Specification

for

XAS file structure

and **format**

prepared by

L.Chiappetti - IFCTR

issue 2.0 - 31 Aug 1992

DAWG-REP 20/92

The reasonable man adapts himself to the world
The unreasonable man persists in trying to adapt the world to himself
Therefore all progress depends upon the unreasonable man
(G.B.Shaw)

Table of content

Amendment history

The layout of the present document is in agreement with the rules in [Ref. 2], section 13.3.

10 Oct 1991	issue 1.0	draft DAWG-REP.15/91 (includes sections 1-3 and 6)
31 Aug 1992	issue 2.0	issued as DAWG-REP.20/92 (revised sections 1 and 6; issued sections 4 and 5)

Note that sections 2 and 3 are not reissued in this issue 2.0 : the corresponding sections in issue 1.0 form full part of the present issue.

Note also that section 5 includes a number of tables, whose numbering may show gaps : the numbering correspond strictly to the equivalent tables in section 3. If a table is missing in section 5 the corresponding table in section 3 is applicable.

1. introduction

The purpose of this note is to make a final proposal for the "XAS own" file structure, according to the guidelines presented by the author in the report [ref.1] at SAX DAWG meeting on May 13, 1991, and in the XAS document [ref.2], chapter 7, to which the reader is referred. The original proposal was advanced in draft 1.0 (Oct 91) of the present document (which is mainly reflected in sections 2 and 3), while now (Aug 92), after the tests referred to in section 4, we are in a position to amend it in a final proposal (section 5 of the present issue 2.0).

We remind here the essential guidelines :

a) file structures *non sunt multiplicanda praeter necessitatem* (**Ockham's razor**). Namely the following logical structures are identified:

- telemetry files
- photon lists
- images (including histograms as 1-d images)
- spectra
- time profiles
- response matrices
- generic tables
- generic auxiliary files

b) one wishes to maximize the **efficiency** of access (i/o speed, including conversion to machine internal representation, and disk space usage)

c) one wishes to have a "**natural**" representation of the data, determined by the most natural memory arrangement, and by the typical usage of the data (ie. bulk access vs record-by-record access).

d) a **tradeoff** between the previous two requirements must be sought

e) administrative (**header**) information must be carried along with the file, and be **flexible and extendabl**e

f) it shall be easy to export the data to other systems **using standard interchange formats**

2. plan for future work

2.1. scope of the present document

The following data structures (as listed under item a in section 1 above) are excluded from the present discussion:

∑ **telemetry file**s : the format of these is determined by the layout of the telemetry packets and will be discussed separately once such layout will be released by Alenia.

∑ **generic auxiliary file**s : these (see eg. [ref.2] 7.2.3.6 and 7.2.4, but also 7.2.5 for generic ASCII tables and 7.2.6 for ASCII tabulated calibration data) will be free-format ASCII files, whose format will be decided and documented in the implementation of the particular program producing or using them

2.2. physical types of data structures

For the remaining data structures one might consider three classes :

∑ **images and assimilated**: these are standard (bulk) binary image files

∑ **response matrices** : these could be implemented as a collection of two image-like files : a 2-d image containing the matrix proper, and a 1-d histogram containing the reference energy grid (when not equispaced).

∑ **all others** : these may be implemented as binary tables

All data structures shall share the same handling of header information.

2.3. a possible format

The future (short term) work will consist in a number of tests based on a prototype structure for images and binary tables. The tests will use limiting cases, which might be individuated as :

∑ **big** (of the order 1024∞1024 REAL*4) **response matrices** (which are a particular case of images, possibly among the biggest one in SAX case, as only WFC images may reach that size too) and

∑ **large light curve files** (which are a particular case of binary tables; spectra are not critical for what size is concerned).

The "new" layout is proposed in the next section. This includes the layout of the header area (although the parameter list may be incomplete at this stage), and the layout of the data area.

Such layout will be designed to be immediately mappable to FITS, the latest definition of which is here assumed to be contained in [Ref.4] (and [Ref.5] for the binary table extension). Note that my current feeling is that FITS is a transport format, not a storage format used by the data analysis system (this is consistent with the traditional view e.g. in IRAF, MIDAS etc.)

The purpose of the test will be to assess whether such dedicated format can be accessed with sufficient efficiency using standard Fortran calls, or whether special routines, encapsulating more efficient system calls have to be used. At the same time the i/o speed will be compared with a direct FITS implementation of the layout.

One of the result of the tests will be the difference in speed between the access to a dedicated format (which will employ machine-dependent internal representation) and the access to a pure FITS format (which necessarily requires a conversion). This way it will also be gained some experience with the FITSIO software [Ref.3]

It is not *a priori* excluded that these elements could also lead to the conclusion that the FITS format itself (though not completely natural) it is suitable to our purposes (on the ground of the saving in programming effort due to reuse of external software).

The usage of disk space (in particular the amount of overheads) will also be assessed at the same time for both cases.

3. detailed proposal

Each file will consist logically of an **header area**, and a **data area**.

The proposed implementation uses fixed record length binary files, to be accessed using Fortran (unformatted) direct access. The record length will be different from filetype to filetype, and even from file to file (this is the *naturality requirement* by which e.g. an image is naturally stored with each row in a record, etc.).

The **data area** will have a predermined fixed size (ie. be not extendable). The size will be determined *while creating* the file (not necessarily *before* creation: the header may be modified accordingly as *last* step of the creation), and any extension or truncation will *not* occur in place, but involve the creation of a new file.

On the contrary the **header area** will be extendable (e.g. to append history information). This may give rise to difficulties in the case a FITS format is selected and one wants in place extensions (without creation of new file), since the header is at the front of the file.

The requirement for extendability, and also the fact that *Unix systems do not store the information on record length anywhere* in the file directories (ie. the record length is not associated univocally to a file) lead to the fact that the header area is physically split in two parts :

∑ a **mini-header** in the first record of the file
∑ the rest of the header (**full header**) at the end of the file

The order of the information will therefore be :

```
1 record :     mini-header (unused part of the record padded with
binary zeros)
n records:     data area (by definition this will have 100% filling
efficiency)
m records:     full header area (unused part of last record padded
with binary zeros)
```

Such order is clearly depicted in the following figure :

In case of a FITS implementation of course one will have only one header (*primary header*) area and one data area (*primary data array*) (in normal case, more *HDUs* will follow if a conforming *image extension* is used - this is currently planned for response matrices only). Refer to {Ref.4] section 4 for details.

3.1. Common header format

3.1.1. mini-header content

The mini-header will always occupy the first 28 bytes of the first record in the file. This implies that all data and header records cannot be shorter than 28 bytes (if they need to be, they shall be padded with binary zeros up to such length, but this should occur quite unfrequently).

The remainder of the first record shall be padded with binary zeros.

The mini-header has a special format designed for quick access (for instance in Unix by dedicated opening routines which need to derive teh record length to reopen the file as direct access - in VMS an `INQUIRE` statement could be used TBV to derive the record length directly) and includes the fields in table 3.1:

Table 3.1 : layout of mini-header

Bytes	Type	Field	Comments
01 to 16	Char	MAGIC	magic number (for use by Unix systems)
17 to 20	I*4	RECLEN	record length of the file in bytes
21 to 24	I*4	DATASIZE	number of records in data area
25 to 28	I*4	HDRSIZE	number of records in full header area

The *magic number* will "univocally" identify a XAS data file. On Unix systems it could be used by the `file` command to tell what kind a file is (this implies adding the definition to the proper file, eg. `/etc/magic` on SunOS, `/usr/lib/file/magic` on Ultrix etc. Of course the same information could also be used by the file opening routines in the scientific applications to check whether a file is of the correct type.

The magic number shall be a bit pattern sufficiently unfrequent to be generated by chance. It is proposed to use a mixture of ASCII and binary patterns for this purpose. ASCII will ensure some legibility, while the binary part will ensure it cannot be generated spuriously (e.g. with an editor). It is suggested to have a hierarchical arrangement of 16 bytes into 4 4-byte fields. Each 4-byte field will consist of 3 printable characters and a binary value. The following layout is proposed :

$XAS_1 PPP_2 TTT_3 VVV_4$

where the subscript notation indicates an octal value (i.e. $_1$ is octal 001 etc.).

The first field will always be the string 'XAS' followed by octal 001;

the second field *PPP* (followed by octal 002) will specify the physical arrangement (i.e. IMG for images and BIN for binary tables).

the third field *TTT* (followed by octal 003) will specify the data types, as listed in section 1, item a (and will be specified below for each data type)

the fourth field VVV (followed by octal 004) is reserved for version identification, and is currently assumed to contain three blanks (coded below as 'ΔΔΔ').

3.1.2. full header content

The full header includes a variable number of records, with length `RECLEN`. This number is given by the mini-header parameter `HDRSIZE`.

The header is composed by **keywords**. Each keyword is composed *logically* by a *name-value couple*. The number of keywords in a header is not predetermined. Keywords follow each other in the header without separators. Keywords may span record boundaries.

This way all records but the last are 100% full. The last record could be only partially full and *shall be padded with binary zeros*.

Each keyword has a **name**. The name shall be an 8-character ASCII string, according to FITS standards [Ref.4 5.1.2.1] This will make easier the mapping to FITS files.

Each keyword has a **data type**. Recognised data types are :

- Character
- INTEGER*2 (I*2, short int, 16-bit integer)
- INTEGER*4 (I*4, long int, 32-bit integer)
- REAL*4 (R*4, float, 32-bit floating point)
- REAL*8 (R*8, double, 64-bit double precision floating point)

It might be wise to avoid usage of REAL*8 variables, as their full precision cannot be mapped in the FITS representation [Ref.4 5.3.2.4]. On the other hand double precision *is necessary* for quantities like angles (at the level of fraction of arcseconds) and perhaps time (e.g. Julian days). This may lead to the following line of reasoning.

It could be possible to introduce further specific data types. These will be *physically* using one of the above formats, but be flagged as special (for instance for special formatting when displayed *or* when *written as character strings* to a FITS header). One might consider data types like:

- Angle (a R*8 value in decimal degrees, to be displayed as `ddd:mm:ss.ff`)
- Date (a TBD value, to be displayed as `yyyy:mm:ss`)
- Time (a TBD value, to be displayed as `hhh:mm:ss`; the usage of other data types to indicate times with a finer resolution is TBD; it is also TBD to define a convention whether a special data type is needed for hour angle coordinates, like *right ascension*, which are naturally stored in decimal degrees, but displayed in time format)

Each keyword has a **value**, which is encoded in the machine-specific binary internal representation appropriate for the data type. The *length* of the value field is *implicitly determined* by the datatype, except for character values.

The **length** of character values may be in line of principle any value between 1 and 255 (I suggest to limit it in practice to *even* values between 2 and 254 bytes for alignment reasons, and to pad it with blanks if odd), and is encoded in the keyword representation.

The format of a keyword is specified below in table 3.2

The proposed way of storage is much more compact than the FITS format, which always requires 80 bytes for each keyword [Ref.4 5.1.1], while here 12 to 264 bytes are used. It might be wise to confine the length of character values in normal cases between 8 and 68 bytes, so they can fit within a FITS card image (columns 12 to 79, see [Ref.4 5.3.2.1]) (otherwise if longer strings are really needed a mechanism should be devised, e.g. mapping them to multi-line `COMMENTS` ?)

The **end of the header** will be *implicitly* signaled either by:

- no more header records beyond `HDRSIZE`
- a character keyword with zero length (automatically ensured by binary-zero-padding)

The presence and order of keywords in the header is specified in the detailed description by data type below. In line of principle the order shall be :

- *mandatory* keywords (these might include mandatory FITS keywords [Ref.4 5.2.1], which shall appear before XAS specific keywords, and have the same name and order as the corresponding FITS keyword)
- *generic* or *data type specific* keywords (in case a FITS reserved keyword [Ref.4 5.2.2] is available for a given function this shall be used)
- *experiment specific* keywords
- *free* keywords

In the descriptions below (data type specific tables, from table 3.3 onwards) the keyword will be described as follow:

The *name* and *datatype* is presented in columns 1 and 2 of the table.

Column 3 indicates whether the keyword is present in the FITS primary header (`YES`), or in an extension header (`EXT`) or some value derived from it is present in the FITS header (`Deriv`), or if it is not present at all (`NO`).

Column 4 indicates whether the keyword is present in the mini-header (`Mini`), in the full header (`Full`) or if it is a FITS keyword not present in our header as it can be derived implicitly (`NO`)

Column 5 classifies the keyword as mandatory (`Mand`), implicitly defined (`Impl`, this refers to FITS keywords not present in our headers), generic for all files or for a given data type (`Gener`), instrument-specific (`Instr`) or optional (`Opt`). The software shall be designed to be able to read files with just mandatory keywords for processing/display purposes assuming suitable defaults for other keywords when relevant; however all mandatory, generic and instrument keywords shall be present in files produced in XAS, and used for administrative purposes.

Column 6 describes the relevant keyword.

A set of routines to access keywords will need to be developed (using Fortran `CHARACTER` handling). It would be easy to read a keyword content, write a new keyword, change the content of a non-character keyword. It might create problems to change the content of a character keyword if this implies altering (particularly extending) its length, as well as deleting a keyword or inserting a keyword at a specific place (this shall never be necessary), as all such operations imply rearrangement of the header records.

Table 3.2 :format of header keywords

Bytes	Type	Field	Comments
01 02 03 to 10 01 to x	Byte Byte Char*8 var.	Type Length Name Value	Data type (see below) Data length (see below) Keyword name, left-justified and padded with blanks Value (see below)

For I*2 data type

Bytes	Type	Field	Comments
01	Byte	Type=I*2	1
02	Byte	Length	2 (redundant)
11 to 12		Value	INTEGER*2 value

For I*4 data type

Bytes	Type	Field	Comments
01	Byte	Type=I*4	2
02	Byte	Length	4 (redundant)
11 to 14		Value	INTEGER*4 value

For R*4 data type

Bytes	Type	Field	Comments
01	Byte	Type=R*4	3
02	Byte	Length	4 (redundant)
11 to 14		Value	REAL*4 value

For R*8 data type

Bytes	Type	Field	Comments
01	Byte	Type=R*8	4
02	Byte	Length	8 (redundant)
11 to 18		Value	REAL*8 value

For character data type

Bytes	Type	Field	Comments
01	Byte	Type=char	0
02	Byte	Length	m (shall be an even number, minimum 2, maximum 254)
11 to p (p=10+m)	Char*m	Value	Character string (left justified, blank padded)

Table 3.2(cont.)

For angle data type

Bytes	Type	Field	Comments
01	Byte	Type=angle	5
02	Byte	Length	8 (redundant, implicit)
11 to 18		Value	REAL*8 value in decimal degrees

For date data type

Bytes	Type	Field	Comments
			TBW

For time data type

Bytes	Type	Field	Comments
			TBW
Any other			TBW

3.2. images

The image format will be used for any 2-d data suitable to be represented as a function *z=f(x,y)*, irrespective whether x and y are true spatial coordinates (detector and sky images) or any other variable (e.g. energy and risetime, or photon spectral index and hydrogen column density, etc.). In any case x and y shall be on an equispaced grid (this is always possible if the units of x and y are just pixels !).

The same format will be used for 1-d histograms (*z=f(x)* with x equispaced) as a degenerate case.

3.2.1. header content

The image header content is described in Table 3.3.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

if BITPIX=16 MAGIC is $XAS_1IMG_2INT_3\Delta\Delta\Delta_4$
if BITPIX=-32 MAGIC is $XAS_1IMG_2FLO_3\Delta\Delta\Delta_4$
if BITPIX=16 RECLEN is NAXIS1*2
if BITPIX=-32 RECLEN is NAXIS1*4
always DATASIZE is NAXIS2

3.2.2. data field content

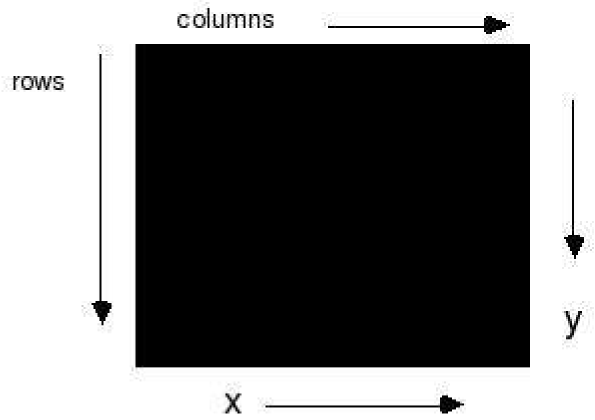
The data field content will be as follows :

there are *n* records (*n*=DATASIZE), where *n* is the number of rows in the image (that is the number of pixels in the y direction, *n*=NAXIS2).

The record length RECLEN is a function of the number *m* of pixels in the x direction, *m*=NAXIS1, and is equal to *m**abs(BITPIX)/8 bytes.

It is TBD whether BITPIX shall always be -32 (REAL*4 data, RECLEN=*m**4 bytes), or whether we should allow either -32 (REAL*4 data) or 16 (INTEGER*2 data); no others cases shall be allowed.
Integers are enough in most cases of true X-ray photon images, but are definitely inconvenient (because of annoying scaling, and loss of precision) in most other cases (e.g. χ^2 maps, WFC sky images). It has to be verified whether the factor 2 saving in disk space and memory is worthwhile the complication of i/o routines capable of handling both cases.
In all cases BITPIX=-32 will indicate *native-format* floating point data *only* as well as BITPIX=16 will indicate native-format integers (any conversion to IEEE format or byte swap will be taken care during FITS conversion).

Data will be stored in the file as a sequence of records, one record for each image row. In each record there are as many pixels as columns in the file. Therefore, if the elements in a record are indexed with subscript i, i will go as the x-coordinate. The record number j will go as the y-coordinate (it is in line of principle possible to store in reversed mode, the highest y-value corresponding to j=1 and the lowest to j=n_rows, and this was actually used in the past to store images "on disk as they appear on the screen"; but this is felt an unnecessary complication). The *natural storage* will instead store an image array A (I,J) as ((A(I,J), I=1,n_columns), J=1,n_rows), where both I and J go with x and y.



The natural storage will facilitate any debugging, and also (in the rare cases when it is needed) the access to specific regions in the image.

3.2.3. convention for pixel coordinates

This section to be finalized later; the details in here do not have any impact on the tests of i/o efficiency etc. and impact only on the choice of header keywords.

One convention TBD concerns which point (centre, lower-left corner, upper-left corner etc.) of the *finite* pixel size has to be associated with the pixel coordinates.

One other convention concerns the transformation of pixel coordinates to physical units. This is :

1) non-existant for images in raw pixels, or channels (e.g. pseudo-images in PHA-channels vs RT-channels).

2) trivial (a simple linear relation) in a majority of cases (e.g. conversion of detector pixels in raw angular coordinates, i.e. arcseconds with arbitrary orientation; or chi-square grids with photon index or temperature or alike on the xy axes). In such cases the `CRPIXn`, `CRVALn`, `CRDELTn` keywords will easily describe the transformation.

3) slightly more complicated if a different relation (logarithmic, inverse) is wished. This might be the case e.g. of chi-square grids with NH on one axis. A TBD mechanism (an extra keyword ?) has to be devised.

4) quite specific if a conversion from detector pixels to celestial coordinates is wished. This shall take into account the orientation of the instrument (and possibly the instrument misalignments too). It is TBD whether this is described by a set of dedicated keywords, or it is implemented within the FITS `CROTAn` keywords.

5) more complex cases occurs if the data are not at allequispaced (in which case the image is more properly a table, ans as such shall be treated), or if they are nearly equispaced (divided into several subsets, each of which is equispaced). More than using a group or tabular structure, it could be convenient to preserve the image representation in such cases. So far this appears to apply only to response matrices, which are described separately in section 3.3 below.

Table 3.3 : image header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1IMG_2INT_3\Delta\Delta\Delta_4$ or $XAS_1IMG_2FLO_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1*2 or NAXIS1*4
DATASIZE	I*4	Deriv	Mini	Mand	equal to NAXIS2
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	16 or -32 see text
NAXIS	I*2	YES	NO	Impl	always 2 implicitly or 1 for histo and 2 for images ?
NAXIS1	I*2	YES	Full	Mand	no of pixels in x
NAXIS2	I*2	YES	Full	Mand	no of pixels in y (1 for histograms)
					Generic information
DATE	Date	YES	Full	Gener	date file written
ORIGIN	C*4	YES	NO	Impl	set implicitly to string 'XAS ' ?
DATE-OBS	Date	YES	Full	Gener	date of observation
FILENAME	C*39	YES	Full	Mand	name of image file (without pathname and filetype extension)
SATELLIT	C*4	YES	Full	Gener	satellite name (for our purposes coded as string 'SAX ') (replaces,maps or duplicates FITS TELESCOP keyword).
INSTRUME	C*4	YES	Full	Gener	instrument code (see Table 3.4 for SAX instruments)
OBSERVER	C*16	YES	Full	Opt	Observation PI name
OBJECT	C*16	YES	Full	Gener	target name
EQUINOX	I*2 ?	YES	Full	Opt?	code as 2000 ? FITS wants it as R*4

Table 3.3(cont.)

					image specific info
BUNIT	Char	YES	Full	Gener	Units for z=f(x,y) (see Table 3.5)
CTYPE1	Char	YES	Full	Gener	units on x-axis (df 'PIXELS' see Table 3.5)
CTYPE2	Char	YES	Full	Gener	units on y-axis (df 'PIXELS' see Table 3.5)
CRPIX1					TBD
CRPIX2					TBD
CRVAL1					TBD
CRVAL2					TBD
CDELT1					TBD
CDELT2					TBD
CROTAn					or other mechanism TBD to describe orientation
DATAMAX	R*4	YES	Full	Opt?	maximum value
DATAMIN	R*4	YES	Full	Opt?	minimum value
					instrument specific info
					TBW
					free keywords
COMMENT	Char	YES	Full	Opt	any comment (optional)
HISTORY	Char	YES	Full	Mand	sequence of commands which created the file
END		YES	NO	Impl	end of FITS file (end of header is implicit)

Table 3.4 : Instrument codes

SATELLIT keyword	INSTRUME keyword	instrument name	comments
'SAX '	'MECS'	ME Concentrator	what about units ?
	'LECS'	LE Concentrator	
	'PDS '	Phoswich	
	'HPGS'	HP GSPC	
	'WFC1'	Wide Field	
	'WFC2'	Cameras	
other		TBD	TBW

Table 3.5 : unit codes for images

Keyword	Value	Comment
BUNIT	COUNTS	Normal images (detector, sky and pseudo-images)
	COUNTS/S	Images divided by exposure time (not recommended)
	CHISQUARE	Chi-square grids
CTYPEn	PIXELS	Normal images (detector images)
	TBD angular coordinates (eg ARCSEC)	Normal images (detector and sky images, to be discussed)
	ENERGY	Pseudo-images (PDS)
	RISETIME	Pseudo-images (PDS)
	PHOTON_INDEX	Chi-square grids
	N_H	Chi-square grids
	other TBD	Chi-square grids
	other TBD	other TBD cases

TBV with FITS convention whether a keyword can have imbedded blanks (e.g. 'PHOTON INDEX'). If not use underscore to replace blanks.

3.3. matrices

A response matrix will be stored as one REAL*4 *image file*, *and* an *associated* 1-d REAL*4 *histogram file* containing the reference energies.
These are separate files, of which the histogram file is pointed at by a keyword in the main file header (the naming convention is TBD). In case of
FITS format the two logical files will be encapsulated in a single FITS file, with the matrix being the *primary array* and the histogram being an *image extension*.

The reference histogram is useful whenever the reference energies are not equispaced (this might be often the case to describe accurately the response close to an edge). In the case energies are equispaced one might consider not having the file, and just using other keywords in the main header to compute the energies.

3.3.1. header content

The image header content is described in Table 3.6 for the matrix image file proper and in Table 3.7 for the associated histogram.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

for matrix image `MAGIC` is $\mathtt{XAS}_1\mathtt{IMG}_2\mathtt{MAT}_3\Delta\Delta\Delta_4$
for histogram `MAGIC` is $\mathtt{XAS}_1\mathtt{IMG}_2\mathtt{FLO}_3\Delta\Delta\Delta_4$
for matrix image `RECLEN` is `NAXIS1*4`
for histogram `RECLEN` is `NAXIS1*4`
for matrix image `DATASIZE` is `NAXIS2`
for histogram `DATASIZE` is 1 (=`NAXIS2`)
etc. TBD due to the fact histogram is an image extension

3.3.2. data field content

The content of the data field of the *matrix file proper* will be a REAL*4 image, with x-axis in adimensional pixels, and y-axis in PHA channels (also adimensional numbers). The matrix will preferably hold the response function already multiplied by the effective area and the energy grid bin width (dimensions of cm^2 kev), such that the convolution is a simple addition of terms (over the energy axis) obtained multiplying a matrix element times an input spectrum computed at the reference energy.

The conversion from x pixels to energy (keV) will be obtained thru the corresponding bin in the data field of the *associated histogram*, which shall contain the start (TBD) energy in keV. If the energy grid is equispaced it is TBD to dispense with the associated file and use instead header keywords to compute the energy of each column of the matrix.

3.3.3. Coordinate and naming conventions

Response matrices shall be used only with a compatible spectrum, i.e. they shall have been rebinned (if at all) in the same way over the PHA axis.

For the distinction between raw PHA channels and PHA bins see the section on spectra (3.5.3 below).

The information on PHA channel (or bin) boundaries need not to be stored in the matrix as it is never used alone, but only *when convolving* a spectrum *with* a matrix.

On the other hand the information on the binning of reference energies pertains to the matrix alone and not to the spectrum. It is of course available when convolving, but also for pure display purposes (for debugging or tutorial usage).

Table 3.6 : response matrix header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is XAS$_1$IMG$_2$MAT$_3$ΔΔΔ$_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1*4
DATASIZE	I*4	Deriv	Mini	Mand	equal to NAXIS2
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	-32
NAXIS	I*2	YES	NO	Impl	2
NAXIS1	I*2	YES	Full	Mand	no of energies
NAXIS2	I*2	YES	Full	Mand	no of PHA channels
EXTEND	L	YES	NO	Impl	always T (implicit)
					Generic information
DATE	Date	YES	Full	Gener	date file written
ORIGIN	C*4	YES	NO	Impl	set implicitly to string 'XAS ' ?
DATE-OBS	Date	YES	Full	Opt	date of observation (could be useful if response changes with time)
FILENAME	C*39	YES	Full	Mand	name of matrix file (without pathname and filetype extension)
SATELLIT	C*4	YES	Full	Gener	satellite name (for our purposes coded as string 'SAX ') (replaces,maps or duplicates FITS TELESCOP keyword).
INSTRUME	C*4	YES	Full	Gener	instrument code (see Table 3.4 for SAX instruments)
OBSERVER	C*16	YES	Full	Opt	Observation PI name
OBJECT	C*16	YES	Full	Opt	target name

Table 3.6(cont.)

					matrix specific info
BUNIT	Char	YES	Full	Gener	'CM2*KEV'
CTYPE1	Char	YES	Full	Gener	'ENERGY'
CTYPE2	Char	YES	Full	Gener	'PHA CHANNELS'or 'BINS'
CRPIX1					1
CRPIX2					1
CRVAL1					start energy in keV
CRVAL2					start PHA channel or bin (see CTYPE2)
CDELT1					energy step (keV) for equispaced energy grids or 0 for non-equispaced
CDELT2					1 if scale is in PHA channels; n if bins are rebinned from PHA channels with binning factor n; 0 otherwise
DATAMAX	R*4	YES	Full	Opt?	maximum value
DATAMIN	R*4	YES	Full	Opt?	minimum value
REFHISTO	C*39	YES	Full	Mand?	name of associated histogram file (if energy grid not equispaced)
ENDENERG	R*4	YES	Full	???	end energy in keV ?
ENDCHAN	R*4	YES	Full	???	last PHA channel (useful if non-equispaced rebinning used)
TBD					mechanism to describe rebinning in PHA
					instrument specific info
					TBW
					free keywords
COMMENT	Char	YES	Full	Opt	any comment (optional)
HISTORY	Char	YES	Full	Mand	sequence of commands which created the file
END		YES	NO	Impl	end of FITS file (end of header is implicit)

Table 3.7 : response matrix associated histogram header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1IMG_2FLO_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1*4
DATASIZE	I*4	Deriv	Mini	Mand	1 (equal to NAXIS2)
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
XTENSION	Char	Ext	NO	Impl	IMAGE ? (implicit)
BITPIX	I*2	Ext	Full	Mand	-32
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	no of energies
NAXIS2	I*2	Ext	Full	Mand	1
PCOUNT	I*2	Ext	NO	Impl	0
GCOUNT	I*2	Ext	NO	Impl	0
all remaining information is optional and should be identical to corresponding fields in main response matrix unless otherwise specified					
					Generic information
DATE	Date	EXT	Full	Gener	date file written
ORIGIN	C*4	EXT	NO	Impl	string 'XAS ' ?
FILENAME	C*39	EXT	Full	Mand	name of histogram file (without pathname and filetype extension)
SATELLIT	C*4	EXT	Full	Gener	satellite name
INSTRUME	C*4	EXT	Full	Gener	instrument code
					matrix specific info
BUNIT	Char	EXT	Full	Gener	'KEV' (specific)
CTYPE1	Char	EXT	Full	Gener	'ENERGY BINS'
CRPIX1					1
CRVAL1					start energy in keV
CDELT1					1
DATAMAX	R*4	EXT	Full	Opt?	end energy
DATAMIN	R*4	EXT	Full	Opt?	start energy
ASSOCMAT	C*39	EXT	Full	Mand?	name of associated matrix file
					free keywords
HISTORY	Char	EXT	Full	Mand	sequence of commands which created the file
END		EXT	NO	Impl	end of FITS file (end of header is implicit)

3.4. tables

The format described here applies to any generic binary table (they could be used to store the output of specific analysis programs). The other data types (spectra, time profiles, photon lists) described below are a subset of this general case.

3.4.1. header content

The generic table header content is described in Table 3.8. Note that in the case the table is written into FITS, it has to be handled as a *generalized extension*, therefore having a *primary header*, no primary data array, an *extension header* and the *table data array*. It is assumed that the primary header contains only the mandatory keywords, while all other keywords are in the extension header. It is however TBD whether "descriptive" keywords (like date, observer name, filename etc.) are instead to be put in the primary header. The distinction is irrelevant for "our" full header.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

for generic tables	`MAGIC` is	$XAS_1BIN_2GEN_3\Delta\Delta\Delta_4$
if stored by row	`RECLEN` is	`NAXIS1` (as defined in extension header)
if stored by row	`DATASIZE` is	`NAXIS2`

3.4.2. data field content

The main question about a generic table is that the preferred or optimal storage (*by rows* or *by columns*) depends on the typical usage of the data.
It is felt that the majority of the tables described below (e.g. time profiles and photon lists) are preferably accessed by row, and since the storage by row is also the natural one in FITS, it appears consequential to use this. However it could be possible to define an header keyword `STORAGE` to assume the values '`BYROW`' or '`BYCOLUMN`' to support an alternate storage (e.g. for spectra ?).

A *natural* implementation of the (default) `STORAGE='BYROW'` (which means in a table T(column,row) one has a sequence T(1,1), T(2,1) ... T(n_column,1),T(1,2) ...) is to have each record equal to a row. There might be cases when this is not altogether efficient (due to the small record length). This will be shown by appropriate tests. In such case one could define a record length multiple of the row datasize, and use an header keyword `PACKING` to define the blocking factor.

In the case `STORAGE='BYCOLUMN'` the choice of the record length might be complicated in case of very long tables, and could not be equal to the column datasize (and also, even if each column has the same number of rows, the datasize is in line of principle different, as each column has a specific data type with different byte length). This is a further reason to defer the definition of this mode (if at all to be implemented).

Table 3.8 : generic table header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1BIN_2GEN_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1
DATASIZE	I*4	Deriv	Mini	Mand	equal to NAXIS2
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	any value ? (use 8 ?)
NAXIS	I*2	YES	NO	Impl	0
EXTEND	L	YES	NO	Impl	always T (implicit)
END		YES	NO	Impl	this ends FITS primary header. All other parameters follow in extension header (or is it best to put the descriptive ones here ?)
XTENSION	Char	Ext	NO	Impl	'BINTABLE'
BITPIX	I*2	Ext	NO	Impl	8
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	size of a row in bytes
NAXIS2	I*2	Ext	Full	Mand	number of rows in table
PCOUNT	I*2	Ext	NO	Mand	0
GCOUNT	I*2	Ext	NO	Mand	0
TFIELDS	I*2	Ext	Full	Mand	number of columns
TFORMnnn	I*2	Ext	Full	Mand	formats of the columns in FITS binary table standard
EXTNAME	Char	Ext	NO	Impl	'GENERIC'
					Generic information
DATE	Date	YES ?	Full	Gener	date file written
ORIGIN	C*4	YES ?	NO	Impl	set implicitly to string 'XAS ' ?
FILENAME	C*39	YES ?	Full	Mand	name of image file (without pathname and filetype extension)

Table 3.8(cont.)

					table specific info
TTYPEnnn	Char	Ext	Full	Gener	label of column nnn
TUNITnnn	Char	Ext	Full	Gener	units of column nnn
TDISPnnn	Char	Ext	Full	Opt	display format (Fortran 90) for column nnn
					instrument specific info
					TBW
					free keywords
COMMENT	Char	Ext	Full	Opt	any comment (optional)
HISTORY	Char	Ext ?	Full	Mand	sequence of commands which created the file
END		Ext	NO	Impl	end of FITS file (end of header is implicit)

3.5. spectra

A *single spectrum* is easily stored in a small table (and there is a case to store it by column, as it will be generally accessed in its entirety). A possibility to store a *sequence of spectra* in a single file is presented as alternate, but is not recommended (it is simpler to have programs handling lists of single-spectrum files).

It is requested to take a decision in favour of one of the two proposed formats. It is not intended to give support to both formats at the same time.

3.5.1. header content

The spectrum header content is described in Tables 3.9 and 3.9a for the primary (recommended) and alternate format (see 3.5.2). Note that in the case the table is written into FITS, it has to be handled as a *generalized extension*, therefore refer to section 3.4.1 above for the location of keywords.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

for spectral files	`MAGIC` is	$XAS_1BIN_2SPE_3\Delta\Delta\Delta_4$
in primary format	`RECLEN` is	`nchannels*4`
in primary format	`DATASIZE` is	4
in alternate format	`RECLEN` is	`2*(nchannels)*4 + 16`
in alternate format	`DATASIZE` is	number of spectra + 1

3.5.2. data field content

A spectrum is here intended primarily as a count spectrum, see however Table 3.10 for other possible units. See 3.5.3 below for a discussion on the energy or channel axis.

The primary format proposed stores a *single* spectrum in a file. The spectrum is *naturally* defined as a table with 4 columns and *n* rows (with *n* equal to the number of PHA channels or bins). Since the number of rows is known in advance and limited to a relatively small size, and an entire spectrum will always be accessed together, it is proposed to store each column as a single record.

The 4 columns correspond to *data* (cts/s), associated *errors*, and to the *start* and *end boundaries* of each channel/bin.

An alternate format allows storage of more spectra in a file. Each spectrum corresponds to a row. The number of rows in the file is equal to the *number of spectra plus one*. One row (first or last) is dedicated to store the *channel boundaries* which are equal for all spectra.

The number of columns is at least 6, of which the first 2 exploit the "3d" capability of binary tables, ie. are arrays, namely :

one *n* channel spectrum (array of n floating point data values)
one *n* element array of the associated (floating point) errors

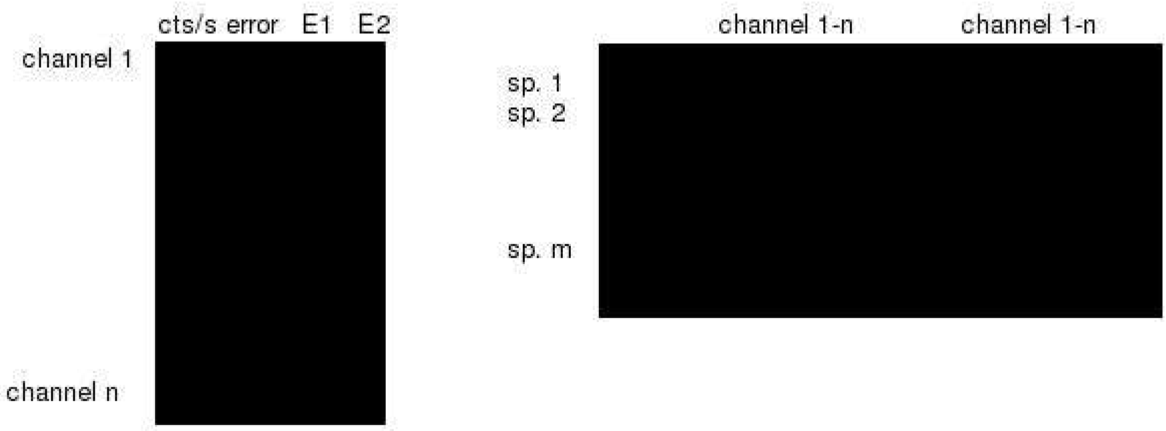
The remaining columns contain administrative information which is variable spectrum to spectrum : a minimum set includes :

spectrum start time
spectrum end time (or duration)
spectrum actual exposure time
spectrum dead time (percentage or absolute)

For the special row containing channel boundaries the interpretation of the 2 array column is different (e.g. start and end channel boundaries), while the content of the other columns is undefined.

This alternate format is proposed as an example, but looks slightly contrived and innatural, and it is suggested not to implement it and tu use a different mechanism (list of files) to act on sequences of spectra.

The two formats are schematically illustrated in the figure here below.



3.5.3. convention for energy channels

In case of X-ray spectra the x-axis is generally expressed in PHA channels or bins. Due to intrinsic non-linearity in the detectors, they seldom correspond to equispaced energies, therefore it is proposed (as usual) to store the lower and upper channel boundaries in the file.

It is also suggested (this applies to response matrices too) to reserve the name of *PHA channels* to the original instrument ADC channels (ranging from 1 to n in steps of 1), and the generic name of *bins* to any further grouping (rebinning). Grouping can be a plain rebinning (1 to n in steps of m), possibly with exclusion of first/last channels (p to q in steps of m; $1 \leq p$; $q \leq n$). In such cases it is useful to store in the header keyword like the start, end and step in PHA channels.

However there are cases of unequal rebinning (i.e. group all channels from p to q until the signal exceeds 3 σ), or even of rebinning with gaps (reject some channels for what fitting is concerned). In this case the binning can be defined only with reference to some mask (which could be stored in an external file, and reference in the header).

Table 3.9 : spectrum header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1BIN_2SPE_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS2*4 ?
DATASIZE	I*4	Deriv	Mini	Mand	4
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	any value ? (use 8 ?)
NAXIS	I*2	YES	NO	Impl	0
EXTEND	L	YES	NO	Impl	always T (implicit)
END		YES	NO	Impl	this ends FITS primary header.
XTENSION	Char	Ext	NO	Impl	'BINTABLE'
BITPIX	I*2	Ext	NO	Impl	8
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	16
NAXIS2	I*2	Ext	Full	Mand	number of PHA channels or bins (rows in table)
PCOUNT	I*2	Ext	NO	Mand	0
GCOUNT	I*2	Ext	NO	Mand	0
TFIELDS	I*2	Ext	Full	Mand	4
TFORMnnn (nnn=1 to 4)	Char	Ext	Full	Mand	'1E'
EXTNAME	Char	Ext	NO	Impl	'SPECTRUM'
					Generic information
DATE	Date	YES ?	Full	Gener	date file written
ORIGIN	C*4	YES ?	NO	Impl	set implicitly to string 'XAS ' ?
FILENAME	C*39	YES ?	Full	Mand	name of spectrum file (without pathname and filetype extension)

Table 3.9(cont.)

					spectrum specific info
STORAGE	Char	NO	Full	Gener	'BYCOLUMN'
TTYPE1	Char	Ext	Full	Gener	'DATA'
TUNIT1	Char	Ext	Full	Gener	'CTS/S' or other possible unit (see Table 3.10)
TDISP1	Char	Ext	Full	Opt	'G12.4'
TTYPE2	Char	Ext	Full	Gener	'ERRORS'
TUNIT2	Char	Ext	Full	Gener	same as TUNIT1
TDISP2	Char	Ext	Full	Opt	same as TDISP1
TTYPE3	Char	Ext	Full	Gener	'LOWER_BOUNDARY'
TUNIT3	Char	Ext	Full	Gener	'KEV'
TDISP3	Char	Ext	Full	Opt	'F8.4'
TTYPE4	Char	Ext	Full	Gener	'UPPER_BOUNDARY'
TUNIT4	Char	Ext	Full	Gener	same as TUNIT3
TDISP4	Char	Ext	Full	Opt	same as TDISP3'
					instrument specific info
					TBW
					free keywords
COMMENT	Char	Ext	Full	Opt	any comment (optional)
HISTORY	Char	Ext ?	Full	Mand	sequence of commands which created the file
END		Ext	NO	Impl	end of FITS file (end of header is implicit)

Table 3.9a : spectrum header (alternate format)

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1BIN_2SPE_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1
DATASIZE	I*4	Deriv	Mini	Mand	no. of spectra + 1
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	any value ? (use 8 ?)
NAXIS	I*2	YES	NO	Impl	0
EXTEND	L	YES	NO	Impl	always T (implicit)
END		YES	NO	Impl	this ends FITS primary header.
XTENSION	Char	Ext	NO	Impl	'BINTABLE'
BITPIX	I*2	Ext	NO	Impl	8
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	2*(number of channels) *4 + 16
NAXIS2	I*2	Ext	Full	Mand	number of spectra + 1
PCOUNT	I*2	Ext	NO	Mand	0
GCOUNT	I*2	Ext	NO	Mand	0
TFIELDS	I*2	Ext	Full	Mand	6
TFORMnnn (nnn=1 to 2)	Char	Ext	Full	Mand	'*n*E' with *n* = number of channels
TFORM3	Char	Ext	Full	Mand	TBD
TFORM4	Char	Ext	Full	Mand	TBD
TFORM5	Char	Ext	Full	Mand	'1E'
TFORM6	Char	Ext	Full	Mand	'1E'
EXTNAME	Char	Ext	NO	Impl	'SPECTRUM'
					Generic information
DATE	Date	YES ?	Full	Gener	date file written
ORIGIN	C*4	YES ?	NO	Impl	set implicitly to string 'XAS ' ?
FILENAME	C*39	YES ?	Full	Mand	name of spectrum file (without pathname and filetype extension)

Table 3.9a(cont.)

					spectrum specific info
STORAGE	Char	NO	Full	Gener	'BYROW'
TTYPE1	Char	Ext	Full	Gener	'DATA'
TUNIT1	Char	Ext	Full	Gener	'CTS/S' or other possible unit (see Table 3.10)
TDISP1	Char	Ext	Full	Opt	'G12.4' or '*n*G12.4' ?
TTYPE2	Char	Ext	Full	Gener	'ERRORS'
TUNIT2	Char	Ext	Full	Gener	same as TUNIT1
TDISP2	Char	Ext	Full	Opt	same as TDISP1
TTYPE3	Char	Ext	Full	Gener	'START_TIME'
TUNIT3	Char	Ext	Full	Gener	TBD
TDISP3	Char	Ext	Full	Opt	TBD
TTYPE4	Char	Ext	Full	Gener	'END_TIME'
TUNIT4	Char	Ext	Full	Gener	same as TUNIT3
TDISP4	Char	Ext	Full	Opt	same as TDISP3'
TTYPE5	Char	Ext	Full	Gener	'EXPOSURE'
TUNIT5	Char	Ext	Full	Gener	'SECONDS'
TDISP5	Char	Ext	Full	Opt	'F7.3'
TTYPE6	Char	Ext	Full	Gener	'DEAD_TIME'
TUNIT6	Char	Ext	Full	Gener	'PERCENTAGE'
TDISP6	Char	Ext	Full	Opt	'F7.3'
					instrument specific info
					TBW
					free keywords
COMMENT	Char	Ext	Full	Opt	any comment (optional)
HISTORY	Char	Ext ?	Full	Mand	sequence of commands which created the file
END		Ext	NO	Impl	end of FITS file (end of header is implicit)

Table 3.10 : unit codes for spectra

Keyword	Value	Comment
TUNITn	COUNTS/S	this is the suggested default for spectra
	COUNTS	this is a possible alternate unit
	COUNTS/S/KEV	another possibility is to normalize to channel width (useful in case of rebinning, specially for plotting)
	COUNTS/S/CM2	if wished to normalize to area
	PHOTONS/CM2/S/KEV	for deconvolved spectra, if sharing same format as count spectra
	Other TBD	for residual spectra, etc.

TBV with FITS convention whether a keyword can have imbedded blanks (e.g. 'PHOTON INDEX'). If not use underscore to replace blanks.

3.6. time profiles

The format proposed for time profiles of any sort is a table with as many rows as *time bins*, and a variable number of columns, self-described in the header. This should be able to accomodate semi-transparently a variety of formats (equispaced and non-equispaced bins, with or without gaps, with or without errors, multiple light curves in different bands, etc.). The format is quite *natural* and consistent with the usual need for access by row, although the small record length may pose efficiency problems (to be investigated).

3.6.1. header content

The time profile header content is described in Table 3.11. Note that in the case the table is written into FITS, it has to be handled as a *generalized extension*, therefore refer to section 3.4.1 above for the location of keywords.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

for spectral files	`MAGIC` is	$XAS_1BIN_2TIM_3\Delta\Delta\Delta_4$
in primary format	`RECLEN` is	`NAXIS1`
in primary format	`DATASIZE` is the number of time bins	

3.6.2. data field content

Each row corresponds to a time bin, which is in general a time interval from T_i to $T_i+\Delta T_i$ (in general ΔT_i will be constant and equal to ΔT, but the case of unequal bins cannot be a priori excluded). Each table row will correspond to a record (see however 3.4.2 above for the possibility of packing).

The number and order of columns is suggested here below.

If time information cannot be derived from the header (this is possible only if the file is an uninterrupted sequence of equispaced equal bins, that is if : $\forall i\ T_{i+1} = T_i+\Delta T$) at least one column shall be dedicated to store the t*ime of each bin* T_i (see also 3.6.3). If bins can be of unequal width an additional column shall store the *bin width*s ΔT_i . In this cases *gaps* are in general accounted for just by *missing data* (see below for another possibility).

In a majority of cases each bin has an *associated dead time* (to be expressed in TBD units), and this shall be stored in an additional column.

In general the *value* stored *as a function of time* $f(T_i)$ is a count rate, and as such it requires two columns, one for the value and one for the errors. However there are cases in which a value without error is used (either the error is not available, or it is redundant, as in the case of raw counts, where Poissonian statistics can be used). An header keyword `ERROR` could be used to indicate whether the value is stored in additional column (`COLUMN`), or is missing (`NONE`) or is Poissonian (`POISSON`) etc.

If it is wished to store the time profiles of *more than one variable* (e.g. count rates in different energy bands), this may occur adding more (couples of) columns. This way might be convenient for a small number of additional columns (as it saves duplicating the time information, and avoids the use of multiple files). However if a large number of columns is wished (e.g. the count rate in all PHA channels) and they are likely to be processed in parallel (e.g. Fourier-analysed) with a great saving in computing time, it might be simpler and more elegant to define only two columns (i.e. $f_j(T_i)$ exploiting the 3d feature of binary tables, one for data and one for errors, each with a dimensionality greater than one (that is pE type columns with j=1,p).

Gaps (or bad data ?) could also be accounted for by marking them using the IEEE NaN values (or other mechanism for NULL values defined in FITS). This is particularly sensible in the case of equispaced data without explicit indication of time.

It is suggested to use the time profile format also for *phase-folded light curves*: this will simply occur replacing (or adding to) the time column(s) with a phase column Φ_i (and $\Delta\Phi_i$). This has the advantage of a common handling .

Photon arrival time files are not covered here, but as a particular case of photon lists in 3.7

3.6.3. convention for time units

It is necessary to define an unique and self-consistent convention for time units: this is TBD but could be used expressing time with a suitable resolution (stored in header) starting from a reference time (stored in the header), which could be 00:0000.0 UT of the day containing the first data point (if the file spans more than one day it shall continue in the next day without recycling).

It is also necessary to agree a TBD convention to where does a bin start (i.e. does the T_i refer to the start or the middle of the bin ?).

Table 3.11 : time profile header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1BIN_2TIM_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1
DATASIZE	I*4	Deriv	Mini	Mand	no. of time bins
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	any value ? (use 8 ?)
NAXIS	I*2	YES	NO	Impl	0
EXTEND	L	YES	NO	Impl	always T (implicit)
END		YES	NO	Impl	this ends FITS primary header.
XTENSION	Char	Ext	NO	Impl	'BINTABLE'
BITPIX	I*2	Ext	NO	Impl	8
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	compute from number and type of rows
NAXIS2	I*2	Ext	Full	Mand	number of time bins
PCOUNT	I*2	Ext	NO	Mand	0
GCOUNT	I*2	Ext	NO	Mand	0
TFIELDS	I*2	Ext	Full	Mand	variable
TFORMn (1 if any)	Char	Ext	Full	Mand	TBD for time (see text)
TFORMn (2 if any)	Char	Ext	Full	Mand	same as TFORM1 (re-served for bin width)
TFORMn (2 or 3)	Char	Ext	Full	Mand	'1E' or TBD for dead time
TFORMn (3 or 4)	Char	Ext	Full	Mand	'1E' or 'pE' for data array
TFORMn (4 or 5)	Char	Ext	Full	Mand	'1E' or 'pE' for error array
TFORMn (5 or 6)	Char	Ext	Full	Mand	'1E' for data if other time profiles added
TFORMn (6 or 7)	Char	Ext	Full	Mand	'1E' for errors if other time profiles added
EXTNAME	Char	Ext	NO	Impl	'RATE' ?
					Generic information
DATE	Date	YES ?	Full	Gener	date file written
ORIGIN	C*4	YES ?	NO	Impl	set implicitly to string 'XAS ' ?
FILENAME	C*39	YES ?	Full	Mand	name of rate file (without pathname and filetype extension)

Table 3.11(cont.)

					spectrum specific info
STORAGE	Char	NO	Full	Gener	'BYROW'
TTYPEn (1 if any)	Char	Ext	Full	Gener	'TIME' or 'PHASE'
TUNIT1	Char	Ext	Full	Gener	TBD time unit (see Table 3.12)
TDISP1	Char	Ext	Full	Opt	TBD
TTYPEn (2 if any)	Char	Ext	Full	Gener	'BIN WIDTH'
TUNIT2	Char	Ext	Full	Gener	same as TUNIT1
TDISP2	Char	Ext	Full	Opt	same as TDISP1 TBV
TTYPEn (2 or 3)	Char	Ext	Full	Gener	'DEAD_TIME'
TUNIT2	Char	Ext	Full	Gener	TBD (% or seconds)
TDISP2	Char	Ext	Full	Opt	TBD
TTYPEn (3 or 4)	Char	Ext	Full	Gener	'DATA' or other label
TUNIT3	Char	Ext	Full	Gener	'CTS/S' or other (see Table 3.12
TDISP3	Char	Ext	Full	Opt	TBD
TTYPEn (4 or 5)	Char	Ext	Full	Gener	'ERROR'
TUNIT4	Char	Ext	Full	Gener	same as TUNIT3
TDISP4	Char	Ext	Full	Opt	same as TDISP4
TTYPEn (5 or 6)	Char	Ext	Full	Gener	'DATA1' or other label
TUNIT5	Char	Ext	Full	Gener	same as TUNIT3 ?
TDISP5	Char	Ext	Full	Opt	same as TDISP3 ?
TTYPEn (6 or 7)	Char	Ext	Full	Gener	'ERROR1'
TUNIT6	Char	Ext	Full	Gener	same as TUNIT5
TDISP6	Char	Ext	Full	Opt	same as TDISP5
ERROR	Char	Ext	Full	Gener	see text
REFTIME	Time	Ext	Full	Gener	reference time (see text)
Other TBD					flags to indicate equispaced, gaps etc. plus BINWIDTH etc. also ephemeris information etc.
					instrument specific info
					TBW
					free keywords
COMMENT	Char	Ext	Full	Opt	any comment (optional)
HISTORY	Char	Ext ?	Full	Mand	sequence of commands which created the file
END		Ext	NO	Impl	end of FITS file (end of header is implicit)

Table 3.12 : unit codes for time profiles

Keyword	Value	Comment
TUNIT1	TBD	time units
	PHASE	for phase
TUNIT2	PERCENT	or other unit for dead time
TUNIT3	COUNTS/S	default unit for data and errors
	HARDNESS RATIO	ev. better specified with indication of energy bands
	Other TBD	for other cases including non X-ray

TBV with FITS convention whether a keyword can have imbedded blanks (e.g. '`PHOTON INDEX`'). If not use underscore to replace blanks.

3.7. photon lists

Photon lists are naturally handled as tables with each row corresponding to a photon, and as many columns as associated information. The number, order and format of the columns shall be highly flexible (and self documented in the header) to allow for a variety of cases corresponding to the different telemetry/operative modes, and to further selections done on the ground. This should allow an uniform handling of direct mode data. The format is quite *natural* and consistent with the usual need for access by row, although the small record length may pose efficiency problems (to be investigated)

3.7.1. header content

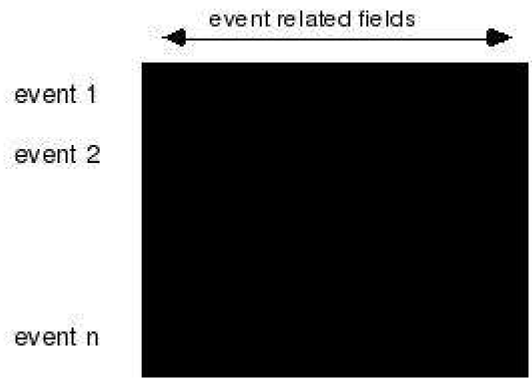
The time profile header content is described in Table 3.13. Note that in the case the table is written into FITS, it has to be handled as a *generalized extension*, therefore refer to section 3.4.1 above for the location of keywords.

The inter-relation between mini-header keywords and FITS keywords is implicitly determined as follows:

for spectral files	`MAGIC` is	$XAS_1BIN_2PHO_3\Delta\Delta\Delta_4$
in primary format	`RECLEN` is	`NAXIS1`
in primary format	`DATASIZE` is the number of events	

3.7.2. data field content

Each individual event will correspond to a row, as illustrated in the figure below. The kind of information (xy position, energy, burst length, time etc.) to be present for each row will vary, and will be described in the header by conventional codes.



It is likely that most of the information (as it is coming from an ADC) will be kept in binary integer format (typically I*2, but I*4 *with particular* TBD *conventions* could be necessary for time; it is also TBD whether the use of *byte fields* is acceptable). The format of a particular field will be described in the header, and anyhow shall (TBV) unique for a particular kind of information.

In the case of time it is suggested to adopt the same convention defined for rate files (see 3.6.? above), that is to express times of programmable resolution relative to some reference time (typically 00:0000.00 UT of the day in which the first event falls) stored in the header as `REFTIME`.

Table 3.13 : photon list header

Keyword	Type	FITS	Located	Class	Comment
					Values in mini-header
MAGIC	Char	NO	Mini	Mand	value is $XAS_1BIN_2PHO_3\Delta\Delta\Delta_4$
RECLEN	I*4	Deriv	Mini	Mand	equal to NAXIS1
DATASIZE	I*4	Deriv	Mini	Mand	no. of time bins
HDRSIZE	I*4	NO	Mini	Mand	variable
					FITS mandatory keywords
SIMPLE	L	YES	NO	Impl	always T (implicit)
BITPIX	I*2	YES	Full	Mand	any value ? (use 8 ?)
NAXIS	I*2	YES	NO	Impl	0
EXTEND	L	YES	NO	Impl	always T (implicit)
END		YES	NO	Impl	this ends FITS primary header.
XTENSION	Char	Ext	NO	Impl	'BINTABLE'
BITPIX	I*2	Ext	NO	Impl	8
NAXIS	I*2	Ext	NO	Impl	2
NAXIS1	I*2	Ext	Full	Mand	compute from number and type of rows
NAXIS2	I*2	Ext	Full	Mand	number of events
PCOUNT	I*2	Ext	NO	Mand	0
GCOUNT	I*2	Ext	NO	Mand	0
TFIELDS	I*2	Ext	Full	Mand	variable
TFORMn	Char	Ext	Full	Mand	to be chosen in the list in Table 3.14
EXTNAME	Char	Ext	NO	Impl	'PHOTON LIST' ?
					Generic information
DATE	Date	YES ?	Full	Gener	date file written
ORIGIN	C*4	YES ?	NO	Impl	set implicitly to string 'XAS ' ?
FILENAME	C*39	YES ?	Full	Mand	name of photon list file (without pathname and filetype extension)

Table 3.13 (cont.)

					spectrum specific info
STORAGE	Char	NO	Full	Gener	'BYROW'
TTYPEn	Char	Ext	Full	Gener	to be chosen in the list in Table 3.14
TUNITn	Char	Ext	Full	Gener	to be chosen in the list in Table 3.14
TDISPn	Char	Ext	Full	Opt	TBD see Table 3.14
REFTIME	Time	Ext	Full	Gener	reference time (see text)
Other TBD					TBD
					instrument specific info
					TBW
					free keywords
COMMENT	Char	Ext	Full	Opt	any comment (optional)
HISTORY	Char	Ext ?	Full	Mand	sequence of commands which created the file
END		Ext	NO	Impl	end of FITS file (end of header is implicit)

Table 3.14 : units and data types for photon lists

Information	TFORMn	TTYPEn	TUNITn	TDISPn	notes
x position	1I (1B)	XPOSITN	PIXEL	I6	
y position	1I (1B)	YPOSITN	PIXEL	I6	
time	1J (1I)	TIME	TBD	TBD	1
energy	1I (1B)	ENERGY	PHA CHANNEL	I3, I4	
burst length	1B (1I)	BL	CHANNEL	I3	
risetime	1I (1B)	RT	CHANNEL	I3, I4	
veto	1I (1B)	VETO	CHANNEL	I3	
other digital info	TBD	TBD	TBD	TBD	
right ascension	1D ?	RA TBV	DEGREES	???	2
declination	1D ?	DEC TBV	DEGREES	???	2

Note 1: time has to be encoded (into an integer word TBV) according to TBD convention (hopefully the same used for rate files and for time keywords in the header). As such any `In` format could be used in FITS-like header for display, however display will be normally done according to the formats foreseen for `TIME` variables.

Note 2: angular coordinates have to be encoded (into a double precision float) according to TBD convention (hopefully the same used for `ANGLE` keywords in the header). As such any `Dw.n` format could be used in FITS-like header for display, however display will be normally done according to the formats foreseen for `ANGLE` variables.

4. test results

The tests planned and described above in section 2 have been performed and are described in [Ref.6]. The tests were essentially aimed to assess the i/o efficiency. The conclusion of such tests were in favour of the use of the XAS file format, which proves to be sometimes substantially faster than FITS, and only slightly more efficient for what concerns disk space usage. At the same time the usage of FITS for transport purposes was confirmed as valuable. During the tests some (marginal) problems emerged, which required some changes to the original specification.

Some further tests, leading to a partial redefinition of the present specifications, were made in the form of the preparation of a set of prototype libary routines for XAS file access, and of demo programs using them (in this case there was no aim on verifying i/o speed). This is described in [Ref.7]. The author gratefully acknowledges discussions with and suggestions by D.Dal Fiume, some of which are reflected in the changes to the original specification.

The next section reports all changes to the original specification which have been found necessary for the sake of efficiency, for cleanliness or for compatibility with FITS. Such changes are essentially of "*syntactic*"nature : the format of the file should now be almost completely defined from the point of view of basic access via software. More refinement of "*semantic*"nature (concerning e.g. instrument-specific parts, or usage-related issues) are likely to be necessary in the future and are not reflected in the present issue of the document.

5. revised proposal

This chapter describes the final specification, as it arises from the tests and prototyping work described in Section 4. To avoid re-issue of bulky material, we make reference to the the previous proposal (section 3 of present document, issue 1.0), whenever the specification is unchanged, and report here only updated material.

Whenever part of the previous proposal is reported *verbatim* with some changes, this is indicated by a double bar on the right margin. Red type details the most relevant changes.

The logical file structure described in the introduction to section 3 (pag. 3-0.1-2) is still valid. The current choice is to have a XAS file structure which is not FITS, but however can immediately be mapped to it.

Most elements on pag. 3-0.1-2 are confirmed. The only changes to the file structure involve the possibility of splitting the mini-header on more than one record, and the possibility of appending miscellaneous data after the main header.

The order of the information will therefore be :

```
p records:   mini-header (unused part of last record padded with
             binary zeros; p is 1 if file record length is
             greater-equal to 28 bytes; otherwise p*recl must be
             the first multiple of recl greater than 28)
n records:   data area (by definition this will have 100% filling
             efficiency)
m records:   full header area (unused part of last record padded
             with binary zeros)
q records:   normally q=0, otherwise miscellaneous information
             not covered by this specification may follow the
             full header area. The content of this area is
             subject to being overwritten without notice if the
             file is modified using official XAS routines.
```

5.1. Common header format

5.1.1. mini-header content

The mini-header will always occupy the first 28 bytes of the file . This means that if the file recl is longer than 28 bytes, the mini-header will occupy the the first 28 bytes of the first record, and the remainder will be filled with binary zeros. If the file recl is shorter than 28 bytes, the mini header will occupy p records, such that p*recl is greater or equal to 28. Any spare bytes in the last record will be filled with binary zeros.

The mini-header has a special format designed for quick access (used also by VOS/Unix routines to determine the record length) and includes the fields in table 3.1:

The *magic number* will "univocally" identify a XAS data file. The hierarchical arrangement of 16 bytes into 4 4-byte fields, proposed in section 3.1.1 is confirmed, with the following refinement. Each 4-byte field will consist of 3 printable characters and a binary value. The layout is indicated in section 3.1.1, but the fourth field will be as defined here :

XAS$_1$*PPP*$_2$*TTT*$_3$SSS4

where the subscript notation indicates an octal value (i.e. $_1$ is octal 001 etc.).

the fourth field SSS (followed by octal 004) is reserved and will contain a three-character code indicating the operating system on which the file was written (e.g. 'VAX', 'DEC', 'SUN' etc.)

5.1.2. full header content

The full header is as described in section 3.1.2, but one shall note what follows :

- The usage of the INTEGER*2 (I*2, short int, 16-bit integer) data type is deprecated. One shall use the INTEGER*4 data type for any integer quantity

Each keyword has a **value**, which is encoded in the machine-specific binary internal representation appropriate for the data type. Usage of the operating system subfield in the magic number shall allow transparent conversion if the system where the file is being read is not the same as the one where it was written. The *length* of the value field is always explicitly indicated *:* it is equal to the string length for character keywords, while for numeric keywords it can be any multiple of the type-specific scalar unit. This allows numeric keyword to be arrays. Usage of array keywords is supported but is discouraged (for compatibility with FITS which does not have array keywords).

The **length** of character values may be in line of principle any value between 1 and 255 (I suggest to limit it in practice to *even* values either of 246 bytes (so that the total size of a keyword does not exceed 256 bytes) or to 68 bytes (which is the maximum size of a FITS keyword).
The length of a scalar numeric keyword is generally 2,4,8 bytes according to the data type. The length of an array keyword is a multiple of that, and the overall length shall not exceed the maximum length of a character value defined above (this gives a limit on the number of array elements).

The format of a keyword is specified below in table 5.2

The presence and order of keywords in the header is specified in the detailed description by data type below. Given the guidelines in section 3.1.2, *which are to be intended as pure recommendations*, the following should however be noted for what concerns the *order* of the keywords :

- the order of the keywords *does not matter* for XAS files (the software can read existing keywords in any order, due to the mechanism which keeps the entire header in storage)
- The order of mandatory FITS keywords will be determined by the FITSIO library when converting to FITS. All other keywords will follow *in the order in which they appear in the XAS file*.
- A program may modify the value of existing keywords, but *not change* their order, type and length (or number of elements) *nor delete* them. A program shall append new keywords *after* existing keywords.
- kewyords *shall not be duplicated*, i.e. no two keywords with the same name may be present in the header, *with the exception* of particular keywords, so far represented by the `COMMENT`, `HISTORY` and `PARENTS` keywords.

The data type specific tables, from table 5.3 onwards, have been modified wrt tables 3.3 onwards to better specify which keywords are present in the XAS or FITS file only and which in both.

A set of routines to access keywords as specified above has been developed and is described in [Ref.7].

Table 5.2 :format of header keywords

Bytes	Type	Field	Comments
01	Byte	Type	Data type (see below)
02	Byte	Length	Data length (see below)
03 to 10	Char*8	Name	Keyword name, left-justified and padded wi blanks
01 to x	var.	Value	Value (see below)

For I*2 data type (officially deprecated)

Bytes	Type	Field	Comments
01	Byte	Type=I*2	1
02	Byte	Length	2 * number of elements
11 to 12		Value	INTEGER*2 values (1 to 34)

For I*4 data type

Bytes	Type	Field	Comments
01	Byte	Type=I*4	2
02	Byte	Length	4 * number of elements
11 to 14		Value	INTEGER*4 values (1 to 17)

For R*4 data type

Bytes	Type	Field	Comments
01	Byte	Type=R*4	3
02	Byte	Length	4 * number of elements
11 to 14		Value	REAL*4 values (1 to 17)

For R*8 data type

Bytes	Type	Field	Comments
01	Byte	Type=R*8	4
02	Byte	Length	8 * number of elements
11 to 18		Value	REAL*8 values (1 to 8)

For character data type

Bytes	Type	Field	Comments
01	Byte	Type=char	0
02	Byte	Length	m (shall be an even number, minimum 2, maximum 68 TBD)
11 to p (p=10+m)	Char*m	Value	Character string (left justified, blank padded)

Provisionally refer to table 3.2 for other data types (not implemented yet

5.2. images

The image format as described in 3.2, with the amendments reported here, will be used for any 2-d data suitable to be represented as a function $z=f(x,y)$, as well as for 1-d histograms ($z=f(x)$ with x equispaced) as a degenerate case. It will also be possible to pack more 2-d images in a single file as a pseudo-3-d image (this is supported by the specification but is not officially encouraged).

The image header content is described in section 3.2.1 and Table 5.3. The data field content is described in section 3.2.2. The following notes are added to such specification :

If `NAXIS2` is absent, it shall default to a value of 1. It is recommended to include always this keyword, even for 1-d histograms.
If `NAXIS3` is absent, it shall default to a value of 1. This keyword shall be present only if `NAXIS3` images are stacked in the same file.

The official XAS format will be such to use `REAL*4` images (`BITPIX=-32`). `INTEGER*2` images (`BITPIX=16`) are supported at specification level, but they will not be supported officially at software level, and are officially discouraged.

5.3. matrices

The format proposed in section 3.3 is confirmed. In particular the FITS `IMAGE` extension (in process of being officially approved by IAU, see [Ref.8]) will be used when converting the XAS file to FITS (the matrix will be in the main HDU, and the associated histogram in the first - and only - extension).

The image header content is described in section 3.3.1 and Tables 5.6-7. The data field content is described in section 3.3.2.

5.4. tables

The format of generic binary tables is confirmed as indicated in section 3.4, with the refinements noted below. It is confirmed that all table files (including the specific cases described in the next sections) are converted to FITS as `BINTABLE` extensions.

The generic table header content is described in Table 5.8. All information in the XAS full header will be copied to the FITS extension header, with the exception of the few mandatory keywords in the main header.

For what concern the data field, the present form of data storage in *all* binary tables is by row (one file record is one table row). This *might be* documented by the `STORAGE='BYROW'` keyword, in the case an alternate storage by column could be supported in the future. See 3.4.2 for more details.

5.5. spectra

The format of the spectra has been quite modified w.r.t. the two proposals presented in section 3.5. The current proposal easily accomodates in an elegant, binary-table compatible way, a *single spectrum* as well as *multiple spectra*, provided they all share a common set of channel boundaries. This is chiefly intended to allow storage in the same file of spectra produced simultaneously by separate instrument units (e.g. PDS). However it could also accomodate also a *sequence of spectra* taken at different times, provided the channel boundaries did not vary (however there is no easy and elegant way to store the information about the times of the individual spectra; I personally regard as simpler to have programs handling lists of single-spectrum files).

The spectrum header content is described in Tables 5.9

The proposed format stores either a *single* spectrum or `n` spectra in a file. The file is a table with 4 multi-dimensional columns and *m* rows (with *m* equal to the number of PHA channels or bins). For commonality of handling with all other binary table files, it is proposed to store data by row, even if this is slightly inefficient from the i/o point of view (but these files are quite small) : each record will correspond to an energy bin.

The 4 columns correspond respectively and in this order to :

1 lower channel boundaries (scalar floating point, FITS `1E`)
2 upper channel boundaries (scalar floating point, FITS `1E`)
3 data (cts/s) for the `n` spectra (floating point array, FITS `nE`)
4 errors for the n spectra (floating point array, FITS `nE`)

5.6. time profiles

The format proposed for time profiles in section 3.6 is essentially confirmed, with the refinements indicated below.

The time profile header content is described in Table 5.11. Some of the keywords are present only depending on the data field content, as explained in the next subsection.

For what concerns the data field, each row corresponds to a time bin, and to a record (packing is for unimplemented). See 3.6.2 for terminology.

The number of columns may vary, and a particular column may not be present, but the order of columns *which are present* shall follow what is suggested here below. This implies adoption of a well-defined, standard nomenclature for columns (`TTYPEn` keywords).

The time of each bin (`TTYPE1='TIME'`) shall come first. If time is absent, it is assumed that bins are equispaced, starting from `timezero` in steps of `binsize` as indicated in header keywords `TIMEZERO` and `BINSIZE`. Keywords `TUNIT1` (or `TIMEUNIT` for equispaced files) shall indicate the time units according to a convention TBD. This column is 1-dimensional integer (FITS `1J` format) TBV.

The next column shall contain the binsize (`TTYPEn='BINSIZE'`). The bin size may be absent for equispaced files (when also time is missing) and also for non-equispaced files with equal-width bins. In this case the information is indicated in the header in keyword `BINSIZE`. Time units shall be the same as those used for time (duplicating the value of `TUNIT1` into another `TUNITn` keyword, or using the same `TIMEUNIT` one). This column is 1-dimensional integer (FITS `1J` format) TBV.

The first two columns may be replaced by phase and phase binsize for folded light curves (it is sufficient to set `TTYPE1` to `'PHASE'` ; additional keywords TBS should contain ephemeris data).

The next column shall contain the deadtime (`TTYPEn='DEADTIME'`), as a 1-dimensional floating point value (FITS `1E`). It has to be agreed whether the units (`TUNITn`) are percentage (0-100) or fraction (0.0-1.0), and whether one stores the actual dead time, or the dead time correction factor to be applied to counts. If this column is missing, it could be that dead time info is not available or not applicable (`DEADTIME='NONE'`), or that it is constant (time-independent; `DEADTIME='FIXED',` the value shall be given in another keyword `DTVALUE` TBV). If this column is present, keyword `DEADTIME` shall be `'TIME DEPENDENT'`. An additional keyword may specify if deadtime correction (either fixed or time-dependent) has been applied to the data (e.g. `DTCORR = 'NOT APPLIED'`, `'FIXED'` or `'TIME DEPENDENT'` TBV)

The next column shall contain data (`TTYPEn='DATA'`). This column is compulsory. This column can be multi-dimensional (FITS format `nE`) if one wants to store information pertaining to `n` related quantities (e.g. `n` energy bands). Note that if unrelated quantities are stored in the same file, they should appear as separate columns.

The next column shall contain the errors (`TTYPEn='ERRORS'`) on the previous data column, and shall have the same dimensionality. If this column is absent, it might mean that errors are not available or applicable (keyword `ERROR='NONE'`), or that they can be derived as Poisson errors from the data (keyword `ERROR='POISSON'`). For documentation one should include `ERROR='COLUMN'` if errors are present.

More columns may follow containing additional time series for other unrelated quantities, and their errors. For simplicity it is suggested that such columns be only 1-dimensional, and be present only if the main data column is 1-dimensional too. These columns must / must not be followed by an error column if the main data column has /has no errors. A possible naming for `TTYPEn` is `DATA1`, `ERROR1`, `DATA2`, etc. (a set of additional keywords `DATAn` might record more information on which kind of quantity is used, than it could be specified in `TUNITn`, e.g. if `TTYPE6='DATA1'` contains the temperature of the outer mirror of MECS unit 1, `TUNIT6` will just say "Kelvin" or "Celsius", but `DATA1` might specify a descriptive string.

The convention used to mark gaps (or bad data ?) is yet TBS.

The definition of a convention for time units is still pending and will be specified separately later.

5.7. photon lists

Photon lists are handled as described in section 3.7, with the refinements specified here below.

The photon list header content is described in Table 5.13.

For what concerns the data field each individual event will correspond to a row, as described in section 3.7.2. The order of information within each row should conform to the following requirements :

It is strongly desirable for alignment reasons that all 32-bit fields come first, all 16-bit fields come next, and all 8-bit fields come last.

All data columns shall be 1-dimensional (`1J`, `1I` or `1B` FITS format)

The order of the columns shall be the same, *at least* for files pertaining to the same instrument. Within the given order, a particular column may or may not be present in a specific file. This implies adoption of a well-defined, standard nomenclature. A proposal is presented in Table 5.15

The size of a row in bytes (`NAXIS1`) shall be a multiple of 4 bytes (this is required by unformatted direct access on VMS and Ultrix systems). If the data columns do not total to such a number, a padding column (1-3 bytes) shall be added. This column may be multi-dimensional, from `1B` to `3B` format. This column will be specified by a `TFORMn` keyword without name (no `TTYPEn` or `TUNITn`) and shall be ignored when reading.

The convention for time units (see 3.7.2) is yet TBD.

Note: all tables reported below are numbered 5.x, where x is the same as the corresponding table 3.x, of which the present table represent an update. There is no table 5.x when there are no changes w.r.t. the previous version in Section 3.

Table 5.3 : image header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1IMG_2INT_3$*SSS*$_4$ or $XAS_1IMG_2FLO_3$*SSS*$_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1*2 or NAXIS1*4
DATASIZE	I*4	absent	Mini-h	equal to NAXIS2
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	primary	absent	always T (implicit)
BITPIX	I*4	primary	Header	16 or -32 see text
NAXIS	I*4	primary	absent	implicit in max NAXISn
NAXIS1	I*4	primary	Header	no of pixels in x
NAXIS2	I*4	primary	Header	no of pixels in y (1 for histograms)
NAXIS3	I*4	primary	Header optional	number of images in file (omit if 1)
				Generic information
DATE	Date	primary	Header	date file written
ORIGIN	C*4	primary	Header	'XAS '
DATE-OBS	Date	primary	Header	date of observation
FILENAME	C*39	primary	Header	name of image file (without pathname and filetype extension)
SATELLIT	C*4	primary	Header	satellite name 'SAX' (replaces FITS TELESCOP keyword).
INSTRUME	C*4	primary	Header	instrument code (see Table 3.4)
OBSERVER	C*16	primary	Header	Observation PI name
OBJECT	C*16	primary	Header	target name
EQUINOX	R*4	primary	Header	2000.0 for FITS compatibility

For all remaining keywords make so far reference to the second part of table 3.3 (page [T3.3]-2. These specifications might be detailed in the future.

Table 5.6 : response matrix header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1IMG_2MAT_3\mathit{SSS}_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1*4
DATASIZE	I*4	absent	Mini-h	equal to NAXIS2
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	Primary	absent	always T (implicit)
BITPIX	I*4	Primary	Header	-32
NAXIS	I*4	Primary	absent	2
NAXIS1	I*4	Primary	Header	no of energies
NAXIS2	I*4	Primary	Header	no of PHA channels
EXTEND	L	Primary	absent	always T (implicit)
				Generic information
DATE	Date	Primary	Header	date file written
ORIGIN	C*4	Primary	Header	'XAS '
DATE-OBS	Date	Primary	Header	date of observation (could be useful if response changes with time)
FILENAME	C*39	Primary	Header	name of matrix file (without pathname and filetype extension)
SATELLIT	C*4	Primary	Header	satellite name 'SAX' (replaces FITS TELESCOP keyword).
INSTRUME	C*4	Primary	Header	instrument code (see Table 3.4)
OBSERVER	C*16	Primary	Header	Observation PI name
OBJECT	C*16	Primary	Header	target name

For all remaining keywords make so far reference to the second part of table 3.6 (page [T3.6]-2. These specifications might be detailed in the future.

Table 5.7 : response matrix associated histogram header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1IMG_2FLO_3SSS_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1*4
DATASIZE	I*4	absent	Mini-h	1 (equal to NAXIS2)
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
XTENSION	Char	Extension	absent	IMAGE (implicit)
BITPIX	I*4	Extension	Header	-32
NAXIS	I*4	Extension	absent	2 (implicit)
NAXIS1	I*4	Extension	Header	no of energies
NAXIS2	I*4	Extension	Header	1
PCOUNT	I*4	Extension	absent	0 (implicit)
GCOUNT	I*4	Extension	absent	0 (implicit)
all remaining information is optional and should be identical to corresponding fields in main response matrix with the exceptions noted below (for the rest make reference to table 3.7)				
				Generic information
FILENAME	C*39	Extension	Header	name of histogram file (without pathname and filetype extension)
				matrix specific info
BUNIT	Char	Extension	Header	'KEV' (specific)
CTYPE1	Char	Extension	Header	'ENERGY BINS'
CRPIX1		Extension		1
CRVAL1		Extension		start energy in keV
CDELT1		Extension		1
DATAMAX	R*4	Extension	Header	end energy
DATAMIN	R*4	Extension	Header	start energy
ASSOCMAT	C*39	Extension	Header	name of associated matrix file

Table 5.8 : generic table header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1BIN_2GEN_3\mathit{SSS}_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1
DATASIZE	I*4	absent	Mini-h	equal to NAXIS2
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	Primary	absent	always T (implicit)
BITPIX	I*4	Primary	Header	always 8
NAXIS	I*4	Primary	absent	always 0 (implicit)
EXTEND	L	Primary	absent	always T (implicit)
END		Primary	absent	this ends FITS primary header.(implicit)
XTENSION	Char	Extension	absent	'BINTABLE' (implicit)
BITPIX	I*4	Extension	absent	8 (implicit)
NAXIS	I*4	Extension	absent	2 (implicit)
NAXIS1	I*4	Extension	Header	size of a row in bytes
NAXIS2	I*4	Extension	Header	number of rows in table
PCOUNT	I*4	Extension	absent	0
GCOUNT	I*4	Extension	absent	1
TFIELDS	I*4	Extension	Header	number of columns
TFORMnnn	Char	Extension	Header	formats of the columns in FITS binary table standard
				FITS auxiliary keywords these keywords are not strictly mandatory, but may be highly desirable
TTYPEnnn	Char	Extension	Header	label of column nnn
TUNITnnn	Char	Extension	Header	units of column nnn
TDISPnnn	Char	Extension	Header optional	display format (Fortran 90) for column nnn
EXTNAME	Char	Extension	absent	'GENERIC'(implicit)

Refer to table 3.8 for all other XAS-specific keywords common to all XAS files.

Table 5.9 : spectrum header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1BIN_2SPE_3\mathit{SSS}_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1
DATASIZE	I*4	absent	Mini-h	equal to NAXIS2
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	Primary	absent	always T (implicit)
BITPIX	I*4	Primary	Header	8
NAXIS	I*4	Primary	absent	0
EXTEND	L	Primary	absent	always T (implicit)
END		Primary	absent	implicit
XTENSION	Char	Extension	absent	'BINTABLE'
BITPIX	I*4	Extension	absent	8
NAXIS	I*4	Extension	absent	2
NAXIS1	I*4	Extension	Header	4*(2n+2) where n is the number of spectra
NAXIS2	I*4	Extension	Header	number of PHA channels or bins (rows in table)
PCOUNT	I*4	Extension	absent	0
GCOUNT	I*4	Extension	absent	1
TFIELDS	I*4	Extension	Header	4
TFORM1	Char	Extension	Header	'1E'
TFORM2	Char	Extension	Header	'1E'
TFORM3	Char	Extension	Header	'nE' (n=no. of spectra)
TFORM4	Char	Extension	Header	'nE'
EXTNAME	Char	Extension	absent	'SPECTRUM'
				spectrum specific info
TTYPE1	Char	Extension	Header	'LOWER BOUNDARY'
TUNIT1	Char	Extension	Header	'KEV'
TDISP1	Char	Extension	Header	'F8.4' (optional)
TTYPE2	Char	Extension	Header	'UPPER BOUNDARY'
TUNIT2	Char	Extension	Header	same as TUNIT1
TDISP2	Char	Extension	Header	same as TDISP1
TTYPE3	Char	Extension	Header	'DATA'
TUNIT3	Char	Extension	Header	see table 3.10
TDISP3	Char	Extension	Header	'G12.4'(optional)
TTYPE4	Char	Extension	Header	'ERRORS'
TUNIT4	Char	Extension	Header	same as TUNIT3
TDISP4	Char	Extension	Header	same as TDISP3'
FILENAME	C*80	[illegible]	Full	name of spectrum file [illegible]

The generic info (DATE to OBJECT or EQUINOX) and the free keywords are as in table 5.3. Instrument specific information is still TBD.

Table 5.11 : time profile header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1BIN_2TIM_3$*SSS*$_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1
DATASIZE	I*4	absent	Mini-h	no. of time bins
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	Primary	absent	always T (implicit)
BITPIX	I*4	Primary	Header	8
NAXIS	I*4	Primary	absent	0
EXTEND	L	Primary	absent	always T (implicit)
END		Primary	absent	implicit
XTENSION	Char	Extension	absent	'BINTABLE'
BITPIX	I*4	Extension	absent	8
NAXIS	I*4	Extension	absent	2
NAXIS1	I*4	Extension	Header	compute from number and type of rows
NAXIS2	I*4	Extension	Header	number of time bins
PCOUNT	I*4	Extension	absent	0
GCOUNT	I*4	Extension	absent	1
TFIELDS	I*4	Extension	Header	variable
TFIELDS columns MAY follow in the order specified in 5.6 above				
TFORMn	Char	Extension	Header	'1J' for time TBV
TFORMn	Char	Extension	Header	same as TFORM1 for binsize
TFORMn	Char	Extension	Header	'1E' for deadtime
TFORMn	Char	Extension	Header	'nE' for data
TFORMn	Char	Extension	Header	'nE' for errors
TFORMn	Char	Extension	Header	'1E' for additional data
TFORMn	Char	Extension	Header	'1E' for additional errors
EXTNAME	Char	Extension	absent	'RATE'
				time specific info
TTYPEn	Char	Extension	Header	'TIME' or 'PHASE'
TUNITn	Char	Extension	Header	time unit (see Table 3.12)
TDISPn	Char	Extension	Header	TBD
TIMEZERO	Time	Extension	Header	time reference
BINSIZE	Time	Extension	Header	bin size
TIMEUNIT	Char	Extension	Header	time unit (see Table 3.12) for 2 preceding keywords
TTYPEn	Char	Extension	Header	'BINSIZE'
TUNITn	Char	Extension	Header	same as TUNITn for time
TDISPn	Char	Extension	Header	same as TDISP1 TBV
TTYPEn	Char	Extension	Header	'DEADTIME'
TUNITn	Char	Extension	Header	TBD (% or seconds)
TDISPn	Char	Extension	Header	TBD
DEADTIME	Char	Extension	Header	see 5.6 for values
DTCORR	Char	Extension	Header	see 5.6 for values
DTVALUE	Real	Extension	Header	value of fixed deadtime
DTUNIT	Char	Extension	Header	Unit for DTVALUE (same as TUNITn for deadtime)

Table 5.11 (cont).

TTYPEn	Char	Extension	Header	'DATA' or other label
TUNITn	Char	Extension	Header	'CTS/S' or other (see Table 3.12
TDISPn	Char	Extension	Header	TBD
TTYPEn	Char	Extension	Header	'ERROR'
TUNITn	Char	Extension	Header	same as TUNITn for data
TDISPn	Char	Extension	Header	same as TDISPn for data
TTYPEn	Char	Extension	Header	'DATA1' or other label
TUNITn	Char	Extension	Header	same as TUNITn for data
TDISPn	Char	Extension	Header	same as TDISPn for data
TTYPEn	Char	Extension	Header	'ERROR1'
TUNIT6	Char	Extension	Header	same as TUNITn for data
TDISP6	Char	Extension	Header	same as TDISPn for data
ERROR	Char	Extension	Header	see 5.6 for values

The generic info (DATE to OBJECT or EQUINOX) and the free keywords are as in table 5.3. Instrument specific information is still TBD.

Table 5.13 : photon list header

Keyword	Type	FITS	XAS	Comment
				Values in mini-header
MAGIC	Char	absent	Mini-h	value is $XAS_1BIN_2PHO_3$*SSS*$_4$
RECLEN	I*4	absent	Mini-h	equal to NAXIS1
DATASIZE	I*4	absent	Mini-h	no. of time bins
HDRSIZE	I*4	absent	Mini-h	variable
				FITS mandatory keywords
SIMPLE	L	Primary	absent	always T (implicit)
BITPIX	I*4	Primary	Header	8
NAXIS	I*4	Primary	absent	0
EXTEND	L	Primary	absent	always T (implicit)
END		Primary	absent	implicit
XTENSION	Char	Extension	absent	'BINTABLE'
BITPIX	I*4	Extension	absent	8
NAXIS	I*4	Extension	absent	2
NAXIS1	I*4	Extension	Header	compute from number and type of rows
NAXIS2	I*4	Extension	Header	number of events
PCOUNT	I*4	Extension	absent	0
GCOUNT	I*4	Extension	absent	1
TFIELDS	I*4	Extension	Header	variable
TFORMn	Char	Extension	Header	to be chosen in the list in Table 5.15/3.14
EXTNAME	Char	Extension	absent	'PHOTON LIST'
				photon list specific
TTYPEn	Char	Extension	Header	to be chosen in the list in Table 5.15/3.14
TUNITn	Char	Extension	Header	to be chosen in the list in Table 3.14
TDISPn	Char	Extension	Header	TBD see Table 3.14

The generic info (DATE to OBJECT or EQUINOX) and the free keywords are as in table 5.3. Instrument specific information is still TBD.

Table 5.15 : event information types

The first part of this table supplements table 3.14 indicating which information might be present for each SAX instrument and what shall be the default bit-width (which is fixed to the maximum bit width present for any mode for that instrument). This table uses 1-2 character mnemonics.

Instr.	T	X	Y	U	C	BL	RT	V	ID	F1	F1	ca	Xn	S	P	Pn	z	EG	ES	O
LECS	1J	1B	1B	1B	1B	1B		1B												
MECS	1J	1B	1B	1B		1B				1B			4B							
HPGSPC	1j			1I	1I	1B				1B	1B	1B		1B	1I	7i				
PDS	1J			1I			1I		1B	1B										
WFCs	1J	1i	1i	1B													1B	1B	1I	nI

The second part of this table presents possible names for the binary table columns (`TTYPEn` keywords) corresponding to each information, indexed on the mnemonics used above.

In general a longer, self-explanatory name should be preferred to a cryptic code, but this shall be agreed, particularly for the more commonly used info, present for almost all instruments. Information present in diagnostic modes or otherwise rarely used could follow a different rule (note that I suggested to handle diagnostic information like raw anode readouts as arrays, to avoid cluttering the header with lots of column names) : it these data never reach the general observer this could be left at discretion of instrument teams.

?	TTYPEn	explanation and comments
T	TIME	normalized format TBD
X	XPOSITN	verbose form 'XPOSITN' is preferred to 'X' ? shall
Y	YPOSITN	one use even a longer one 'X POSITION' ?
U	PHA	this is the only or uncorrected energy information; PHA is preferred to ENERGY as name (or use E, UE ?)
C	CE ?	corrected energy when applicable
BL	BL	burst length
RT	RT	rise time
V	VETO	veto
ID	ID	unit identifier
F1	FLAG	flag : in/out (MECS); correl. event (HPGSPC); coincidence (PDS)
F2	CAL FLAG	calibration flag
ca	CAL ID	calibration source id
Xn	RAW POSITN	X1 X2 Y1 Y2 raw anode readout (MECS diagnostic)
S	SECTOR	HPGSPC only
P	PIXEL ?	HPGSPC only
Pn	PM READOUT	HPGSPC diagnostic only
Z	Z PLANE	longer form preferred to 'Z' ?
EG	GUARD PHA	WFC BAM only
ES	EVENT STAT	WFC BAM and diagnostic
O	other TBD	WFC diagnostic modes

6. references

[1] Chiappetti, L. 1991, "Ockham's Razor" Handout of the presentation at the Palermo SAX DAWG Meeting, May 13 1991 (DAWG-REP.2/91)

[2] Chiappetti, L. & Morini, M. 1990, "A proposal for XAS software specifications" Final draft 0.5 January 5 1990 (DAWG-REP.1/90)

[3] Pence, W. D. 1992, "FITSIO. A subroutine interface to FITS format files" release 3.21 July 1992 (supersedes DAWG-EXT 22/91 and release 3.0 reference in issue 1.0 of present document)

[4] NASA/OSSA Office of Standards and Technology 1991, "Implementation of the Flexible Image Transport System (FITS)" Draft standard NOST 100-0.3b, November 6 1991 (supersedes DAWG-EXT.10/91)

[5] Cotton W.D. & Tody D. 1991, "Binary Table Extension to FITS. A Proposal", Draft 20 Sep 1991 (DAWG-EXT.17/91); see also appendix A of [Ref.4].

[6] Chiappetti, L. 1992, "Report/Log of XAS file i/o tests", February 12 1992 (DAWG-REP.2/92)

[7] Chiappetti, L. 1992, "Status report on XAS software libraries", August 1992, (DAWG-REP.19/92)

[8] Ponz J.D., Thompson R.W. & Muñoz J.R. 1992, "The FITS IMAGE Extension. A proposal" February 7 1992.