF_DataBlocks

Description

Obligatory second keyword in a general header. The value is the number of data blocks in the file.

Value

Positive long integer number larger or equal to 1. If the file contains only a single data block the value can be replaced by "Undetermined".

Default

Undetermined

EDF_BlockBoundary

Description

Optional: The data block length (default: 512 bytes)

Remark: If the data consists of a pure text header file (EDF_BinarySize = 0) and separate binary files (EDF_BinaryFileName = "<binary filename>") the text header file can be written with block boundary 1. The extension of a pure text header file should be ".ehf".

Value

Long integer value larger than 0.

Default

512

Data Block Header Keywords (EDF)

EDF_DataBlockID

Description

The value of the data block identifier must be unique in the file. It must be the first keyword in a data block header, if it is not the general block (see General Block Header Keywords).

Value

Format

<sequence>.<class>.<instance>[.<memory>]

<sequence>

Positive or negative long integer value describing the order of the data blocks. In time sequences the sequence number n should usually increase with the start time t(n) (t(n)<t(n+1)). The sequence number is independent of the physical order of the data blocks in the file. Usually, the sequence number of the first data block in the file is n=1 and the sequence number of each following data block is incremented by 1.

<class>

String describing the data class, currently used: "Image". An image is a n-dimensional array of rastered data and must be described by the keywords "Dim_1" to "Dim_n", "RasterOrientation" and "DataType".

<instance>

"Psd": extension for primary (scientific) data.

"Error": extension for error data

The extensions "FlatField" an "Mask" are reserved, but currently not used.

<memory>

A positive long integer number. It is used in analogy to the "bsl" format to store calibration data or data from other detectors. The extension ".1" must be omitted for memory=1.

Examples

Sequence 1 to 3, primary data in memories 1 and 2

EDF_DataBlockID = 1.Image.Psd ;

EDF_DataBlockID = 2.Image.Psd ;

EDF_DataBlockID = 1.Image.Psd.2 ;

EDF_DataBlockID = 2.Image.Psd.2 ;

Sequence 1 to 3, error data in memories 1 and 2

EDF_DataBlockID = 1.Image.Error ;

EDF_DataBlockID = 2.Image.Error ;

EDF_DataBlockID = 1.Image.Error.2 ;

EDF_DataBlockID = 2.Image.Error.2 ;

Default

no default

Version

2.30 (redefined)

EDF_BinarySize

Description

Obligatory It is equal to the number of bytes that follow after the header.

Value

Long Integer Value

Default

0

EDF_HeaderSize

Description

Optional: The length of the header section in bytes. It is currently ignored.

Value

Long Integer Value

Default

no default

EDF_BinaryFileName

Description

Optional: Name of an **EXTERNAL** file containing the binary data. It must be in the same directory as the EDF file and must be **DIFFERENT** from this EDF file.

Value

Filename without a path.

Default

no default

EDF_BinaryFilePosition

Description

Obligatory together with EDF_BinaryFileName, otherwise ignored. Start position of the data in the external file.

Value

Long Integer Value

Default

no default

EDF_BinaryFileSize

Description

Optional: The size of the data buffer in the external file (currently ignored)

Value

Long Integer Value

Default

no default

3. Description of Binary Data (EDF)

Keywords

ByteOrder

Description

This keyword shows whether the more significant data bytes of a single data element are stored before or after the less significant data bytes.

Values

HighByteFirst (default), LowByteFirst

HighByteFirst

The binary data are stored in big endian format.

LowByteFirst

The binary data are stored in little endian format.

Default

HighByteFirst

Compression

Description

This keyword describes the compression of the binary data.

Remark: It should be distinguished from other types of compressions that are applied to specific data (e.g. 2d data) and that can only be applied after endian correction and reoordering. The latter might be called RasterDataCompression (proposal).

Values

None (default)

None

No compression

DataType

Description

The data type of each data element. It defines, according to the following list, type and size of data elements. The following data types are currently defined. Not all of them are used (see Tab. 3:). Aliases are used for backward compatibility.

Values

Unsigned8, Signed8, Unsigned16, Signed16, Unsigned32, Signed32, Unsigned64, Signed64, FloatIEEE32, DoubleIEEE64, UnAssigned, UnAssigned, FloatVAX32, DoubleVAX64, FloatConvex32, DoubleConvex64

Aliases

UnsignedByte, SignedByte, UnsignedShort, SignedShort, UnsignedInteger, SignedInteger, Unsigned64, Signed64, FloatValue, DoubleValue, UnAssigned, UnAssigned, FloatVAX32, DoubleVAX64, FloatConvex32, DoubleConvex64

Tab. 3: Data type values

value	data size / bytes	data type	comments
Unsigned8	1	unsigned 8 bit integer	
Signed8	1	signed 8 bit integer	
Unsigned16	2	unsigned short	
Signed16	2	signed 16 bit integer	
Unsigned32	4	unsigned 32 bit integer	
Signed32	4	signed 32 bit integer	
Unsigned64	8	unsigned 64 bit integer	currently unused
Signed64	8	signed 64 bit integer	currently unused
FloatIEEE32	4	32 bit IEEE float	
DoubleIEEE64	4	64 bit IEEE float	
FloatVAX32	4		currently unused
DoubleVAX64	8		currently unused
FloatConvex32	4		currently unused
DoubleConvex64	8		currently unused

Default

FloatIEEE32

DataValueOffset

Version

2.20

Description

The actual data Value is calculated from the binary Data by the following transformation (c-type notation):

(2) DataType Value;

(3) long DataValueOffset;

(4) Value = (DataType) Data + (DataType) DataValueOffset;

The offset is added after a data type conversion of each binary data item and of itself. The offset is added at the very end, after endian correction and binary decompression.

Values

Long Integer Value

Default

0

N-dimensional Raster Data (Arrays, Images and Volumes) Dim_(n)

Keys are indexed with an underscore followed by a number, e.g."Dim_3" for the array length *Dim* in direction *3*. Keywords ending with an underscore followed by a number are automatically interpreted as a coordinate number and must not be used for other purposes, e.g. for other indexed data, like parameters of a multi channel counter.

Description

The keys Dim_1 to Dim_(n) are used to describe a n-dimensional raster array. Dim_(n) is the number of array elements in dimension n, where n is a positive integer larger than zero. *RasterOrientation* defines the order of the array elements in the linear data buffer and *Dim*_(n) the number of array elements in direction n. If Dim_(J) is found in the header, but not Dim_(J+1) the array has the dimension J. The default for Dim_1 is zero, for all others 1. This allows to read the data array always as a n-dimensional array, e.g. to read a m-dimensional array A[Dim_1, ..., Dim_(m)] as a n-dimensional array A[Dim_1, ..., Dim_(m),1, 1, ..., 1] by setting the higher indizes to 1 or by shortening a m-dimensional array to its first n-dimensions. The values of the key *RasterOrientation* are defined in such a way that this is easily possible.

One dimensional data can be stored in two dimensional arrays with Dim_2 = 1.

Values

Positive integer values (currently limited to long integers)

Default

The default for Dim_1 is zero, for all others 1.

RasterOrientation

Description

The raster orientation describes the storage order of array elements in a binary data buffer. The raster orientation is 1 when a change in a lower index corresponds to a smaller distance between corresponding data elements in the stored data. In other words: "a smaller index runs faster". The default orientation is 1. RasterOrientation is only used to convert between different ways of data storage and not to convert between different coordinate systems. In the second case all indexed keys (keys with an underscore and a number) must be reordered in a similar way as the indices, e.g. when the coordinates are swapped. This must be done in a later step because it requires specific knowledge about the specific coordinate sytem and its description.

The raster orientation of a N-dimensional stored array is defined recursively and has a value between 1 and 2^N*N! For example, the elements of a one-dimensional array can either be stored in ascending or in descending order. In the first case the raster orientation is 1 in the latter case 2.

In the two dimensional case the number of possible raster orientations is 8. The two first cases are identical to the one dimensional case, only a second index is added after the first index. In the following two cases the storage direction of the second index is inversed. In the four last cases the data elements of the second index appear faster than the data elements of the first index.

The raster orientation of N-dimensional data is given by an N-tupel of indices that corresponds to the order from fast to slow in which the data elements are stored, with a negative sign if the direction is inverted. When the second index is the fastest and the direction of the first index is inverted the corresponding tupel would be: (2,-1).

This is raster orientation number 6 (see following figure). The raster orientation number and the n-tupel are equivalent.

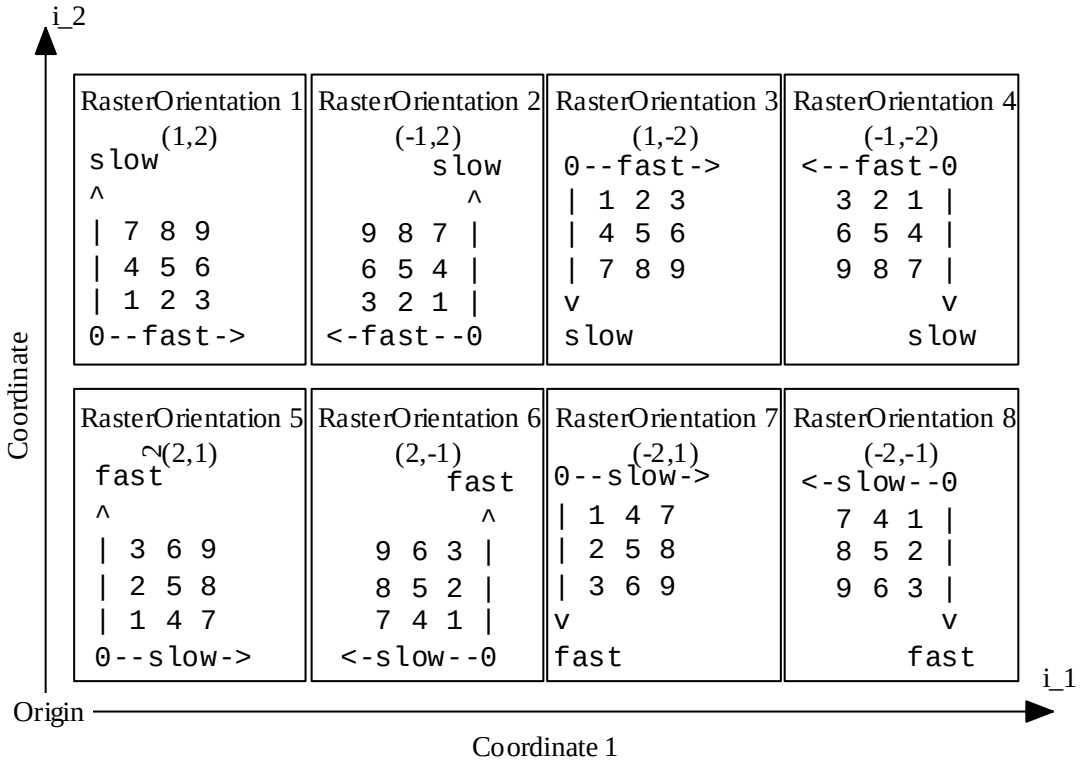

Fig. 1: Raster orientations for N=2 (two-dimensional image). The geometrical origin of the image is always at the lower left corner, index 1 corresponds always to the (horizontal) coordinate 1 and index 2 corresponds always to the (vertical) coordinate 2.

It should be noted that in the cases 5 to 8 the order of the indices is swapped but not the array dimensions. Data with dimensions Dim_1 = 500 and Dim_2 = 600 that has been stored in raster orientation 1 to 4 must be stored in raster orientation 5 to 8 with Dim_1 and Dim_2 unchanged..

If the concept of raster orientation is used to swap binary data the array dimensions and all dimension dependent keywords, like offset , pixel size and center must be written in that way that Dim_1 etc. corresponds after the swap to coordinate 1 and Dim_2 etc. to coordinate 2.

This concept was extended generally to more than 2 dimensions.

Values

Integer values from 1 to A(N)=2^N * N! where N is the dimension. Currently only long integer value are used. For N>9 the number of raster orientations is larger than the value range of long integers. In these cases only raster orientations from 1 to 2^9*9! are possible.

Default

1

Pixel Coordinates

The pixel (or data) coordinate along each direction is expressed by a float type number rather than a integer pixel number. The coordinates start with p0=0.0 at the lower edge of the first pixel (with pixel number 1) and end at the upper edge of pixel number n with the coordinate n. Read access to coordinates outside these limits should not cause an error but should return a dummy values as result. Write access to these coordinates will be ignored.

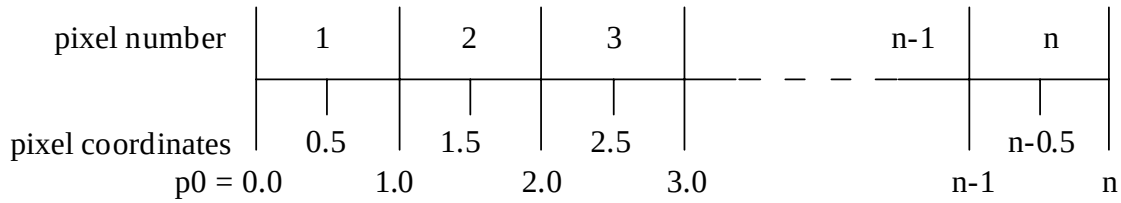

Fig. 2: Pixel coordinates

4. Data History (SAXS)

In version 2.11 and higher the history of command lines are stored in the file header. They can be used to repeat the calculation.

Keywords

DataHistory-U

Description

Each data history keyword is followed by a command line with all options and arguments. The history starts with DataHistory-1. Each call to a function that produces a history line adds a new keyword with the unsigned integer number U incremented by 1. The most recent command line has the highest number U.

The format of the history line is free, but should enable to reproduce the step.

Values

The values are of the type String Value.

5. Description of the Scattering Geometry (SAXS)

The geometry is only defined for two dimensional scattering data. One dimensional data can be represented by setting the second index to 1.

Image Coordinates

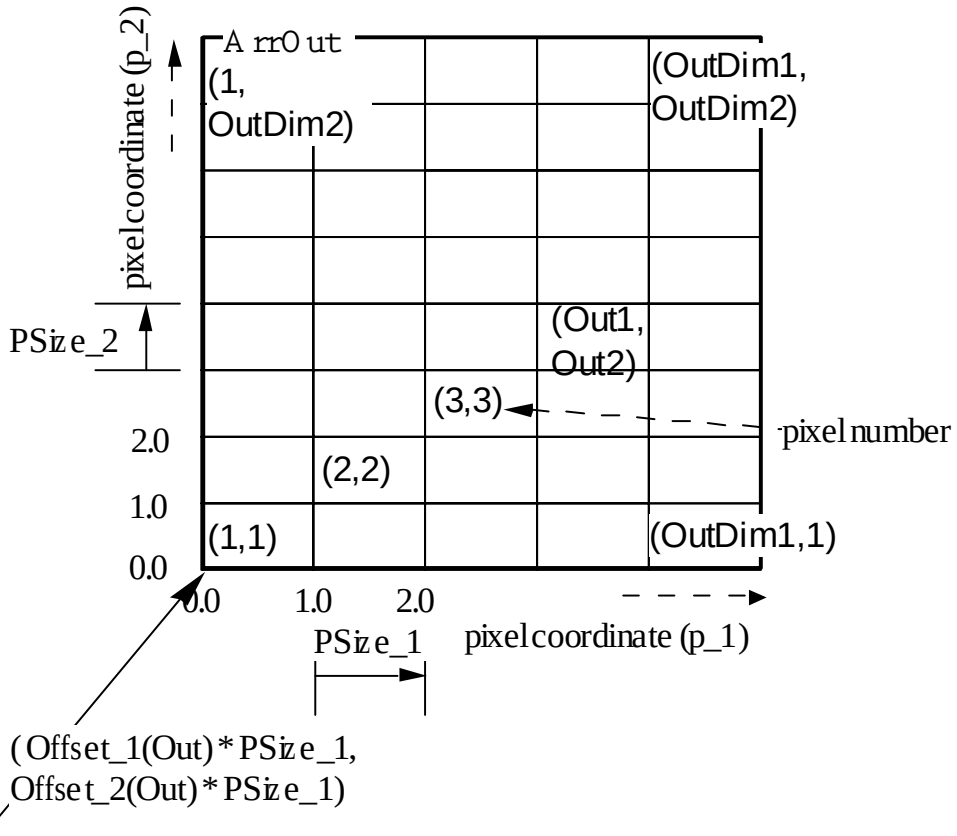

Fig. 3: Two dimensional detector coordinate system. The detector is seen from the sample.

Reference Systems

Currently the following reference system are defined. All reference systems are defined for the same data set.

The simplest description is ARRAY. It uses only the pixel coordinates without any additional keyword.

The standard coordinate system is the IMAGE coordinate system. It consist of the ARRAY coordinates plus an offset.

REAL coordinates consists of IMAGE coordinates multiplied with a pixel size.

In the SAXS coordinate system the coordinates are oriented around a center. The distance from the center is given by tan(2Theta)/(wavelength/wavelength0), where

wavelength0=1e-9 has been chosen for convenience. For small 2-theta angles the resulting coordinate can be approximated by s=2sin(Theta)/wavelength [nm], where s is the scattering vector in nanometers. At large scattering angles this coordinate system is still exact, but the coordinates cannot be interpreted directly as scattering vectors. The tan(2*Theta) must be converted into 2*sin(Theta).

The SAXS coordinate system allows the (exact) combination of scattering data from different detectors (small and wide angle), which are mounted perpendicular to the primary beam. Or from the same detector placed at different distances from the sample.

A further extension, which could be called WAXS, could even take into account detectors which are mounted under an angle with respect to the primary beam.

(5)

ARRAY coordinate p	= pixel coordinate
IMAGE coordinate i	= array coordinate + offset
REAL coordinate r	= image coordinate * pixel size
SAXS coordinate s	= (image coordinate - center) * (pixel size/sample distance) * (WaveLength0/wave length)

Binning

A binning of the regular raster array by an integer factor B (1,2, ...) in one direction leads to the following parameter transformations:

(6)
$$p' = p * B$$
$$Offset' = p0 \times (1 - B) + Offset / B$$
$$Center' = Center / B$$

The detector data is described by the pixel size p [m], an offset [pixel] and a center coordinate [image coordinate]. Because the coordinate system was chosen in such a way that p0=0.0 (see Fig. 2:), the transformation of the offset simplifies to:

(7)
$$Offset' = Offset / B$$

SAXS Geometry

The principal SAXS geometry is shown in Fig. 4:. The data is described with an origin in the lower left corner. The detector is seen from the sample ("with the beam"). The data can be treated in one of the four reference systems that are shown in (5).

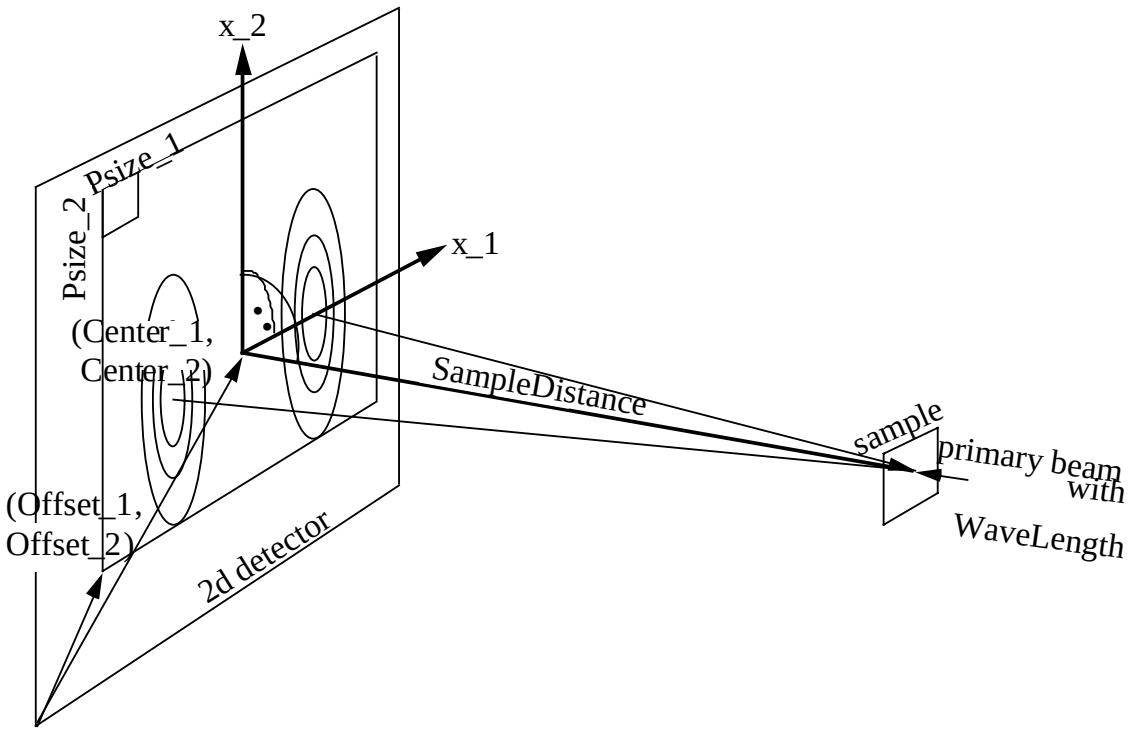

Fig. 4: SAXS coordinate system: The primary beam with a monochromatic WaveLength is coming from the right side and is scattered by the sample. The active detector surface is perpendicular to the beam. The normal incidence point of the primary beam on the detector surface is defined by Center_1 and Center_2. A detector subset area, defined by Offset_1 and Offset_2, is taken into account only. The horizontal and vertical pixel size of the detector is Psize_1 and Psize_2, the distance between the sample and the detector is given by SampleDistance.

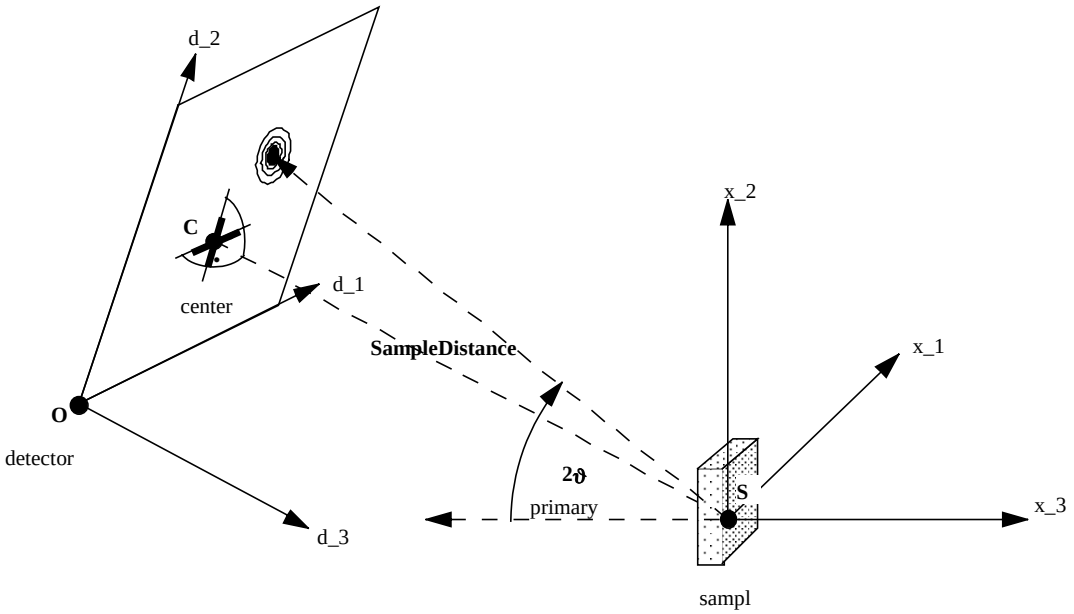

Fig. 5: Extension of the description to inclined geometry (WAXS). Three additional parameters must be added to SAXS: RotationX_1, RotationX_2, RotationX_3, where RotationX_1 stands for a CCW rotation around axis X_1, that corresponds to the direction of index 1 in the unrotated case.

Keywords

SaxsDataVersion

Description

Describes the version of the definitions

Values

current version 2.0

Default

0

Offset_1, Offset_2

Description

Offset of coordinate 1 and 2 of the data array as shown in Fig. 3:

Unit

array coordinate (pixel)

Value

Double Float Value

Default

0.0, 0.0

PSize_1, PSize_2

Description

Pixel size of coordinate 1 and 2 of the data array as shown in Fig. 3:

Unit

meter

Value

Double Float Value

Default

no default

Center_1, Center_2

Description

Coordinate 1 and 2 of the normal incidence point of the beam on the detector as shown in Fig. 4: If the detector surface is perpendicular to the beam Center_1 and Center_2 are identical to the point where the primar beam would hit the detector.

Unit

image coorinate (pixel)

Value

FloatValue

Default

no default

WaveLength

Description

Monochromatic wavelength of the beam.

Unit

meter

Value

FloatValue

Default

no default

SampleDistance

Description

Distance between the sample and the normal incidence point (Center_1, Center_2) of the beam on the detector.

Unit

meter

Value

FloatValue

Default

no default

Title

Description

A title of the data block.

Value

Character string with escape sequences (to avoid curly braces and semicolons, see paragraph String Value). Strings longer than MaxValLen (currently 512) are truncated. There are still some subroutines which can only pass 255 characters.

Default

no default

Dummy

Description

A dummy is a value that represents an invalid data point. Zero (0.0) can never be used as a dummy value. All values between Dummy-DDummy and Dummy+DDummy are

taken as invalid data points. If -DDummy < Dummy < +DDummy no dummy value is defined.

Value

FloatValue

Default

0.0 (no dummy)

DDummy

Description

Radius around Dummy to represent a dummy value.

Value

positive FloatValue

Default

max(0.1, 1e-4 * Dummy)

6. Raw Data (SAXS)

The following keywords have been defined ad hoc for immediate use at the beamlines ID01 and ID02. They are only used in conversion routines of the form saxs_norm... which convert the raw data into the standard representation.

Info Keywords

All keywords ending with "Info" contain a free format string value about the corresponding item.

DetectorInfo

Description

Info string about detector

Value

String Value

Default

no default

ExperimentInfo

Description

Info string about the experiment in general

Value

String Value

Default

no default

MachineInfo

Description

Info string about machine status or setup

Value

String Value

Default

no default

OpticsInfo

Description

Info string about optical setup

Value

String Value

Default

no default

ProposalInfo

Description

Info string about the proposal

Value

String Value

Default

no default

StationInfo

Description

Info string about the station

Value

String Value

Default

no default

SubTitle

Description

A general description of the experiment, should be replaced by the keyword ExperimentInfo

Value

String Value

Default

no default

Detector Keywords

Keywords starting with "Detector" are used to describe detector properties.

DetectorPosition

Description

The position of the detector on the optical bench, starting with 0.0 a the end stop close to the sample. The distance between sample and detector is calculated from this by adding a constant.

Value

Double Float Value

Default

no default

DetectorName

Description

Name of the detector (currently free format)

Value

String Value

Default

no default

Multichannel Scaler Keywords

A 32 channel scaler is used to count calibration data during each exposure (incoming photons, transmitted photons, exposure time, anode couts). All keywords starting with "HS32" are related to the 32 channel scaler.

A scaler value is calculated by

(8) ScalerValue = (ScalerCounts - ScalerZero * TimerValue) * ScalerFactor

ScalerZero is used to subtract an offset from a signal of a voltage to frequency converter. The units are as follows: ScalerCounts [counts], ScalerZero [counts/s], TimerValue [s], ScalerFactor [<unit>/counts], where <unit> stands for the physical unit that is measured by the scaler. The keywords for ScalerCounts, ScalerZero and ScalerFactor are shown below.

At least one scaler must be fed with a constant frequency, e.g. 1MHz, and must be used as a timer. The timer value is calculated by

(9) TimerValue = TimerCounts * TimerFactor

The units are as follows: TimerValue [s], TimerCounts [counts], TimerFactor [s/counts]. The scaler that is actually used as timer is given by the keyword "HSTime" (see below).

HS32Len

Description

Number of scalers, from 1 to HS32Len <= 32

Value

Long Integer Value

Default

32

HS32C<nn>

Description

ScalerCounts (see eq. (8))

Syntax

All keywords starting with HS32C... are followed by a two digit channel number (HS32C01, HS32C02, ..., HS32C10, ...). The maximum scaler number is given by HS32Len.

Value

Double Float Value

Default

no default

HS32N<nn>

Description

The name of the corresponding scaler

Syntax

All keywords starting with HS32S... are followed by a two digit channel number (HS32S01, HS32S02, ..., HS32S10, ...). The maximum scaler number is given by HS32Len.

Value

String Value

Default

no default

HS32Z<nn>

Description

ScalerZero (see eq. (8)). The zero value of the scaler.

Syntax

All keywords starting with HS32Z... are followed by a two digit channel number (HS32Z01, HS32Z02, ..., HS32Z10, ...). The maximum scaler number is given by HS32Len.

Value

Double Float Value

Default

no default

HS32F<nn>

Description

ScalerFactor (see eq. (8)). The multiplication factor for the corresponding scaler.

Syntax

All keywords starting with HS32F... are followed by a two digit scaler number (HS32F01, HS32F02, ..., HS32F10, ...). The maximum scaler number is given by HS32Len.

Value

Double Float Value

Default

no default

HS32Depth

Description

Double float value that shows the dynamics of the scaler counts. The multi channel scaler has only a dynamics of 24 bits. With an input frequency of 1MHz the counter gets an overflow after approximately 16 seconds. To allow longer counting times the counts of the multi channel scaler are accumulated externally in software. The dynamics of the software counters after k=0, 1, 2, etc. external accumulations is calculated with

(10) HS32Depth = (ln(k+1)/ln(2.0)) + 24.0

With HS32Depth the maximum scaler count is given by

(11) max_count = exp(ln(2)* HS32Depth)

Value

Double Float Value

Default

24.0

Default

no default

Binary Data Keywords

All keywords starting with HM (Histogramming Memory) are related to the binary data which were originally data from a histogramming memory for a multiwire proportional chamber.

HMFrame

Description

Frame number of the binary data in HMFile, starting with 1

Value

Long Integer Value

Default

no default

HMFile

Description

File name and path of the binary file that was originally created by the acquisition program.

Value

String Value

Default

no default

HMStartTime

Description

Absolute start time of the experiment (currently still in unix date notation)

Value

Time Value

Default

no default

HMDeltaTime

Description

Relative start time in seconds of the exposure of this image relative to the absolute start time of the experiment (see Default

no default

HMDeltaTime).

Value

Double Float Value

Default

no default

Keywords to Assign Scaler Numbers to Incoming Photons, Transmitted Photons, Anode Counts and Exposure Time

The following keywords assign scalers to specific measurements. Each value can be measured by a primary and a secondary (alternative) scaler. If a scaler is not defined or used the value after the keyword is 0.

The input frequency of the secondary scaler must be a divided frequency of the primary scaler. The calibration factors and zero values must be correctly set for both scalers.

If an alternative scaler exists it is checked for an overflow of the primary scaler. The maximum allowed value of the primary scaler (without subtraction of the zero value) is given by

(12) MaximumScalerValue' = ScalerFactor*exp(ln(2)* HS32Depth)

The scaler value without zero level subtraction (') is given by

(13) ScalerValue' = ScalerFactor*ScalerCounts1

(14) ScalerValueS' = ScalerFactorS*ScalerCountsS

If the scaler value of the secondary scaler (ScalerValueS') is larger than 99% of MaximumScalerValue' the value is calculated from the second scaler, otherwise from the primary scaler.

HSI0

Description

Scaler number of the I0 monitor (flux before sample) (1 .. HS32Len). The I0-value is the number of photons in the primary beam during the exposure.

Unit of ScalerValue

#photons

Value

Long Integer Value

Default

0

HSI0S

Description

Scaler number of the alternative I0 monitor (flux before sample) (1 ... HS32Len)

Unit of ScalerValue

#photons

Value

Long Integer Value

Default

0

HSI1

Description

Scaler number of the I1 monitor (flux after sample) (1 .. HS32Len)

Unit of ScalerValue

#photons

Value

Long Integer Value

Default

0

HSI1S

Description

Scaler number of the alternative I1 monitor (flux after sample) (1 ... HS32Len)

Unit of ScalerValue

#photons

Value

Long Integer Value

Default

0

HSAnode

Description

Scaler number of the anode pulse counter

Unit of ScalerValue

#counts

Value

Long Integer Value

Default

0

HSAnodeS

Description

Scaler number of the alternative anode pulse counter

Unit of ScalerValue

#counts

Value

Long Integer Value

Default

0

HSTime

Description

Scaler number of the time counter

Unit of ScalerValue

seconds

Value

Long Integer Value

Default

0

HSTimeS

Description

Scaler number of the alternative time counter

Unit of ScalerValue

seconds

Value

Long Integer Value

Default

0

Keywords for Explicit Values of Incoming Photons, Transmitted Photons, Anode Counts and Exposure Time

Version

2.10

Intensity0Monitor

Description

Name of the I0 monitor (beam intensity monitor before the sample)

Value

String Value

Default

no default

Intensity1Monitor

Description

Name of the I1 monitor (beam intensity monitor after the sample)

Value

String Value

Default

no default

AnodeCountsMonitor

Description

Name of the anode counts monitor

Value

String Value

Default

no default

ExposureTimeMonitor

Description

Name of the exposure time monitor

Value

String Value

Default

no default

Intensity0

Description

Integrated number of photons in the primary beam before the sample during the exposure.

Unit

#photons

Value

Float Value

Default

no default

Intensity1

Description

Integrated number of photons in the primary beam after the sample during the exposure. Attention, the ratio Intensity1/Intensity0 can be larger than the sample transmission if only a part of the primary beam hits the sample.

Unit

#photons

Value

Float Value

Default

no default

AnodeCounts

Description

Integrated number of anode counts during the exposure.

Unit

#counts

Value

Float Value

Default

no default

ExposureTime

Description

Integrated exposure time

Unit

seconds

Value

Float Value

Default

no default

ESRF ID01 Specific Keywords

Version

2.10

ESRF_ID01_TDC_ResolutionMode

Description

Internal code of the time to digital converter for its resolution mode

Value

Long Integer Value

Default

no default

ESRF_ID01_TDC_Resolution

Description

TDC resolution in seconds

Unit

seconds

Value

Float Value

Default

no default

ESRF_ID01_TDC_Offset

Description

TDC offset in seconds

Unit

seconds

Value

Float Value

Default

no default

ESRF_ID01_TDC_Timeout

Description

TDC time out in seconds

Unit

seconds

Value

Float Value

Default

no default

ESRF_ID01_Goniometer_Theta

Description

Theta angle of the esrf id01 goniometer

Unit

degree

Value

Float Value

Default

no default

ESRF_ID01_Goniometer_Chi

Description

Chi angle of the esrf id01 goniometer

Unit

degree

Value

Float Value

Default

no default

ESRF_ID01_Goniometer_Chi

Description

Chi angle of the esrf id01 goniometer

Unit

degree

Value

Float Value

Default

no default

ESRF_ID01_Goniometer_Phi

Description

Phi angle of the esrf id01 goniometer

Unit

degree

Value

Float Value

Default

no default

7. Backward Compatibility (EDF and SAXS)

This section describes keywords that are only used for backward compatibility and which might disappear in the future.

The following keywords are not required any more and can be omitted if the file will not be read by old software.

Keywords

HeaderID

Description

Data block header identification starting with *EH:* and consisting of three 6 digits long unsigned decimal numbers separated by colons. Only the first number is used and larger than zero. The headers are numbered in ascending order starting with 1.

Value

Syntax

EH:<header_id>:000000:000000, with <header_id> = 1, 2, ...

Example

The first header in the data file looks like this:

HeaderID = EH:000001:000000:000000 ;

Size

Description

Size of the binary section after the header in bytes.

Value

This keyword duplicates the value after Default

no default

Version

2.30 (redefined)

EDF_BinarySize.

Image

Description

This value duplicates the sequence number of the data block identifier (EDF_DataBlockID) for the case that the data block identifier ends with "Image.Psd"

Value

Syntax

String Value ("All") or Long Integer Value ("1"). All data blocks with valid data have a Long Intger Value.

Example

Image = 1;

Keywords in General Header

The keys "DataKey-<memory>" are used to redefine "<class>.<memory>" identifiers. "DataKey-1" corresponds always to "Image.Psd". The defaults of "DataKey-2", "DataKey-3" etc. correspond always to "Image.<memory>. These keywords are only usd in the general header.

DataKey_<memory>

Description

"<class>.<memory>" of this item

Value

String Value

Default

DataKey-1 => Image

Version

2.30 (redefined)

8. History

Version < 1.00

Historic version with multiple headers, before and after the binary data, not restricted to 512 byte blocks (before 1995)

Version 1.00

Version with a single header preceeding the binary data, usually restricted to 512 byte blocks (until February 1998)

Version 2.00

Totally renewed version as described in this document

Version 2.10

New keywords: Intensity0, Intensity0Monitor, Intensity1, Intensity1Monitor, AnodeCounts, AnodeCountsMonitor, ExposureTime, ExposureTimeMonitor, ESRF_ID01_TDC_ResolutionMode, ESRF_ID01_TDC_Resolution, ESRF_ID01_TDC_Offset, ESRF_ID01_TDC_Timeout,

ESRF_ID01_Goniometer_Theta, ESRF_ID01_Goniometer_Chi,
ESRF_ID01_Goniometer_Chi, ESRF_ID01_Goniometer_Phi

Version 2.11

History of command lines

Version 2.20

New keywords: DataValueOffset

Version 2.30

The only defined class is "Image", the only used instances are "Psd" and "Error". The block identifier has still the form <sequence>.<class>.<instance>[.<memory>].

9. Appendix

Example of a General Header

Words in italics are historical keywords, that are only written for backward compatibility. Underlined values are only written for backward compatibility and will be replaced by their alias names.

```
{
EDF_DataFormatVersion = 2.30 ;
EDF_DataBlocks = 9           ;
EDF_BlockBoundary = 512 ;




}
```

Example of a Data Block Header

Words in italics are historical keywords, that are only written for backward compatibility. Underlined values are only written for backward compatibility and will be replaced by their alias name.

{
EDF_DataBlockID = 1.Image.Psd ;
EDF_BinarySize = 1048576 ;
ByteOrder = LowByteFirst ;
Center_1 = 269 ;
Center_2 = 268 ;
Compression = None ;
DataType = FloatValue ;
DDummy = 0.1 ;
Dim_1 = 512 ;
Dim_2 = 512 ;
Dummy = -1 ;
HeaderID = EH:000001:000000:000000 ;
Image = 1 ;
Offset_1 = 0 ;
Offset_2 = 0 ;
Psize_1 = 0.000343 ;
Psize_2 = 0.000337 ;
RasterOrientation = 1 ;
SampleDistance = 9.82514 ;
SaxsDataVersion = 1.0 ;
Size = 1048576 ;
Title = vacuum setup ;
WaveLength = 9.90376e-11 ;

}

<binary data follows here>

Example of a Data Block Header for Raw Data

Words in italics are historical keywords, that are only written for backward compatibility. Underlined values are only written for backward compatibility and will be replaced by their alias name.

{

EDF_DataBlockID = 1.Image.Psd ;

EDF_BinarySize = 1048576 ;

ByteOrder = HighByteFirst ;

Center_1 = 269 ;

Center_2 = 268 ;

Compression = None ;

DataType = UnsignedInteger ;

DetectorName = two dimensional delay line detector (IF = 176, SN = 3) ;

DetectorPosition = 9.10013 ;

Dim_1 = 512 ;

Dim_2 = 512 ;

Dummy = -1 ;

HeaderID = EH:000001:000000:000000 ;

HMDeltaTime = 0.006 ;

HMFile = /scratch/data/com38/com38_001-001 ;

HMFrame = 1 ;

HMStartTime = Wed Dec 4 02:51:48 1996 ;

HS32C01 = 59807 ;

HS32C02 = 3562 ;

HS32C03 = 2228 ;

HS32C04 = 8743 ;

HS32C05 = 54374 ;

HS32C06 = 2150 ;

HS32C07 = 57630 ;

HS32C08 = 17714 ;

HS32C09 = 54374 ;

HS32C10 = 28 ;

HS32C11 = 0 ;

HS32C12 = 0 ;

HS32C13 = 726006 ;

HS32C14 = 355 ;

HS32C15 = 1.05002e+08 ;

HS32C16 = 51268 ;

HS32C17 = 0 ;

HS32C18 = 0 ;

HS32C19 = 0 ;

HS32C20 = 0 ;

HS32C21 = 0 ;

HS32C22 = 20095 ;

HS32C23 = 1.07518e+08 ;

HS32C24 = 1.07518e+08 ;

HS32C25 = 0 ;

HS32C26 = 0 ;

HS32C27 = 0 ;

HS32C28 = 0 ;

HS32C29 = 0 ;

HS32C30 = 0 ;

HS32C31 = 0 ;

HS32C32 = 0 ;

HS32Depth = 26.8074 ;

HS32F01 = 7.59e+07 ;

HS32F02 = 7.27e+07 ;

HS32F03 = 3.51228e+07 ;

HS32F04 = 4.0644e+08 ;

HS32F05 = 9.78206e+06 ;

HS32F06 = 1.05443e+11 ;

HS32F07 = 1.35616e+06 ;

HS32F08 = 3.56277e+11 ;

HS32F09 = 9.78206e+06 ;

HS32F10 = 2.00337e+10 ;

HS32F11 = 1 ;

HS32F12 = 1 ;

HS32F13 = 1 ;

HS32F14 = 2048 ;

HS32F15 = 1e-06 ;

HS32F16 = 0.002048 ;

HS32F17 = 1 ;

HS32F18 = 1 ;

HS32F19 = 1 ;

HS32F20 = 1 ;

HS32F21 = 1 ;

HS32F22 = 0.01 ;

HS32F23 = 1 ;

HS32F24 = 0.0005 ;

HS32F25 = 0 ;

HS32F26 = 0 ;

HS32F27 = 0 ;

HS32F28 = 0 ;

HS32F29 = 0 ;

HS32F30 = 0 ;

HS32F31 = 0 ;

HS32F32 = 0 ;

HS32Len = 32 ;

HS32N01 = PIN1 ;

HS32N02 = PIN2 ;

HS32N03 = PIN3 ;

HS32N04 = PIN41 ;

HS32N05 = PIN42 ;

HS32N06 = PIN5 ;

HS32N07 = PIN6 ;

HS32N08 = PIN7 ;

HS32N09 = I0 ;

HS32N10 = I02 ;

HS32N11 = I1 ;

HS32N12 = I12 ;

HS32N13 = anode ;

HS32N14 = anode2 ;

HS32N15 = time ;

HS32N16 = time2 ;

HS32N17 = bad ;

HS32N18 = bad ;

HS32N19 = bad ;

HS32N20 = bad ;

HS32N21 = bad ;

HS32N22 = LINKAM ;

HS32N23 = vfc27 ;

HS32N24 = thc2 ;

HS32N25 = 0 ;

HS32N26 = ;

HS32N27 = ;

HS32N28 = ;

HS32N29 = ;

HS32N30 = ;

HS32N31 = ;

HS32N32 = ;

HS32Z01 = 41.01 ;

HS32Z02 = 14.16 ;

HS32Z03 = 19.82 ;

HS32Z04 = 82.31 ;

HS32Z05 = 416 ;

HS32Z06 = 20.8 ;

HS32Z07 = 549.4 ;

HS32Z08 = 168.6 ;

HS32Z09 = 416 ;

HS32Z10 = 0.203125 ;

HS32Z11 = 0 ;

HS32Z12 = 0 ;

HS32Z13 = 0 ;

HS32Z14 = 0 ;

HS32Z15 = 0 ;

HS32Z16 = 0 ;

HS32Z17 = 0 ;

HS32Z18 = 0 ;

HS32Z19 = 0 ;

HS32Z20 = 0 ;

HS32Z21 = 0 ;

HS32Z22 = 20050 ;

HS32Z23 = 0 ;

HS32Z24 = -4000 ;

HS32Z25 = 0 ;

HS32Z26 = 0 ;

HS32Z27 = 0 ;

HS32Z28 = 0 ;

HS32Z29 = 0 ;

HS32Z30 = 0 ;

HS32Z31 = 0 ;

HS32Z32 = 0 ;

HSAnode = 13 ;

HSAnodeS = 14 ;

HSI0 = 5 ;

HSI0S = 10 ;

HSI1 = 7 ;

HSI1S = 0 ;

HSTime = 15 ;

HSTimeS = 16 ;

Image = 1 ;

MachineInfo = " Ie=165.58mA,gap46=25.54mm,taper46=

0.00mm,gap26=20.31mm,taper26= 0.01mm" ;

Offset_1 = 0 ;

Offset_2 = 0 ;

OpticsInfo = optics ;

Psize_1 = 0.000343 ;

Psize_2 = 0.000337 ;

RasterOrientation = 1 ;

SampleDistance = 9.82514 ;

SaxsDataVersion = 1.0 ;

Size = 1048576 ;

StationInfo = id2 ;

ExperimentInfo = detector with 2.02% R14 + 20.1% C2H6 + QS Xe(Air

Liquide ;

Title = vacuum setup ;

WaveLength = 9.90376e-11 ;

}

<binary data follows here>