XAS

X-ray astronomy Analysis System

User Manual

compiled by L.Chiappetti - IFCTR

an online HTML version of this manual is kept up-to-date at

http://sax.ifctr.mi.cnr.it/Xashelp

This PostScript snapshot obtained on 08−Jan−1997 18:37

Why yet another analysis system ?

Pourquoi PAS ?

Why not ?

[name of J.B.Charcot's polar ship (circa 1905)]

---

i. introduction

The X−ray astronomy Analysis System (XAS) is a software package developed in the framework of the SAX mission (but aiming to mission independence) by a few persons in the Italian CNR institutes since before the ’90s.

An overview of the design of XAS is presented elsewhere and is available on line at `http://sax.ifctr.mi.cnr.it/Sax/xasblurb.html`.

A presentation of XAS has been made at the 5th International Workshop on "*Data Analysis in Astronomy*" (Erice, Oct 27−Nov 3 1996) and a compressed PostScript version of the paper is available.

XAS has been distributed internally to the SAX consortium since 1992 care of ITESRE . With the establishment of SAX SDC the task of distribution to the public has been assigned to the latter institution.

A preliminary version of this HTML XAS user manual has been setup since late 1995. It has now (3 Jan 1997) been upgraded and a "PostScript snapshot" produced. For correction and updates make reference to the online version at `http://sax.ifctr.mi.cnr.it/Xashelp/`.

It is planned to add in a more or less near future also a HTML *XAS Programmers manual* : watch this space.

---

ii. contributors

This XAS HTML manual has been written by Lucio Chiappetti (IFCTR).

The following persons have participated to the development of XAS software :

Lucio Chiappetti (IFCTR)
- overall design and coding
- VMS and Unix Virtual Operating System (VOS) library
- Ultrix, Sun and Alpha ports
- MECS specific modules

Marco Morini (formerly at IFCAI)
- overall design and concepts (1985−1990)

Daniele Dal Fiume (ITESRE)
- overall design
- Unix VOS library
- xasbuild
- Ultrix, HP−UX and Alpha ports
- PDS specific modules

Fabrizio Giambertone (IFCAI)
- fromfits module

Teresa Mineo (IFCAI)
- MECS response matrix

Mauro Orlandini (ITESRE)
- PDS specific modules

Luciano Nicastro(ITESRE)
- PDS specific modules
- IDL xasplot contributed s/w

Andrea Santangelo (IFCAI)
- HPGSPC specific modules

Giacomo Fazio (IFCAI)
- HPGSPC specific modules

Fabrizio Fiore (SDC)
- overall support and instrument team interface
- Sun and Alpha ports

Matteo Guainazzi (SDC)
- overall support and archive configuration

Stefano Signorile (SDC)
- mergewindow module

---

iii. standard disclaimer

This software has been written by staff members of several Institutes of the Italian National Research Council (CNR). Its distribution (and the mantainance of relevant files) is arranged by the SAX Science Data Centre (SDC) of the Italian Space Agency (ASI).

CNR, ASI or any of the authors provide absolutely no warranty of any kind either expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should this program prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event shall CNR or ASI or any of theirs component institutions including IFCTR or SAX SDC be liable to you for damages, including any lost profits, lost monies, or other special, incidental or consequential damages arising out of the use or inability to use (including but not limited to loss of data or data or its analysis being rendered inaccurate or losses sustained by third parties) the program.

(above disclaimer modified from the GNU no warranty statement.)

Ουκ εστι βασιλικη οδος

There is no Royal Way

"*There is no Royal Way [to Geometry]*"

Euclid’s way of saying "RTFM" when presenting his "Elements" to king Ptolemy I (the king was asking whether there was any shortcut to learn Geometry other than reading the several volumes)

(after Proclus, *Commentaria in Euclidium*, *4,2*)

0. XAS Help Table of Content

Preliminaries

Textual help pages

Appendixes and Item lists

---

1. XAS Help top menu

Welcome to XAS, the X−ray Astronomy analysis System devised for SAX but not only...

This page provides an index to an essential cookbook for XAS users, while hints for XAS programmers are supplied elsewhere. It is assumed that a successful XAS installation has already been performed.

For systematic navigation through subsections see also this table of content

The following list of items is in the recommended order of reading corresponding to the typical flow of analysis of SAX data.

- the following two items are of general nature and shall be read carefully by first time users.
- **Using XAS programs**
- **Setting the XAS environment**
- the first step for SAX data reduction is the following
- **Filing a Final Observation Tape (FOT)**
- **Concatenating observations**
- one may have to return at this point several times. One first may accumulate uncorrected and unselected data pertaining to the entire time interval, and to the whole field of view, and later come back applying time windows, based on the inspection of HK and special accumulations.
- **Accumulating spectra, images, time profiles**
- **Accumulating photon lists**
- **Accumulating HK time profiles**
- **Cross−accumulating from photon lists**
- **Special accumulations**
- **Instrument specific corrections**
- **Other corrections**

- **Generation of time windows**
- after one has generated time windows and other special files, one may have to return back to perform accumulations.
  The next items are of general nature and may apply at any step of the reduction.
- **Handling XAS files**
- **Generation of response matrices**
- **Exporting XAS files to other packages**
- **Displaying XAS files graphically**
- **Analysis programs**

---

2. Using XAS programs

The XAS system is a collection of programs (cooperating essentially via the XAS environment which are invoked from the usual Unix shell or VMS DCL prompt.
In this respect XAS is unlike packages with an overarching monitor (like the MIDAS monitor or the IRAF cl) and also unlike monolithic programs like XSPEC.

Each XAS command can be invoked in one of the following way (please note the notation conventions used throughout this document) :

- interactively typing the command name
- passing all arguments on the run string
- passing some arguments only on the run string
- using a command file (like a parameter file)
- in a mixed way

It is also possible to combine a sequence of commands to make a procedure

2.1 Notation conventions

The following notation conventions are adopted in these help files (we hope that they show up adequately using your WWW browser ... these have been tested conservatively using NCSA XMosaic, exclude graphics whenever possible, and trying to use HTML as a true markup language).

Program syntax notation

All program run strings are indicated as standalone lines in what should appear as a large typewriter font like

```
this font
```

The same reference to program names and arguments imbedded in other text appears as `this text`. In particular any item which is not hilighted otherwise (like the program name, but also some arguments) must be typed exactly as it is written (also respecting lower and upper case if indicated). Note however that in most cases XAS is case−insensitive and the first letter(s) of a word are indicated in upper case to indicate the shortest abbreviation allowed (i.e. the compulsory part is in upper case and the optional one in lower case).
Program arguments which are to be typed exactly as they are written, but which have some associated explanation, have an hypertext link associated to their first letter only, like

`this example`

Program arguments which have to be replaced by a value are indicated as a complete hypertext link, pointing to more help, like in

`this example`

However if there is no associate hypertext link, but the argument shall be replaced by some (self explanatory) value, it is indicated emphasized like in

`this` *`example`*

(The same *`emphasization`* is used sometimes to indicate prompts issued by a program (or other output))

In addition, while usually all program arguments are optional, sometimes the first ones are compulsory. A single square bracket indicates where the optional arguments begin (of course this has NOT to be typed !!), like in this example :

```
command comp_arg1 comp_arg2 [ opt_arg1 opt_arg2
```

Moreover, while usually program arguments are strictly positional and must appear

in the order indicated (with placeholders replacing defaulted ones), in some cases the order of the arguments depends on presence of previous arguments. In such cases the run string may be presented more than once for the different cases, or the arguments which is present only conditionally is shown surrounded by braces (which have not to be typed) like in

```
command comp_arg1 comp_arg2 [ opt_arg1 {opt_arg2} opt_arg2_or_3
```

Program dialogue examples

It is PLANNED to include some examples either in text or as screen dumps.

Navigation

Each topic is arranged hierarchically in a set of pages, each one of which is kept as short as possible, ideally a single screenful. These pages will provide usual hypertext links, allowing free navigation and jumping from topic to topic.
However to aid a sequential navigation through the hierarchy at the bottom of each page there is a set of four "text" buttons appearing like the following :

[Previous][Next][Up][Down]

These buttons have been removed from the PostScript hardcopy version of this manual.
Some of them will be hilighted (like the "Down" one in the example above) and clicking on them will lead you respectively to :

- Previous : to the previous help page at the same level, if one exist, otherwise up one level
- Next : to the next help page at the same level, if one exist, otherwise to the next page one level up
- Up : to the page you came from one level up
- Down : to the next page at the level below the one you are in

Of course your browser may hilight with a different colour those which you have already navigated. Try the "Down" button and come back here.

Buttons which appear unhilighted correspond to inactive choices

---

2.1.1 About navigation

O voi che siete in piccioletta barca
desiderosi d'ascoltar, seguiti
dietro al mio legno che cantando varca
tornate a rivedere li vostri liti :
non vi mettete in pelago, che, forse,
perdendo me, rimarreste smarriti
...
Voi altri pochi che drizzaste il collo
per tempo al pan degli angeli, del quale
vivesi qui, ma non sen vien satollo
metter potete ben per l'alto sale
vostro navigio ...

(Par. II,1−6 10−14)

You who are in a tiny boat
wishful to listen, following
after my vessel singing and crossing
go back to meet your shores again :
do not set yourself to sea, because, perhaps,
if you miss me, you will get lost
...
You few other who turned your neck
on time to the angels' bread, of which
one lives here, but gets not satiated
well you can set on the high sea
your craft ...

2.2 Invoking XAS commands

Each XAS program can be invoked in a variety of ways (the simplest of which is the interactive one) all of which may be useful in certain situations. We give below some examples of an hypothetical command performing the rotation of an image of a given angle around a given centre (this command does not exist !), using the standard notation conventions

---

Interactive invocation

All programs may be invoked interactively just by typing the command name, e.g.

```
rotate
```

after which you will be prompted at the terminal, e.g. as follows :

```
rotate
Input image name : pinco
Rotate around (X,Y df 128,128) : 100,120
Rotation angle : 75
Output image name : panco
```

Note that the answer to some questions is a single value (numeric or string), and the answer to some other is an array of values (numeric or string). In all cases it is NOT necessary to enclose strings in quotes.

Passing defaults

To accept the default value proposed by the program, just press return. In the case a question wants more than one value as answer, it is possible to type in only part of the values (and let the last ones default when pressing return), and also to default values at the beginning, using a comma separator for each skipped argument. E.g. to use (128,140) as answer to the question above one can answer as

```
Rotate around (X,Y df 128,128) : ,140
```

Signalling end of file

In reply to any question, one can also press the system−dependent end−of−file sequence (usually control−D on Unix, control−Z on VMS). Depending on the action coded in the program, this will terminate sooner or later, or continue assuming a default value.

Interrupting a program

If you press one of the system−dependent interrupt sequences (e.g. control−C or control−Y) while the program is waiting for input, this will be terminated ungracefully. However if you use the control−C (only) while the program is

executing some long loop, most programs will trap this interrupt correctly and terminate in a graceful way (for accumulation programs this means all data files will be closed with a meaningful header : however the file content may be shorter, since it reflects the conditions at the moment of the break).

Passing all arguments on the run string

```
rotate pinco 100 120 75 panco
Input image name : pinco
Rotate around (X,Y df 128,128) : 100,120
Rotation angle : 75
Output image name : panco
```

Compulsory arguments

Some programs require (always or under some conditions) that some (usually the first) arguments are always given on the run string. This is typically the case of arguments which specify an action which is dispatched to another program (e.g. `accumulate image` or `accumulate spectrum` will invoke separate programs for images and spectra). These cases will be identified in the examples as specified in the notation conventions. All remaining arguments, can be omitted as explained below.

Disabling echo

If you are not interested in the simulated dialogue with the echo of prompts and asnwers, you can disable it using

```
xasset echo OFF
```

If you later run interactively and forget to re−enable echo, you will get "anonymous" prompts of the form

```
WARNING : you are running with  ECHO OFF but ...
..I need input for (an) unspecified parameter(s)
```

Passing some arguments only

If only some arguments are passed on the run string, they will be accepted and echoed as above and you will be prompted for the missing arguments, as in the example :

```
rotate pinco 100 120
Input image name : pinco
Rotate around (X,Y df 128,128) : 100,120
Rotation angle : 75
Output image name : panco
```

Omitting or skipping arguments

Compulsory arguments described above cannot be omitted.
If the last arguments of the runstring are omitted (as in `rotate pinco 100`

`120`), they will be prompted interactively.
If one wants to skip some arguments (which will be prompted interactively) one can use null or blank fields separated by commas, as in the following two examples `rotate ,, 100 120 75 panco` or `rotate pinco ,,, 75 panco`.

Note that in the case of a question requiring more than one value, omitting the first value is equivalent to omitting the entire set of values, which will therefore be prompted. In the example you can do `rotate pinco 100,, 75 panco` to use a range of (100,128) changing the first value, but cannot do `rotate pinco ,,140 75 panco`.

An alternate form of omitting a parameter, is to put a placeholder (you need a single placeholder instead of both delimiting commas). In this case the above examples (with a dot placeholder) become:

```
rotate . 100 120 75 panco
rotate pinco . . 75 panco
rotate pinco 100 . 75 panco
rotate pinco . 140 75 panco
```

where the last example remains "illegal".
Note that both notations wilfully do not allow to take the program defaults unless accepting them explicitly with a carriage return. In order to take silently defaults one must use command files.

Using a command file

The complete syntax for command files is described elsewhere. If a command file is in use, the program will use it instead of interactive input from the terminal. The simulated dialogue of prompts and echoes of the answers read from the command file will be produced, unless echo is disabled.

A command file is established for use with

```
xasset command cfile
```

Compare the following example (the content of the particular command file can be viewed separately)

```
xasset command mycomfil
rotate
Input image name : pinco
Rotate around (X,Y df 128,128) : 100,120
Rotation angle : 75
Output image name : panco
```

A command file remains defined ONLY for the next XAS command invoked, and is reset afterwards, unless one explicitly asks to make it permanent. This is desirable if one wants to run several times in sequence the same program using the same setup.

A command file can be used as a parameter file containing a "standard"

configuration, and one can also use an "all blank" command file to preserve all defaults.

Mixed cases

It is possible (and it is actually the most common usage) to combine the use of command files with runstring arguments. The latter take precedence over the command file, which is used as a sort of setup.
Compare the following examples. The first command file contains the setup for a rotation of 75 degrees around (100,120) and can be applied to any file. The second command file contains instead a default "all blank" setup and is used to accept all program defaults.

```
xasset command mycomfil
rotate tizio . . . caio
Input image name : tizio
Rotate around (X,Y df 128,128) : 100,120
Rotation angle : 75
Output image name : caio
```

```
xasset command alldefault
rotate pallino
Input image name : pallino
Rotate around (X,Y df 128,128) : ,,
Rotation angle : ,,
Output image name : panco
```

Note that if a command file contains "blank" defaults for an argument which does not accept defaults the results are not predictable.
For this reason, but not only, one may construct command files such that some arguments will always be prompted interactively at the terminal, unless explicitly present on the run string.
This is typically used for file names and other string arguments, like in the following example command file

```
xasset command newdefault
rotate pallino
Input image name : pallino
Rotate around (X,Y df 128,128) : ,,
Rotation angle : ,,
Output image name : usertypesnamehere
```

2.3 XAS command files

A command file is used as an alternate source of input instead of the terminal. Its usage is exemplified elsewhere, while its format is described here.

- A command file contains one line for each question (argument(s)) asked by a program.
- This line contains one or more values (individual arguments), AS MANY AS requested by the particular question.
- If a command file is used as a standard setup, each line may be terminated with comments (separated by an exclamation mark), which are useful to document what the arguments are.
- Instead of explicit arguments, one line may start with a terminal escape character (currently an exclamation mark) : in this case the program will ALWAYS prompt interactively the user for this set of arguments (unless they are passed on the run string).
- A command file is established for use (with the NEXT program only unless otherwise requested) with

  ```
  xasset command cfile
  ```

The following examples will clarify what said above.

The example refer to the hypothetic rotation program used in the description of XAS commands.

Example 1

This file `mycomfil.command` corresponds to the run string used in the command invocation example. It is used in the first example of mixed mode.

```
pinco         ! input file name
100,120       ! centre coordinates
75            ! angle
panco         ! output file name
```

Example 2

This file `alldefault.command` contains all default "blank" values and is used in the second example of mixed mode.

```
,,,,,         ! input file name
,,,,,         ! centre coordinates
,,,,,         ! angle
,,,,,         ! output file name
```

Note that commas must be explicitly inserted in each line. It is NOT possible to leave a line blank, since Fortran list−directed i/o will then wrap into the

next line. It is prudent (and harmless) to put MORE commas than required, particularly if one intend to use the command file as a "generic" all−default file for many different programs which may have different order and number of arguments. In this case one usually omits comments.

Example 3

This file `newdefault.command` contains all default "blank" values and is used in the third example of mixed mode. In this case the file names are always asked interactively or derived from the run string, since the program does not provides defaults for them.

```
!           ! input file name
,,,,,       ! centre coordinates
,,,,,       ! angle
!           ! output file name
```

Example 4

This example is not used elsewhere. It is a standard setup file for a rotation of 45 degrees around the default centre, with filenames asked interactively.

```
!           ! input file name
,,,         ! centre coordinates
45          ! angle
!           ! output file name
```

2.4 XAS procedures

XAS is not Yet Another Scripting Language and does not explicitly supports the writing of procedure scripts, but defers this to the facilities provided by the various operating systems (e.g. shell scripts or DCL command files), with which the user is perhaps more familiar.

We give here only a brief tutorial on the possibilities of assembling sequences of XAS commands to perform repetitive tasks (with particular emphasis on the Unix side).

Simple sequences

If one needs to execute often a standard sequence of XAS commands to perform a standard reduction, using always the same standard file names, without need to pass any argument, and without the need of "branching" (no IF statements depending on previous results), it is sufficient

- to write into a file the sequence of XAS commands
- to invoke the file in the current process, without the need to make it an executable script, e.g. in Unix C shell using the `source` command.
- The file header keywords and the XAS environment will take care of exchanging information between one program and the next.

Parametric sequences

By "parametric" sequence we mean a standard sequence of XAS commands used to perform a standard reduction, without the need of "branching" (no IF statements depending on previous results), but which requires some arguments (e.g. file names or numeric parameters).

This can be arranged as an executable shell script with arguments, or even as a "source" file by setting the arguments in specific shell environment variables. The same concepts given above for simple sequences are valid here too, provided one does not need to access the XAS environment at shell level.

Note also that, if one wishes to supply some arguments interactively, one can use a simple sequence containing XAS commands (and invocation of command files) in which some runstring or command file arguments are left "blank" and will therefore be prompted interactively.

Procedure scripts

One will need to write a proper script under two conditions :

- one needs programming facilities like branching (IF statements) or manipulation of shell variables
- one need to interact with XAS environment variables

Programming is left to the facilities of the user's favourite shell and is not of our concern here.
Coming to the interaction of the XAS and system variables one shall instead note what follows.

- XAS environment variables are usually used to contain setup information arranged by the user. There are however cases in which they are set by XAS programs. In particular there is a result variable intended to pass back program results.
- The XAS and system environment shall be regarded as two disjoinct subsets of a larger environment : each "half" can be accessed as "read−write" from either XAS or the shell, and as "read−only" from the other.
- use only xasset to set XAS variables.
- use only shell commands (e.g. set, setenv) to set system environment variables

- In particular if a XAS environment variable is not set, but a system variable of the same name exists, XAS program are able to use its value.
- Conversely in principle all XAS environment variable are accessible to the system shells as system environment variables with uppercase names prefixed by XAS_. E.g. the XAS environment variable ECHO will be known to the system as XAS_ECHO
    - This is immediately true in the case of VMS, where XAS variables are global symbols.
    - This is not straightforward in the case of Unix, where the system environment of a process (e.g. a XAS program, like xasset itself) is not inherited back to the parent process (e.g. the shell). The XAS environment is stored in a file linked to the current session, and must be re−imported in the shell.
    - one way to import back all XAS variables (deleting and resetting the file) is, under csh, to issue the following command (which can be aliased)

      ```
      eval 'importback'
      ```

    - another way to copy back a single XAS variable into another is to intercept the result of a xasset query. Compare e.g. the following csh example (note the use of uppercase) :

      ```
      set temp = 'xasset fotdir \? | grep FOTDIR'
      set fotdirname = $temp[3]
      ```

Other useful Unix tricks

One can, if desired, access directly the content of the file where the environment is stored. This file is in the user login directory, and has a name linked to the current

terminal name (e.g. `ttyp1`) and to the hostname (e.g. `poseidon`), like `ttyp1_poseidon.environment`.

One may then use also a command of the following form to copy back a XAS environment variable into a shell variable :

```
set fotdirname = ‘grep FOTDIR ~/ttyp1_poseidon.environment | cut −f2 −d’=’‘
```

Note also that the environment files remain there after you log off, but are in general cleared when you log in again and issue your first XAS command in the new session. If you wish to continue preserving the last session XAS environment, just `touch` the environment file before issuing ANY XAS command.
A similar effect is obtained if you copy or rename the environment file of a session into another.

3. The XAS environment

The XAS system is unlike packages with an overarching monitor (like the MIDAS monitor or the IRAF cl) and also unlike monolithic programs like XSPEC, but is a collection of cooperating programs invoked from the usual Unix shell or VMS DCL prompt.

These program may communicate between each other in several ways

- via XAS environment variables
- via keywords in XAS file headers
- via communication channels

A XAS environment variable is assigned a value with the xasset command and is deleted using the xasunset command.
The interaction of XAS and system environments is described elsewhere.

One can also consult a list of all known global variables in alphabetic order or by subject.

3.1 XAS environment variables

XAS *environment variables* (or "global variables") are used to retain and exchange information between one XAS program and another, or to set global properties which have to be remembered for a whole XAS session (e.g. default values, or program behaviours).

Global variables are assigned a value either implicitly by some programs, or by using special programs, or, in the most general case, via the xasset program.

The same xasset program can also be used to query the current value of a global variable, or to deassign it (note that for XAS purposes a not existing variable, a variable with a null value, or a variable with all−blank value are equivalent : in all such cases most programs assume a suitable default value). Note however that if a non−XAS global variable with same name exists, its value may be used instead of the missing XAS variable.

XAS environment variables are implemented in VMS as global symbols, and in Unix as environment variables. In both cases their "system" name can be built prefixing their XAS name with the prefix "XAS_". See elsewhere for using XAS variables in procedures (in Unix via importback).

This achieves a separation between the XAS and the system environment, by which system variables are readonly for XAS (they may be used instead of a missing XAS variable, but not set with xasset while XAS variables can be accessed via system facilities using their prefix).

The XAS environment is linked to a session, which in VMS means a *process* (and related subprocesses), while in Unix means a *login session* on a particular machine and a particular pseudo−terminal (in particular one must set the environment separately in each window !). To allow back−inheritance from a child process to the parent, in Unix the XAS environment is saved to a file : the date of creation of this file is compared with the date of login to determine whether it has still to be considered applicable.

3.2 The xasset command

This command can be used to set, query or delete a XAS environment variable

- **Setting a scalar variable**

  `xasset [ variable value`

  This form is used to set the majority of variables, which contain a single string without blanks or commas (a numeric value is also stored as a string).

- **Setting an array variable**

  It is not possible to use `xasset` to set special variables which are a list of string values separated by commas. In such cases (like for some graphics setting) dedicated programs like xasplot must be used to set the variable

- **Setting more than one variable**

  `xasset FROM [ xfile`

  This form can be used to set many variables at once, reading couples "variable value" from the given file. Incidentally this means an environment variable cannot have the name `from`.

- **Querying a value**

  `xasset [ variable ?`

  This form is used to query (show at terminal) the value of a variable (which of course cannot be a single question mark).
  Unix csh users must note that if they are not using the "set noglob" option, they cannot type a single question mark "?", but, to prevent interpretation by the shell they have to escape it "\?"

- **Deleting a variable**

  The convenience command to delete a variable is

  `xasunset [ variable`

  The actual way the software uses to delete a variable is to assign to it a blank value. The above convenience command is provided because, since the shell strips any blank value, one could only delete a variable interactively, issuing a `xasset variable` command, and replying with a carriage return to the prompt for a value.

3.3 XAS files

We give here an user's overview of XAS data files, while programming details will be given elsewhere.

Overview

There are the following basic kind of files used by XAS :

- the FOT telemetry files, i.e. the semi−raw (mostly binary) data in mission−dependent format
- the *reduced data files*, which are described in detail below, and are produced (in mission−independent format) by the user using XAS programs
- ancillary user files, which are usually (tiny) ASCII files like time windows or log files produced by some programs
- the *calibration files* used by the software and distributed with it.

XAS data files

XAS data files are binary files using native internal representation of the specific operating system, and using a mission−independent format. All families of XAS files, listed below, share a common organization consisting of :

- a mini−header in front of the file
- a data area with either image (n rows of m pixels) or tabular (n rows with p logical columns of different widths) organization
- an *extensible header* at the end of the file, which is organized as a sequence of keywords

The following are the types of XAS files :

- *images* (of type `.image`) containing counts or other parameters as a function of spatial position
- *pseudo−images* (equally of type `.image`) containing counts or other parameters as a function of any two (non−spatial) coordinates
- *response matrices* (of type `.matrix`) containing instrumental response as a function of energy and channel. They are also in image format. An associated histogram (of type `.histo`) carries the reference energy grid, in form of an unidimensional image.

- all the above files are in *image format*, there are as many records as image rows, and each row contains the right number of REAL*4 pixels (the software can support also INTEGER*2 images but their use is discouraged).
  all files which follows are instead in *tabular format*.

- *spectra* (of type `.spectrum`) containing a count/s or count histogram as a function of energy or of another "channelized" parameter. These files have (usually) four (depth one) columns :
    1. the lower channel boundary (in channels or in keV)
    2. the upper channel boundary (in channels or in keV)
    3. the data value (in cts/s or cts)
    4. its error (in cts/s or cts)
- *time profiles* (of type `.time`) containing any quantity as a function of time. These files may have up to 7 columns, although usually not all columns are present :
    1. the start time of the bin (in double precision, elapsed seconds since midnight of the first day of the observation). This is present virtually in all cases.
    2. the bin size (present only if not constant)
    3. the deadtime (present in a few cases, currently not yet used)
    4. the data value (in cts/s, cts, temperature, voltage or other units), which may also have a depth greater than one
    5. its error
    6. an optional second data column
    7. and its optional error
- *photon lists* (of type `.photon`) containing a list of events. For each events one may have a number of columns containing information like X, Y, energy, time etc. according to the particular case.

- generic tabular files are also possible

Where do XAS data files reside ?

XAS uses its own system−independent file naming convention (VOS names), which is translated into system dependent names by appropriate routines. This is in practice not a concern for the general user, and in particular for the Unix user, since VOS names virtually (*sic!*) coincide with Unix names. A name may be of the form :

```
/dir/dir/.../name.type
dir/dir/.../name.type
name.type
name
```

The type (usually one of the standard ones given above) is often omitted and assigned automatically by the programs, or by context.

Also the path (either absolute or relative) is usually omitted, and is instead built by programs.

It is important to remember that, *irrespective of what is the current working directory*, XAS programs will look for files (without an absolute path) or create them in specific directories previously decided by the user, and namely

- in a "FOT directory" for FOT telemetry files
- in a "data directory" for all reduced data products

- in a "print directory" for ancillary output (log files etc.)
- in a system (or user−specified) directory for calibration files

Relative paths are relative to this particular directory !

The full path of each of these directories is constructed on the basis of several environment variables, in order to allow each user the maximum flexibility. We give here examples for the "data directory", while the cases of the "FOT" and "print" directory are similar, replacing fotdir or printdir to datadir, and optionally replacing fotorder or printorder to order (if this is not done one obtains the same hierarchy).

The full path for the "data directory" is constructed of up to 5 parts (each part being in turn a path), e.g.

```
/part1/part2/part3/part4/part5
```

`part1` is always a logical root, the value of rootdir
The other four parts (of which three can be omitted) are controlled by the four variables datadir, target, date, instrument, and occur in the order specified by variable order

Compare the following examples :

Flat arrangement

```
xasset rootdir /home/me
xasset datadir mydata
```

Default order assumed, data files are in `/home/me/mydata`

Complex arrangement

```
xasset rootdir /home/me
xasset datadir mydata
xasset target Crab
xasset date Sep06
xasset instrument mecs
xasset order ctdi
```

Data files are in `/home/me/mydata/Crab/Sep06/mecs`

Changing order

If one instead does e.g. `xasset order cidt` data files go in
`/home/me/mydata/mecs/Sep06/Crab`
All permutations are possible.

One can also omit a part, e.g. if one does `xasset order cd` data files go in
`/home/me/mydata/Sep06`
If one `xasunset order` the default path `/home/me/mydata` is restored.

Using different orders

Let us assume that one wants to do a standard reduction for a particular object, or observation, or dataset called `pinco`, then the same reduction for another dataset called `panco` and finally combine the results :

```
xasset rootdir /home/me
xasset datadir mydata
xasset target pinco
xasset order ct
```

Files for first data set are created as `/home/me/mydata/pinco/name.type`

```
xasset target panco
```

Files for other data set are created as `/home/me/mydata/panco/name.type`

```
xasunset order
```

The system will look for data files in `/home/me/mydata`, therefore one may refer to the files of either data sets with relative paths like `pinco/name.type` or `panco/name.type` and call merged files just as `name.type`. The same result will occur doing `xasunset target`

3.4 XAS header keywords

The header of a XAS file is composed by a sequence of keywords. Programs store in keywords basic parameters of a file, or details of the data processing done. The content of a file header can be accessed programmatically by later programs (in this way constituting a way of exchange of information), and viewed or edited by the user,

---

Each keyword has a *name*, a *type*, a *length* and (a) *value*(s).

- the name is a possibly meaningful, case−insensitive word, no longer than 8 characters for FITS compatibility. Not all FITS mandatory keywords are used in XAS files (when necessary they can be generated by conversion programs) but if they are the same names are used.
- the type is one of character, integer, real or double precision as detailed elsewhere
- the length of a character keyword is the string length. For FITS compatibility this is limited to 68 characters (although in principle could be longer).
- the length of a numeric keyword is either one for scalar keywords, or the number of elements for array keywords. The total length in bytes cannot exceed 246.
- the value is either a string, or an array of one or more numeric values

---

Keywords in the file usually appear in the following order (which is however irrelevant since entire headers are read in memory at once) :

- subset of FITS mandatory or recommended keywords. In particular the `NAXIS2` keyword is the number of bins or photons in tabular files (while `NAXIS1` and `NAXIS2` are the X and Y sizes of an image). Also all the binary table keywords like `TFIELDS`, `TTYPEn` etc. are supported.
- the original file name without path and type is in `FILENAME`
- the `OBJECT` and `OBSERVER` keywords are filled deriving the information from FOT tape directory.
- there is a `SATELLIT` keywords instead of the FITS `TELESCOP`, the (SAX) instrument name is in `INSTRUME` and the sub−unit in `DETNAM`
- then accumulation specific and instrument specific keywords follow.
- the last keywords are the `HISTORY` (i.e. the runstring of the command which created the file) and optionally its `PARENT` files and some `COMMENT`s.

When a program manipulates a file in place, it may alter some of the keywords, and add some new ones. In particular it will add a new `HISTORY` section.
If a program reads from a file and produces a new one, it may copy part or all of the original file header and then add its own `HISTORY` section.

Only the `HISTORY`, `COMMENT` and `PARENT` keywords may appear more than once in a header (they are "*duplicatable*").

If an `HISTORY` keyword is longer than the maximum allowed, it is continued on the next line, prefixing the value with a plus sign.

3.5 Communication channels

Communication channels are used to exchange information between two programs which run at the same time (unlike environment variables and keywords used to exchange information between programs running one after the other).

In practice communication channels are reserved for client−server communication, used seldom, mainly in the area of graphics

Communication channels are implemented in Unix as named pipes and in VMS as mailboxes, and their usage is normally transparent to the user.

4. SAX Final Observation Tapes

Final Observation Tapes (FOTs, the name being mutuated from the EXOSAT mission) are the way by which SAX observation data are made available to observers by SAX−SDC.
The layout of FOT tapes is described elsewhere.

The first step of SAX data reduction (and the only one truly mission−dependent) is the *filing of tape data to disk.* This is done

- setting the appropriate environment variables
- mounting a tape in a tape drive
- invoking the `fotfile` command

After this one will find the requested subset of the following telemetry files on disk.

4.1 SAX Final Observation Tape Layout

A FOT is physically either a DAT 4mm cassette, or an Hexabyte 8mm cassette, or a 1.2 inch tape reel (or even more than one in case of multivolume FOTs).

The FOT is a sequence of tape files, each one separated by a tape mark, and terminated by a double tape mark.

Physically each file is in "fixed blocked" format, i.e. all tape blocks are of the same length (in a file; this length may differ from one file to another) with the exception of the last block which may be shorter. All block lengths are multiple of a common logical record length.
The filing process is essentially the unblocking and the reconstruction of disk files with the original record lengths.

Top level FOT layout

The logical structure of a FOT is a sequence of files and file groups ordered in a topdown hierarchy as follows :

- one tape directory file
- a group of files relevant to the entire observing period
- an instrument dataset for one SAX instrument
- another instrument dataset
- and another ... etc. etc.

  The instrument order is typically by increasing energy (LECS MECS HPGSPC PDS) or number (WFC1 WC2). NFI and WFC data are never mixed on the same FOT.

- **Observing period related file group**

  - one ephemeris file
  - one reconstructed attitude file
  - one OBT to UTC conversion file
  - zero or more spacecraft HouseKeeping (telemetry) files

  Instrument dataset

  - one instrument dataset directory (instdir)
  - zero or more observation filesets for the ingoing slew
  - one or more observation filesets for the target pointing
  - zero or more observation filesets for the outgoing slew

Observation fileset

- one observation directory (obsdir)
- one experiment configuration file (expconf)
- one (or none) instrument HouseKeeping (telemetry) file
- one or more instrument Engineering (telemetry) file
- zero, one or more instrument science (telemetry) file

4.2 Environment setting for FOT filing

Before invoking the `fotfile` command one shall notify the system on which disk directory the files have to be put.

This is done setting (at least) the following two environment variables :

- `xasset rootdir abspath`
- `xasset fotdir  path`

4.3 The fotfile command

Provided you have set the appropriate environment variables and loaded your FOT on a tape drive, you can invoke the command

```
fotfile [ tape instrument inslewobs targetobs outslewobs datatypes
```

In practice it is suggested that :

- you first file the tape directory only with command

  ```
  fotfile tape XX 0 0 0 A
  ```

- you look at the `tapedir` file
- you reissue the `fotfile` command filing the wished data for one instrument
- you reduce the data for such instrument
- eventually you change `fotdir`
- and repeat for another instrument

IT IS PLANNED to add an example session.
See elsewhere about the naming convention for telemetry files

4.4 SAX telemetry files on disk

As result of FOT filing the currently defined fotdir will contain the requested subset of the following files, where `typ` is the file type

---

File naming convention

- `saxfot.typ`

  for any common file independent of the instrument (like tapedir, ephemeris, attitude, etc.) e.g.

  `saxfot.tapedir`

- `ee.typ`

  for any instrument file independent of the observation (like the instdir e.g. for instrument `ee=w1` one has

  `w1.instdir`

- `oeennn.typ`

  for any instrument file for observing period type `o` (`=i|n|f` for initial slew, normal pointing or final slew), instrument `ee` and observation `nnn`, e.g. for observation 7 of normal pointing for MECS one may have

  `nme007.meexconf`

  `nme007.m3dir002`

  etc.

File hierarchy and naming example

We report here the above naming scheme along with the FOT layout hierarchy for an example of a single instrument.

- `saxfot.tapedir`
- a group of files relevant to the entire observing period
- instrument dataset for MECS

  Observing period related file group

  - `saxfot.ephemeris`

- saxfot.attitude
- saxfot.obt_utc
- saxfot.xxhkd000 (zero or more optional) HouseKeeping files for subsystem xx

Instrument dataset

- me.instdir
- zero or more observation filesets for the ingoing slew
- one or more observation filesets for the target pointing
- zero or more observation filesets for the outgoing slew

Observation fileset

Here shown for pointing type o=n and observation number nnn=1

- nme001.obsdir
- nme001.meexconf
- nme001.mehkd000
- nme001.m1eng000, nme001.m2eng000 and nme001.m3eng000
- nme001.m1dir002, nme001.m2dir002 and nme001.m3dir002

File content and layout

The detailed layout of all files is contained in documentation produced by Telespazio and available at the SAX−SDC. We give here basic information on the main file types.

- *tape directory*
  An ASCII file with one header record with target and observer name, observation date and other ancillary information, plus one record for each tape file, describing the blocking parameters (record length, number of records etc.) etc.
- *ephemeris file*
  An ASCII file with the orbital position and velocity at 1 sec resolution
- *attitude file*
  An ASCII file with the reconstructed attitude at 0.5 sec resolution
- *OBT UTC correlation file*
  An ASCII file with sporadic information for the correlation between On Board Time and Universal Time
- *spacecraft HK*
  These files are raw binary telemetry and are not normally present nor used by the general observer
- *instrument directory*
  An ASCII file which contains one record per observation, with the start and end time, the target name and a reference pointing

- *observation directory*
  An ASCII file listing the number of records and record length for each of the following data files present in the respective observation
- *experiment configuration file*
  A self−explanatory ASCII file listing the setup of all commandable parameters of an instrument during the respective observation
- *instrument HK*
  One binary telemetry file containing periodic readouts (with instrument dependent periodicities, e.g. 64, 32, 16 and 2 s) of parameters like voltages, temperatures, safety ratemeters etc.
  This file is normally always present.
- *engineering files*
  Binary telemetry files (in some cases more than one, e.g. by instrument sub−unit) containing periodic readouts (usually at 1 sec periodicity) of scientifically relevant information (like ratemeters).
  These files are normally present only when instrument data processing is enabled.
- *science files*
  Binary telemetry files containing the scientific telemetry. More files may be present according to the number of instrument sub−units and on the operating modes. May be at variable rate (DIRect mode event data) or fixed rate (INDirect modes).
  These files are normally present only when instrument data processing is enabled.

5. Concatenating observations

A FOT contains data from one OP (*Observing Period*)

- An OP is defined essentially as one continuous time interval with constant pointing, or a slew manouvre. Non−classifiable intervals are named *contingency OPs*)

A FOT usually contain a single normal OP plus the preceding and following slew OPs if applicable.

In turn each OP can be divided in a number of finer time intervals called *observations.*

- An observation boundary (change) occurs whenever the instrument configuration is modified by a telecommand (or of course at an OP boundary)

The names of observing period and observations used for SAX are mutuated from EXOSAT. The equivalent names for XMM have *later* been changed to observation and exposure, respectively.

Of course only observations in the same or compatible configuration can be concatenated together. Before proceeding to concatenation, one shall check the instrument configuration. The following procedure is recommended.

- setting the appropriate environment variables
- check experiment configuration
- invoke the `concatenate` command

5.1 Environment setting for accumulations

Before invoking the check_expconf or concatenate or command one shall notify the system on which disk directory FOT files reside, and what SAX instrument has to be analysed.

This is done setting (at least) the following environment variables (the first two shall already have been set before FOT filing) :

- `xasset rootdir abspath`
- `xasset fotdir  path`
- `xasset printdir  path`
- `xasset instrument instrument`

Before invoking any of the `accumulation` commands one shall notify the system on which disk directory FOT files reside, what SAX instrument has to be analysed, and where XAS has to create or look for data files.
In addition one may want to set instrument specific corrections or to establish time windows

---

This is done setting (at least) the following environment variables (the first four shall already have been set in advance) :

- `xasset rootdir abspath`
- `xasset fotdir path`
- `xasset printdir path`
- `xasset instrument instrument`

  Consult the separate tutorial on the location of XAS data files.

- `xasset datadir path`

- `[xasset target anystring]`
- `[xasset date anystring]`
- `[xasset order order]`

  Consult the separate tutorial on instrument specific corrections.

- `[xasset correction Enabled]`
- `[xasset gainhistory gainfile]`
- `[xasset selectfile selfile]`
- `[xasset selectmethod method]`

- [xasset timewindow wfile]

5.2 The check_expconf command

In order to verify the compatibility of each observation with the previous and the next one you shall inspect the Experiment Configuration Files. These are ASCII FOT files (of type *ee*`exconf` for instrument *ee*) which can be printed or viewed easily. You can get an impression of the changes between one observation and the next also using a visual difference tool like DEC's `dxdiff`

However, since the Experiment Configuration Files may contain a plethoric list of commandable items, which are normally kept at their nominal value, a command is provided which produces a summary of the relevant parameters, together with a coding of helpful hints.
Relevant parameters are defined by hardware teams, according to these guidelines.

The summary can be in an ASCII file or in a black&white or colour Postscript file (in current printdir) or be "plotted" in colour on a graphics window mantained by an X−window graphics server.

Provided you have set the appropriate environment variables you can invoke this command as

---

`check_expconf [ outtype` *`filename`*

Note that if `outtype` is `Graph` then `filename` is not an output file name but a graphic X−window server instance of the form `xw1`, `xw2`, etc.

All output files are arranged similarly as follows :

- there is a FOT identification heading with e.g. the source and observer's name.
- there are as many columns as overall observations in the observing period (this means also slewsare always included). The observation number is identified in the first row.
- the second row indicates whether the observation is a slew (`in` or `out`) or `no`. Normally one uses only non−slew observations.
- Two more rows indicate the start and end time of the observation (the day is found in the FOT heading line)
- A number of rows follow, corresponding to those parameters in the Experiment Configuration Files (named in the first column) which are deemed relevant. Even if a subset is present, they occur in the same order as in the file, therefore the typical values to look at are :
    - instrument (subunit) HV status (ON|OFF)
    - instrument HV value (if relevant)
    - other thresholds etc. (if relevant)

    - the mode and POP status (typically use only `POPSTAT=ENABLE` observations)
- Three lines at the bottom indicate the presence of "spacecraft−provided" instrument HK, of ratemeter HK, and of scientific data (ev. divided by subunit).

A particular coding hilights cases of interest for the observer

- "good" values, i.e. in nominal state
- "bad" values, i.e. slew status, HV off, POP not enabled, data not present. Usually these observations will not be used for scientific analysis.
- "suspicious" values, e.g. data not present for all subunits. Usually these observations can be used with some caution.
- "invalid" (i.e. meaningless) values (e.g. HV settings are meaningless if HV is off) The actual value shall be in the Experiment Configuration File, but is not reported here as it might be confusing for the observer. The word `invalid` is reported instead
- "missing" values (the Experiment Configuration File could be missing for so−called "contingency" observations
- values which have changed since previous observations. If both observations are good, some cautionary check must be applied before concatenating.

We provide below example files as HTML hyperlinks (or also in printed form) , together with some useful hints on their interpretation

ASCII file

ASCII files may awkward to print or view if there are many observations, since they are not paginated, and all observation occur on a single very long row. A single flag character hilights the following cases:

- none or blank for nominal values
- minus (−) for values changed since previous observation
- asterisk (*) for non−nominal values
- hash (#) for non−nominal AND changed values

B&W Postscript file

These are in landscape orientation, and are paginated to have 15 or less observations per page. Hilight on grayscale files is indicated as follows

- black normal font on white background for nominal values
- bold font in reverse on black background for non−nominal values
- white is reversed to gray for values changed since previous observation
- invalid values are dimmed (in gray over white)
- missing values are in bold, italic, white over gray
- suspicious values are in bold, italic, black over gray

Colour Postscript file

For colour Postscript hilighting occurs as follows

- nominal values are green
- non−nominal values are red and in a bold font
- a yellow background indicates change since previous observation
- invalid values are dimmed (in gray over white)
- missing values are in bold, italic, magenta
- suspicious values are in bold, italic, yellow over gray

Graphics window (GIF dump)

This is useful for quick look (e.g. if no printers available). As many rows and columns as required are always squeezed in the existing window, therefore they may appear quite crowded if there are many observations (use a larger window if possible). Hilighting uses a colour coding.

- green for nominal values
- red for non−nominal values
- changes since previous observation currently not indicated
- blue for invalid values
- magenta for missing values
- yellow for suspicious values

We provide here the examples of output for the printed version of the manual.

ASCII file example

MECS configuration summary for FOT n.000331 produced on 1996-Jul-29 - target Cyg X-2
observed on 1996-Jul-23 for SAX TEAM CY G X-2 (OP_683)

Obs.No.	I01	N01	N02	N03	N04	N05	N06	N07
Slew?	in	no	no	no	no	no	no	no
StartEnd	0006-0019	0019-0054	0054-0221	0221-0237	0237-0405	0405-0421	0421-0547	0547-0605
INSTRACT	ON	ON	ON	ON	ON	ON	ON	ON
M1DU	*OFF	*OFF	-ON	#OFF	-ON	#OFF	-ON	#OFF
M3DU	*OFF	*OFF	-ON	#OFF	-ON	#OFF	-ON	#OFF
ALLDUS	*OFF	*OFF	-ON	#OFF	-ON	#OFF	-ON	#OFF
LED	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
M1PMTHV	invalid	-invalid	- 838.000	invalid	- 838.000	invalid	- 838.000	invalid
M2PMTHV	invalid	-invalid	- 838.000	invalid	- 838.000	invalid	- 838.000	invalid
M3PMTHV	invalid	-invalid	- 838.000	invalid	- 838.000	invalid	- 838.000	invalid
M1WINHV	invalid	-invalid	- 6500.00	invalid	- 6500.00	invalid	- 6500.00	invalid
M2WINHV	invalid	-invalid	- 6500.00	invalid	- 6500.00	invalid	- 6500.00	invalid
M3WINHV	invalid	-invalid	- 6500.00	invalid	- 6500.00	invalid	- 6500.00	invalid
M1DRIFHV	invalid	-invalid	- 6000.00	invalid	- 6000.00	invalid	- 6000.00	invalid
M2DRIFHV	invalid	-invalid	- 6000.00	invalid	- 6000.00	invalid	- 6000.00	invalid
M3DRIFHV	invalid	-invalid	- 6000.00	invalid	- 6000.00	invalid	- 6000.00	invalid
HWOPMODE	*invalid	#invalid	-NORMAL	#invalid	-NORMAL	#invalid	-NORMAL	#invalid
ITOPMODE	*invalid	#invalid	-DIR2	#invalid	-DIR2	#invalid	-DIR2	#invalid
POPID	*invalid	#invalid	- 1	#invalid	- 1	#invalid	- 1	#invalid
POPSTAT	*DISABLE	*DISABLE	-ENABLE	#DISABLE	-ENABLE	#DISABLE	-ENABLE	#DISABLE
HK ?	X	X	X	X	X	X	X	X
RM ?			123		123		123	
Sci?			123		123		123	

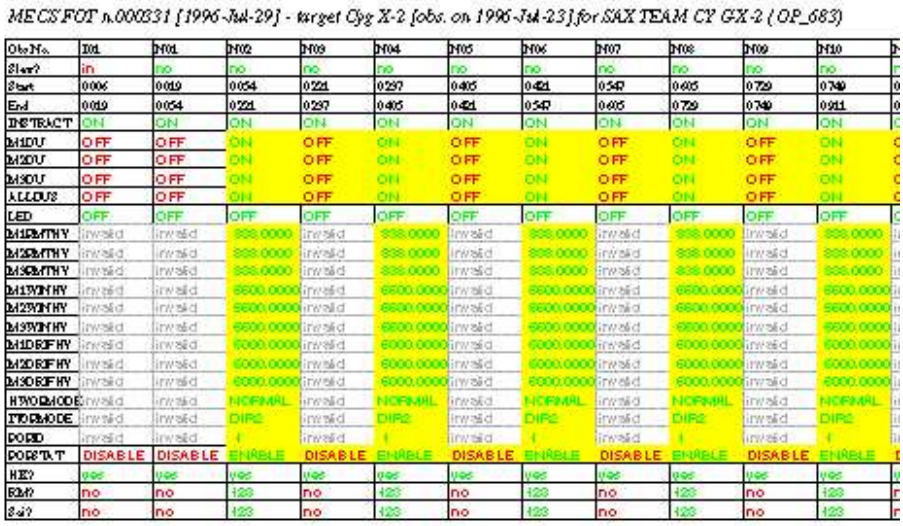
B&W Postscript example

MECS FOT n.000331 [1996-Jul-29] - target Cyg X-2 [obs. on 1996-Jul-23] for SAX TEAM CY GX-2 (OP_683)

Obs.No.	I01	N01	N02	N03	N04	N05	N06	N07	N08	N09	N10
Slew?	in	no	no	no	no	no	no	no	no	no	no
Start	0006	0019	0054	0221	0237	0405	0421	0547	0605	0729	0749
End	0019	0054	0221	0237	0405	0421	0547	0605	0729	0749	0911
INSTRACT	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
M1DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON
M2DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON
M3DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON
ALLDUS	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON
LED	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
M1PMTHV	invalid	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000
M2PMTHV	invalid	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000
M3PMTHV	invalid	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000	invalid	-838.0000
M1WINHV	invalid	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000
M2WINHV	invalid	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000
M3WINHV	invalid	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000	invalid	-6500.0000
M1DRIFHV	invalid	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000
M2DRIFHV	invalid	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000
M3DRIFHV	invalid	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000	invalid	-6000.0000
HWOPMODE	invalid	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL
ITOPMODE	invalid	invalid	DIR2	invalid	DIR2	invalid	DIR2	invalid	DIR2	invalid	DIR2
POPID	invalid	invalid	1	invalid	1	invalid	1	invalid	1	invalid	1
POPSTAT	DISABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE
HK?	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
RM?	no	no	123	no	123	no	123	no	123	no	123
Sci?	no	no	123	no	123	no	123	no	123	no	123

The next figures are appropriate for colour printing only.

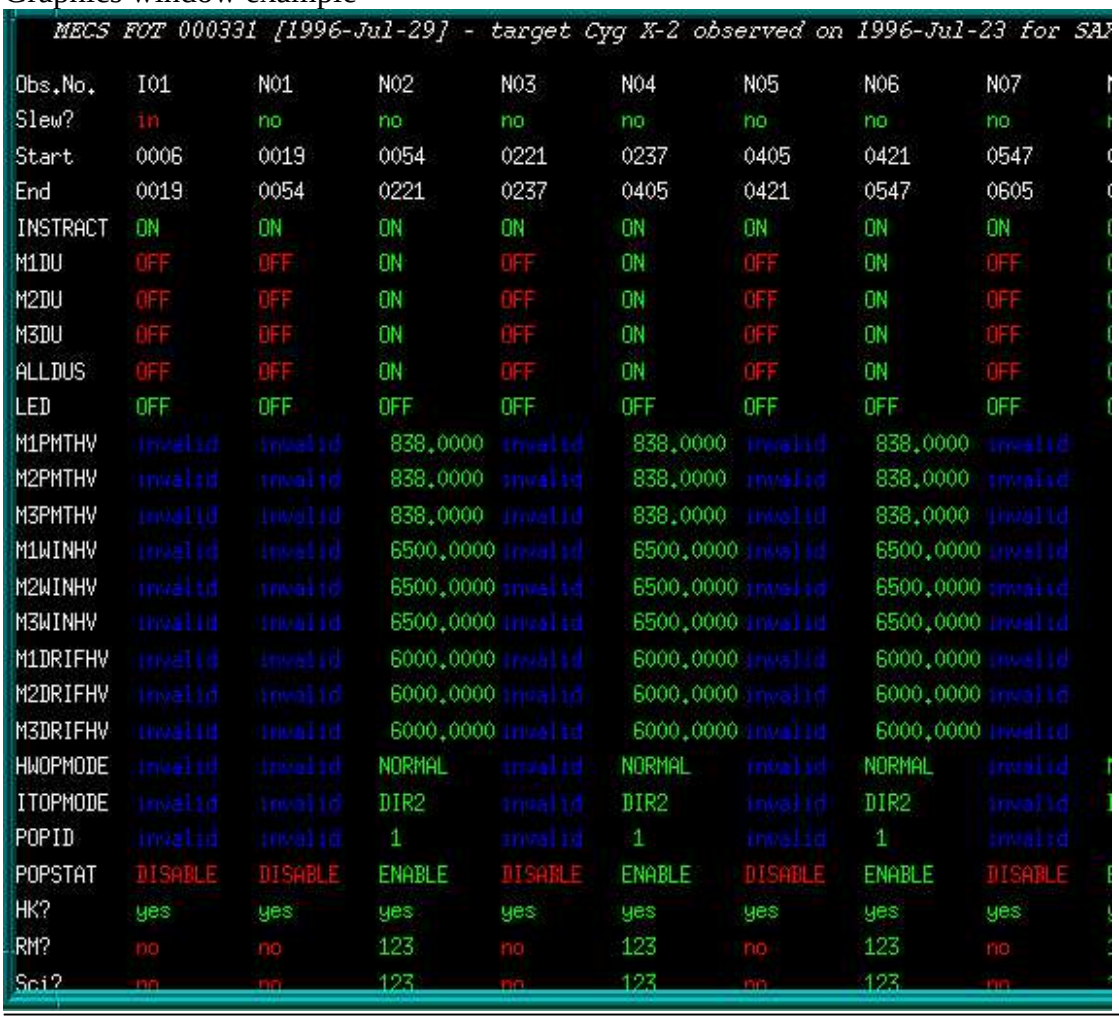
Colour Postscript example



MECS FOT n.000331 [1996-Jul-29] - target Cyg X-2 [obs. on 1996-Jul-23] for SAX TEAM CY GX-2 (OP_683)

Obs.No.	I01	N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N
Slew?	in	no	no	no	no	no	no	no	no	no	no	n
Start	0006	0019	0054	0221	0237	0405	0421	0547	0605	0729	0749	0
End	0019	0054	0221	0237	0405	0421	0547	0605	0729	0749	0911	0
INSTRACT	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	O
M1DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	O
M2DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	O
M3DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	O
ALLDUS	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	O
LED	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	O
M1PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	in
M2PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	in
M3PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	838.0000	in
M1WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	in
M2WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	in
M3WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	in
M1DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	in
M2DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	in
M3DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	in
HWOPMODE	invalid	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	in
ITOPMODE	invalid	invalid	DIR2	invalid	DIR2	invalid	DIR2	invalid	DIR2	invalid	DIR2	in
POPID	invalid	invalid	1	invalid	1	invalid	1	invalid	1	invalid	1	in
POPSTAT	DISABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	D
HK?	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	y
RM?	no	no	123	no	123	no	123	no	123	no	123	n
Sci?	no	no	123	no	123	no	123	no	123	no	123	n

Graphics window example



MECS FOT 000331 [1996-Jul-29] - target Cyg X-2 observed on 1996-Jul-23 for SA

Obs.No.	I01	N01	N02	N03	N04	N05	N06	N07	N
Slew?	in	no	no	no	no	no	no	no	n
Start	0006	0019	0054	0221	0237	0405	0421	0547	0
End	0019	0054	0221	0237	0405	0421	0547	0605	0
INSTRACT	ON	ON	ON	ON	ON	ON	ON	ON	O
M1DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	O
M2DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	O
M3DU	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	O
ALLDUS	OFF	OFF	ON	OFF	ON	OFF	ON	OFF	O
LED	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	O
M1PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	in
M2PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	in
M3PMTHV	invalid	invalid	838.0000	invalid	838.0000	invalid	838.0000	invalid	in
M1WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	in
M2WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	in
M3WINHV	invalid	invalid	6500.0000	invalid	6500.0000	invalid	6500.0000	invalid	in
M1DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	in
M2DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	in
M3DRIFHV	invalid	invalid	6000.0000	invalid	6000.0000	invalid	6000.0000	invalid	in
HWOPMODE	invalid	invalid	NORMAL	invalid	NORMAL	invalid	NORMAL	invalid	in
ITOPMODE	invalid	invalid	DIR2	invalid	DIR2	invalid	DIR2	invalid	in
POPID	invalid	invalid	1	invalid	1	invalid	1	invalid	in
POPSTAT	DISABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	ENABLE	DISABLE	D
HK?	yes	yes	yes	yes	yes	yes	yes	yes	y
RM?	no	no	123	no	123	no	123	no	n
Sci?	no	no	123	no	123	no	123	no	n

5.3 The concatenate command

If there are no non−routine configuration changes across observations, you can in general *concatenate* together

- for HK accumulations all observations for which HK data are present (since these data are produced "via spacecraft" they may be present even if POP is not ENABLEd.
- for ratemeter and science accumulations, all observations for which data is present for a particular sub−unit (be careful if data for only one sub−unit is present if all are expected). These data shall be there if POP is ENABLEd only, but sometimes a packet may be mis−attributed to an adjacent observation.
- thus if you want to change the type of accumulation, you might have to reissue a different concatenation command.

As a guide to selection of observations to be concatenated inspection of the summaries produced by the check_expconf program may be useful. Once you have decided, make sure you have properly set up environment variables and invoke the command

```
concatenate [ obsrange
```

Please note that the syntax for obsrange is mandatory, use "plus" as concatenation operator and do not use blanks as separators !

6. Accumulating spectra, images, time profiles

Accumulation is the process of generating data structures (like images, spectra, time profiles and photon lists) in a mission−independent format starting from telemetry files.

As prerequisites one shall ensure that

- FOT tape data have been filed to disk
- setting of the appropriate environment variables is complete
- a valid observation chain has been defined.

One can accumulate directly "reduced" data structures of immediate usage (like spectra, images or light curves) or pass thru intermediate photon lists. The latter are described in another section.

- accumulating spectra
- accumulating images
- accumulating time profiles
- accumulating photon lists
- accumulating HK time profiles
- generating response matrices

6.1 Accumulating spectra

A *spectrum* is defined as an histogram of counts distributed in channels (or bins) of a given quantity. This quantity defaults to digitized energy (i.e. PHA), but can also be any other information present for photon events in the given observing mode.

One can control the behaviour of the accumulation program in several ways.

- **accumulating a default PHA spectrum (normal form)**

  the simplest form of issuing the command is

  ```
  accumulate spectrum [ sfile {packet} xrange xbin  {range range ...}
  ```

- **accumulating a default PHA spectrum (short form)**

  in the case one wishes to accumulate many spectra, one can default the kind of accumulation setting the context to spectra, after which one can issue an arbitrary number of accumulation commands in the short form (if one wishes to interleave with another kind of accumulation it is sufficient to use the the normal form

  ```
  xasset context s[pectrum]
  accumulate sfile [{packet} xrange xbin  {range range ...}
  ```

- **accumulating a single spectrum of another quantity**

  in the case one wishes to accumulate spectra for a quantity other than PHA (e.g. a BURSTLENGTH spectrum) one may set this in the environment (and reset it to PHA or to blank − which defaults to PHA − afterwards !). One can use either the normal or the short form.

  ```
  xasset xquantity quantity
  accumulate {spectrum} sfile {packet} xrange xbin  {range range ...}
  ```

- **accumulating spectra of other quantities explicitly**

  in the case one wishes to switch between interleaved accumulations of different quantities, one may set the "pseudospectra" accumulation mode, in this case one is prompted to supply the name of the quantity to be accumulated

  ```
  xasset accummode pseudo
  accumulate {spectrum} sfile {packet} quantity xrange xbin  {range range ...}
  ```

  to revert to the usual behaviour, reset the accumulation mode to "normal" (or to blank which defaults to "normal").

6.2 Accumulating images

An *image* is defined as any 2−d distribution of counts in pixels of two given quantities. These quantities default to X and Y positions, but can also be any other information present for photon events in the given observing mode.

One can control the behaviour of the accumulation program in several ways.

- **accumulating a default XY image (normal form)**

  the simplest form of issuing the command is

  ```
  accumulate image [ ifile {packet} xrange xbin yrange ybin  {range range ...}
  ```

- **accumulating a default XY image (short form)**

  in the case one wishes to accumulate many images, one can default the kind of accumulation setting the context to images, after which one can issue an arbitrary number of accumulation commands in the short form (if one wishes to interleave with another kind of accumulation it is sufficient to use the the normal form

  ```
  xasset context i[mages]
  accumulate ifile [{packet} xrange xbin yrange ybin  {range range ...}
  ```

- **accumulating a single "pseudoimage" of other quantities**

  in the case one wishes to accumulate a "pseudoimage" for quantities other than X,Y (e.g. a PHA−BURSTLENGTH distribution) one may set this in the environment (and reset it to X,Y or to blank − which defaults to X,Y − afterwards !). One can use either the normal or the short form.

  ```
  xasset xquantity xquantity
  xasset yquantity yquantity
  accumulate {image} ifile {packet} xrange xbin yrange ybin  {range range ...}
  ```

- **accumulating (pseudo)images of other quantities explicitly**

  in the case one wishes to switch between interleaved accumulations of different quantities, one may set the "pseudoimage" accumulation mode, in this case one is prompted to supply the name of the quantity to be accumulated

  ```
  xasset accummode pseudo
  accumulate {image} ifile {packet} xquantity yquantity xrange xbin yrange ybin  {range
  ```

  to revert to the usual behaviour, reset the accumulation mode to "normal" (or to blank which defaults to "normal").

6.3 Accumulating time profiles

A *time profile* is defined as a light curve, an histogram of counts, or of any other quantity, distributed in temporal bins of a given duration.

One can control the behaviour of the accumulation program in several ways.

- **accumulating a time profile (normal form)**

  the simplest form of issuing the command is

  ```
  accumulate time [ tfile {packet} trange tbin  {range range ...}
  ```

- **accumulating a time profile (short form)**

  in the case one wishes to accumulate many light curves, one can default the kind of accumulation setting the context to time, after which one can issue an arbitrary number of accumulation commands in the short form (if one wishes to interleave with another kind of accumulation it is sufficient to use the the normal form

  ```
  xasset  context t[ime]
  accumulate tfile [{packet} trange tbin  {range range ...}
  ```

- There is no way nor sense in using a quantity different from TIME or to use the pseudo accumulation mode for time profiles, the environment variables xquantity, yquantity or accummode have no effect.

7. Accumulating photon lists

Accumulation of a photon list is similar to other accumulations but produces an event list in mission−independent format which can be later the starting point of other cross−accumulations or subject to manipulations

---

As prerequisites one shall ensure that

- FOT tape data have been filed to disk
- setting of the appropriate environment variables is complete
- a valid observation chain has been defined.

A photon list can contain a subset of the quantities present for each of the original events. Moreover events can also be selected by their value being in a given range. There are a few ways of controlling the behaviour of the accumulation program.

- **accumulating a photon list (normal form)**

  the simplest form of issuing the command is

  ```
  accumulate photon [ pfile {packet} {include include ...} {range range ...}
  ```

- **accumulating a photon list (short form)**

  in the case one wishes to accumulate many photon lists, one can default the kind of accumulation setting the context to photon, after which one can issue an arbitrary number of accumulation commands in the short form (if one wishes to interleave with another kind of accumulation it is sufficient to use the the normal form

  ```
  xasset context p[hoton]
  accumulate pfile [{packet} {include include ...} {range range ...}
  ```

- There is no way nor sense in specifiying a quantity or to use the pseudo accumulation mode for photon lists, the environment variables xquantity, yquantity or accummode have no effect.

---

8. Accumulating HK time profiles

An *HouseKeeping* (HK) time profile is like a normal light curve, only the quantity present as a function of time are not counts taken from a science data file, but is the value of an HK parameter (voltage, temperature, setting ...).

These accumulations are similar to normal accumulations, one main difference being that one shall select a mnemonic code for the wished HK parameter, and this often automatically selects the telemetry packet to use.
As for normal accumulations as prerequisites one shall ensure that

- FOT tape data have been filed to disk
- setting of the appropriate environment variables is complete
- a valid observation chain has been defined.

- the default accumulation is invoked with command

  ```
  accumulate hk [ tfile hkname {packet} trange tbin
  ```

- there is no HK context to be selected
- There is no way nor sense in using a quantity different from TIME or to use the pseudo accumulation mode for HK time profiles, the environment variables xquantity, yquantity or accummode have no effect.

- However the following environment variable can be used to control the default behaviour. By default the digital values of the HK parameter in telemetry are converted to engineering (or physical :−)) units as appropriate. This behaviour can be reversed if raw digital data are wished. Toggle is possible using commands :

  ```
  xasset hkconvert Disabled
  xasset hkconvert Enabled
  ```

- A list of the various instrument HK parameters is provided elsewhere.

8.1 HK parameter list

It is intended to provide here pointers to instrument specific pages with the list of all mnemonic codes used to refer to SAX HK parameters. In absence of such pages, the `$XASTOP/calib/sax/*/*.pcf` files contain such a list in computer−readable (*but also human−readable* !) ASCII files.

- spacecraft HK, inclusive of all derived attitude and ephemeris information, is described elsewhere.
- LECS HK
- MECS HK
- HPGSPC HK
- PDS HK
- WFC HK

8.2 MECS HK parameter list

We list here the names of all MECS HK parameters. The notation `mi` or `i` is used to refer to the three MECS units (`m1`,`m2`,`m3`) and/or to parameters relevant to a particular unit.
Parameters of some use for the general observers are some of the ratemeters, the POP status, and perhaps the temperatures and the LED status (which shall be OFF). The remaining status parameters are usually never commanded other than at the nominal level, while the voltages should be very stable (therefore uninteresting).

- **Engineering ratemeters**

  These are count values, sampled every 1 s, in Engineering packets `MiENG000`. These packets are present only when scientific data production (POP) is enabled.
  The parameter name is the same for the three MECS units, one accesses the parameters of one specific unit by selecting the appropriate packet.

  - `trigger`, counts triggering the detector
  - `valemin`, events accepted above the low energy threshold
  - `rejemax`, events rejected above the high energy threshold
  - `valblmin`, events accepted above the low burst length threshold
  - `rejblmax`, events rejected above the high burst length threshold
  - `rejmpbus`, events rejected when microprocessor is busy (+ pile−up)
  - `reje`, events rejected by s/w energy thresholds
  - `rejbl`, events rejected by s/w burst length thresholds
  - `rejxy`, events rejected by s/w spatial (X,Y) thresholds

- **Safety ratemeters**

  These are count values, sampled every 1 s, present in instrument HK packets `MEHKD000`. These packets should be present even when no Engineering packet is there.

  - `rmi` (i=1,2,3), same as `valemin`, see above

- **(High) voltages**

  every 16 s, in instrument HK packets `MEHKD000`.

  - `hvpmti` (i=1,2,3), HV of the PhotoMultiplier Tube
  - `hvdrifti` (i=1,2,3), HV of the Drift region in the gas cell
  - `hvwini` (i=1,2,3), HV applied to the Window of the gas cell
  - `eup5` and `eum5`, voltage of the +/− 5 V supply line
  - `eup12` and `eum12`, voltage of the +/− 12 V supply line

- `dup28`, voltage of the +28 V supply line

- **Instrument temperatures**

  every 32 s, in instrument HK packets `MEHKD000`.

  - `temppmti` (i=1,2,3), temperature of the PhotoMultiplier Tube supply
  - `tempdrii` (i=1,2,3), temperature of the Drift HV supply
  - `tempdc1` and `tempdc2`, temperatures of the prime and redundant DC/DC converters

- **Miscellaneous status information**

  every 64 s, in instrument HK packets `MEHKD000`.

  - `popstat`, POP status (ENABLE,SUSPEND,DISABLE)
  - `popid`, POP identifier (not interesting)
  - `itmode`, instrument FIFO mode (normal or diagnostic, not interesting)
  - `duall`, ON/OFF flag of the "all detector unit" relay
  - `dumi` (i=1,2,3), ON/OFF flag of individual unit relay (not used)
  - `led`, ON/OFF flag for the Light Emitting Diode stimulation
  - `dbiif`, Data Bus Interface flag (not interesting)
  - `pmtmi` (i=1,2,3), ON/OFF flag of individual unit PMT (not used)
  - `windowmi` (i=1,2,3), ON/OFF flag of individual unit window HV (not used)
  - `driftmi` (i=1,2,3), ON/OFF flag of individual unit drift HV (not used)
  - `xelomi` and `xehimi` (i=1,2,3), low and high energy threshold
  - `dbl1lomi`, `dbl1himi`,`dbl2lomi`, `dbl2himi` (i=1,2,3), low and high (1st and 2nd) burst length (digital) threshold
  - `dxlomi`, `dxhimi`,`dylomi`, `dyhimi` (i=1,2,3), low and high (X and Y (digital) threshold
  - `throff` safety count threshold for automatic switch off

- **Spacecraft controlled temperatures**

  every 5 s, in Thermal Control Unit telemetry packets `TCHKD000`

  - `tmumi` (i=1,2,3), for the Mirror Units associated to each MECS unit
  - `tboxmi` (i=1,2,3), for the Detector Unit box of each MECS unit

---

9. Cross−accumulating from photon lists

Cross−accumulation is the process of generating data structures (like images, spectra, time profiles and photon lists) in a mission−independent format starting from other mission independent files.
Currently prototype programs exist which accumulate images, spectra, time profiles from XAS photon files.
It is planned to add also other form of cross−accumulators (e.g. to accumulate time profiles or spectra integrating a TIME−PHA pseudo−image)

The syntax of the relevant command is similar to the standard accumulate command. Note however that these programs have been used seldom, and their syntax is not yet fully upgraded to the one of the other programs (in particular for what concerns time quantities), therefore the following syntaxes are to be considered only indicative.

```
xaccumulate spectrum [ sfile {inpfile} quantity xrange xbin  {range range ...}

xaccumulate image ifile {inpfile} xquantity yquantity xrange yrange  xbin ybin  {range rang

xaccumulate time [ tfile {inpfile}  quantity trange tbin  {range range ...}

xaccumulate photon [ pfile {inpfile} {include include ...} {range range ...}
```

---

10. Special accumulations

Special accumulations are in general related to parameters which are computed from other HK or science data, or which have some other peculiarities. These parameters are usually in the form of time profiles which are used to generate time windows which can then be applied to drive other accumulations.

The following kinds of special accumulations can be identified so far :

- spacecraft HKtime profiles
- auxiliary quantity calculation
- instrument specific gain history accumulation

10.1 Spacecraft HK time profiles

We consider here some different kinds of spacecraft−related HK quantities.

- instrument−related HK present in spacecraft telemetry files (and not in instrument telemetry)
- other HK present in spacecraft telemetry files
- reconstructed attitude parameters (computed on the ground)
- orbit ephemeris parameters (computed on the ground)

A characteristics of such data in the case of SAX is that the relevant files are present as a single file for the entire observing period, and not as a file for each observation. The current observation chain is used only as a mean to select the time range for the accumulation.

The command used to accumulate spacecraft HK time profiles is the same used for other HK. Only the parameter hkname has to be selected in the following list

- **instrument−related HK in spacecraft telemetry files**

  will be listed together with other instrument parameters (apart from the nature "per observing period" there is logically no difference).

- **other HK present in spacecraft telemetry files**

  none useful known so far.

- **attitude parameters**

  - `xra` and `xdec`, the celestial coordinates of the satellite X−axis
  - `yra` and `ydec`, the celestial coordinates of the satellite Y−axis
  - `zra` and `zdec`, the celestial coordinates of the satellite Z−axis (i.e. the one close to the Narrow Field Instrument pointing direction)
  - `strconf`, the star tracker flag, which assumes the values
    - 0 : no star tracker in use
    - 1 : −X star tracker in use
    - 2 : Y star tracker in use
    - 3 : −X and Y star trackers in use
    - 4 : Z star tracker in use
    - 5 : −X and Z star trackers in use
    - 6 : Y and Z star trackers in use
    - 7 : all three star trackers in use
    - 9 : star tracker information missing in attitude file (old files)

- **orbit ephemeris parameters**
  - `xpos`,`ypos` and `zpos`, the satellite position
  - `xvel`,`yvel` and `zvel`, the satellite velocity
  - `utcflag`, the UTC to TAI difference
  - `saga`, the SAGA (South Atlantic Geomagnetic Anomaly) flag [definition to be clarified care of Telespazio]
  - `fromsaga`, time since last SAGA entrance
  - `edr`, the EDR flag [definition to be clarified care of Telespazio]
  - `fromedr`, time since last EDR entrance

10.2 Auxiliary quantity calculation

Auxiliary quantities are defined here as some interesting parameters which are not contained in the distributed FOTs but are computed using information contained therein (namely using attitude and ephemeris data).

The command `saxauxcalc` produces five time profiles with one second resolution, spanning the entire observing period. The time profiles have fixed names and content.

- `bright.time`, the angle between the NFI axis and the bright Earth
- `earth.time`, the angle between the NFI axis and the Earth
- `pointtype.time`, the pointing flag, defined as
  - 0 for a normal pointing (NFI looking at the target in the sky)
  - 1 NFI looking at the bright Earth
  - 2 NFI looking at the dark Earth
- `rigidity.time`, the magnetic rigidity
- `sunangle.time`, the sun aspect angle

The `pointtype.time` file is the one normally used to make time windows.

This program is a wrapper written by Daniele Dal Fiume around original code by A. Hazell of ESA SSD.

11.Instrument specific corrections

This section is reserved for any other kind of correction or data reduction procedure which is specific to a particular instrument. Whenever possible a common, instrument−independent interface is used to tie together the same "class" of operations.

So far the following classes of operations have been identified.

- event−by−event corrections like
  the MECS corrections, including
  - the linearization (correction of positional, geometric distortions)
  - the positional dependency of gain
  - the time dependency of gain (see gain history below)
- fancy event selections
  currently defined only for the MECS as spatial region selections and handled similarly to corrections
- the accumulation of gain history

11.1 MECS event corrections

In the case of the MECS a number of corrections need to be applied to raw events in order to normalize their position and energy information. In principle these corrections could be applied post−facto to a raw photon list before using it as input for cross−accumulations, however so far a different, faster, way has been implemented, which is to handle all corrections on request, transparently during the primary accumulation of images, spectra, time profiles or photon lists.

Some corrections are mandatory (unless one is interested in raw detector events), while other corrections are optional (but require the mandatory ones).

Selection of data in fancy spatial regions (other than a single standard XY box), although strictly not a correction, is implemented as an optional correction.

These are the kind of corrections and a very brief description of their nature.

- **Positional dependency of gain**

  This correction is mandatory if other corrections are wished. The raw PHA (energy) of an event is corrected dividing it by a factor g(x,y) which is the relative gain w.r.t. the detector centre (mapped in raw detector positions), i.e. the gain is effectively brought back to the detector centre.

- **Linearization**

  This correction is mandatory if other corrections are wished. The raw detector positions x,y of each event are first converted to mm on the focal plane using a cubic relationship (with the first order term having a slight energy dependency), and then to "new" pixels of user selected size.

- **Time dependency of gain**

  This optional correction multiplies the positionally−corrected PHA of each event by a correcting factor. This takes into account the gain history described elsewhere (relative gain vs time) and also normalizes to a common energy scale (so−called PI channels).

- **Selection using arbitrary spatial regions**

  This is not properly a correction, but just the selection or rejection of an event on the basis of its position laying inside or outside a region of interest. For convenience rejected events are assigned an illegal position (so that they are automatically rejected by preexisting software)

In order for corrections to be applied during primary accumulations they must currently be explicitly enabled. Otherwise one obtains uncorrected files using raw detector X,Y and PHA.

- **Mandatory corrections**

  These are enabled as :

  ```
  xasset correction Enabled
  ```

  Additionally all primary accumulations programs will currently require three additional arguments following all others. THIS IS PROVISIONAL AND WILL BE CHANGED IN THE FUTURE TO USE ENVIRONMENT VARIABLES. These extra arguments may take the following forms :

  ```
  N npix seed

  S pixsize seed
  ```

  In the first form one requires `npix` pixels to cover the original field of view, in the second case one requires the pixel size to be `pixsize` mm. In either case `seed` is a seed for a random number generator.
  The current default uses 256 pixels covering the field of view, which gives a pixel size (of approximately 0.17 mm) DIFFERENT for the three MECS units and corresponding to the unlinearized pixel size close to the detector centre.

- **Time dependency of gain**

  The (optional) correction of the time dependency of gain is enabled as :

  ```
  xasset gainhistory gainfile
  ```

  where `gainfile` is a gain history produced by the appropriate command

- **Arbitrary spatial region selection**

  The (optional) selection of data using spatial regions is enabled in either of the following two ways :

  - ```
    xasset selectfile ifile
    xasset selectmethod Map
    ```

    In this form `selectfile` is a XAS image file containing a mask defining the region of interest of arbitrary shape. The image is interpreted as a binary mask, all pixels with a non−zero value define the region of interest.
    No facilities are provided to generate such maps, but it is very easy to do so e.g. in IDL or MIDAS.
    ```

- `xasset selectfile rfile`
  `xasset selectmethod Region`

  In this form `selectfile` is a SAOimage−style region file (recommended filetype is `.region`) which is interpreted as follows :
  - lines starting with # are comments
  - all positions are in linearized pixels of the current size setting
  - at the beginning it is assumed that all pixels are to be rejected
  - any non−comment line defines a region, which can be of INCLUDE or EXCLUDE type
  - regions are processed strictly in the order in which they appear, flagging the relevant pixels to be accepted or rejected (and eventually changing a previous selection)
  - the following SAOimage−style syntaxes are supported and interpreted as explained

    `BOX(xc,yc,xs,ys,angle)`
    a rectangular box of full sizes `xs,ys` with centre in `xc,yc`, optionally rotated by a given `angle` (rotation of the X−size w.r.t. the X−axis)

    `CIRCLE(xc,yc,radius)`
    self−explanatory

    `ELLIPSE(xc,yc,xs,ys,angle)`
    an ellipse of SEMI−major and minor axes `xs` and `ys`, centred in `xc,yc`, optionally rotated by a given `angle` (rotation is interpreted as the rotation of the semimajor axis w.r.t. the X−axis ; this semantics is seemingly different from the SAOimage one but appears to give the same results.

  - an EXCLUDE region is defined prefixing with a minus sign, e.g. `−BOX`, `−CIRCLE`, `−ELLIPSE`
  - no other SAOimage syntaxes are supported, in particular polygons are not supported, and expressions are not supported. Annuli can be obtained using in region file a `CIRCLE` followed by a concentric `−CIRCLE`
  - the following additional syntax is supported

    `SECTOR(xc,yc,radius,theta1,theta2))`
    a sector (pie−slice) of the circle of centre `xc,yc` and given `radius`, included between the two angles `theta1` and `theta2`. Such angles are in degrees from the X−axis in the range 0−360. It is not allowed to have a sector crossing the +X axis (i.e. from −45 to +45), an equivalent effect can be obtained using two adjacent sectors (0−45 and 315−360)

xasunsetting the `selectfile` variable is sufficient to disable the selection.

11.2 Gain history accumulation

The purpose of this command is to accumulate a *time profile of the relative gain* (PHA channel vs energy relation). This is done in an instrument dependent way using on−board radioactive sources. We provide an instrument−independent command to accumulate a gain history (which is just a XAS time profile), as a front end to instrument−specific programs.
The syntax of the command and the usage of the gain history file may of course be different for each instrument.

- **format of the common command**

  `accumulate gain [gainfile ncounts {packet} {range range ...}`

  This accumulates a gain history collecting a sequence of spectra from the on−board calibration sources. Each spectrum has a statistics of about `ncounts` events. The spectrum is fitted to determine the gain (channel of the peak), and the gain relative to an appropriate reference is written to the output time profile (which has one time bin per spectrum). The syntax is in general similar to the spectrum accumulation command, except that the selection of calibration events is done automatically.

- **MECS**

  For the MECS the instrument−specific program run is `mecsgainaccum`. This program requires that the mandatory corrections have been enabled. Of course it will ignore any existing gain history, since it is creating one. The program does not ask for ranges in PHA, X or Y but accumulates a (sequence of) full range spectra of *both* on−board radioactive sources, with the gain reported to the detector centre.

  There are a variety of options which can be used to control the behaviour of the program which are documented elsewhere . Here we report only two environment variables used to control the amount of terminal output produced:

  `xasset quiet Yes|No`

  `xasset verbose Yes|No`

  A `quiet=YES` (default) run produces the gain history file and no significant terminal output.
  A `quiet=NO` run produces, in addition to the gain history file, a screenful of information for each fitted spectrum, and a log file in the current printdir.
  A `verbose=YES` run produces a lot of terminal output, including each single iteration of the fit.

Note that spectra are accumulated until enough counts are collected (in an entire number of packets). Spectra with less counts can be accumulated in the case of data gaps. The fitting procedure is run only on spectra with significant statistics. The fitting uses a CURFIT routine stepping on the peak position and fitting the Gaussian normalization and FWHM. The errors are 90% confidence errors for one interesting parameter. The result of all fits is reported in the log files, but only non−suspicious ones are included in the gain history (ANY trouble during fitting, including an open confidence contour, flags a fit as suspicious). The gain vs time correction will then duly interpolate.

Note that no provision for interpolation or smoothing are made IN the gain history generation itself. If one wants to, it may manipulate the gain history (e.g. in IDL) and apply the manipulated version.

- **other instruments**

  This command is currently not (yet) implemented for other instruments.
  Its purpose will be different for those instrument like the PDS and the HPGSPC, which have an Automatic Gain Control (AGC), i.e. it will just be a verification of the AGC correct operation.

---

12. Other corrections

This section is reserved for any other kind of correction or data reduction procedure which is not instrument specific and acts on a generic XAS data file.
Examples may include dead time correction, background subtraction, etc.

There are no such programs at the time of writing.

13. Generation of time windows

Time windows are series of time intervals which can be used to select data according to some acceptance criterion, and applied to accumulations.

Time windows are saved as ASCII files and can be generated and manipulated in a number of ways :

- graphically from a displayed time profile
- as intensity windows
- via manual edit of an existing time window file
- merging pre−existing time windows
- as phase windows

Time windows can be easily applied to all forthcoming accumulations, and can be visualized graphically.

13.1 Time window files

Time window files are ASCII files, with filetype `.window`, which are created in the current data directory.

A time window file includes a number of records, each one specifying the start and end time (UT) of a time interval, in free format, as follows :

```
yyyy mm dd hh mm ss ff  yyyy mm dd hh mm ss ff
```

where `ff` is the fraction of seconds (in hundredths of s). Note that not all programs may support such precision.

The usage of a free ASCII form allows easy manual editing

13.2 Graphical generation of time windows

Time window files can be generated very easily in an interactive way (with the mouse) from the graphical representation of a XAS time profile using the following command :

```
twindow [wfile [NOLOOP]
```

Unless the NOLOOP argument is specified, one can enter an arbitrary number of time intervals, just clicking on their start and end times in the graphical window.
If NOLOOP is specified, the program terminates automatically after a single interval.
Otherwise one can terminate the program either :

- clicking twice outside the axis frame in the graphical window (recommended)
- typing control−C in the terminal window from which the program was invoked, and clicking once anywhere in the graphical window (obsolescent)

Note that clicking outside of the axis frame always causes termination.

13.3 Generation of intensity windows

Intensity windows are time window files defined according to the count rate level of a time profile (or more generally the intensity level of an HK time profile) being in a given range. This range can be specified interactively in a number of ways.

- **Graphically**

```
iwindow [wfile CURSOR Above
iwindow [wfile CURSOR Below
iwindow [wfile CURSOR Inside
iwindow [wfile CURSOR Outside
```

- **Based on a named time profile**

```
iwindow [wfile tfile Above y1
iwindow [wfile tfile Below y1
iwindow [wfile tfile Inside y1 y2
iwindow [wfile tfile Outside y1 y2
```

The main difference is that the value y1 above or below which is defined the acceptance, or the couple of values y1 y2 inside or outside whose range is defined the acceptance, are specified

- in the first case clicking the mouse respectively once or twice on the graphical representation of a time profile present on a graphical window
- in the second case entering manually the values and the name of the time profile to which they refer

---

13.4 Manual editing of time window

Time window files are plain ASCII files and can therefore edited in a straightforward way with any editor, in the case any adjustment to the time windows is wished. Accumulation programs may some simple compatibility checks to detect obvious mistakes.

13.5 Merging time windows

Time window files created according to a variety of criteria (e.g according to intensity levels of different HK parameters and/or phase criteria) can be merged into a single time window file before application to accumulations.

If more than two time window files have to be merged, the following command has to be applied repeatedly to couples of files.

To merge a couple of time window files use on of the following commands :

```
mergewindow [wfile wfile1 OR  wfile2
mergewindow [wfile wfile1 AND wfile2
```

where the first form takes the union of `wfile1` and `wfile2` (a time is considered within a good time window if it belongs to a time window in either files), while the second form takes the intersection of `wfile1` and `wfile2` (a time is considered within a good time window if it belongs to a time window in both files).

13.6 Phase windows

Not yet available

13.7 Applying time windows

Time windows can be applied to ALL forthcoming accumulations just by setting the appropriate environment variable to the name of a window file :

```
xasset timewindow wfile
```

Please note that :

- time windows remain in effect for all accumulations, UNTIL disabled setting the timewindow variable to blank.
- accumulations allow to specify a further time range, in which case only the time window intervals within such range are considered
- the name of the time window files is written to the accumulated file header

---

13.8 Visualizing time windows

Time windows can be displayed graphically over a pre−existing time profile plot with command

`overtrace window [ wfile`

Note that time windows are plotted as filled rectangles in the current overtrace colour, therefore they completely overwrite the plotted light curve. To replot it, issue again an appropriate overtrace command.

NOTE: the following forms are equivalent to the above command :

`display window [ wfile`

`wplot [ wfile`

---

14. Handling XAS files

A small number of convenience commands are provided for the generic handling of XAS data files. They include

- listing file headers on the screen
- listing the content of tabular files
- Editing the file header
- Conversion to other formats is described elsewhere
- in particular usage of XAS files on different operating systems

14.1 Listing file headers

A XAS file has a header composed by a list of keywords, which are normally used programmatically.
It is also possible to list the entire header at the terminal using the command

`hlist [ file`

which produces a listing of the form given below. It is also possible to use standard operating system facilities to redirect the output to a file, or to restrict the output to a limited subset of keywords (e.g. for Unix users `hlist` *`file > outfile`* or `hlist` *`file`* `| grep` *`keyword`*.

This is an example of the output. The type of each keyword is indicated (before its name and value) using codes

- (C) character (string)
- (I) 16−bit integer (INTEGER*2, use discouraged)
- (J) 32−bit integer (INTEGER*4)
- (R) 32−bit floating point (REAL*4)
- (D) double precision floating point (REAL*8)

```
File prova.spectrum                            of type SPE
Record length                                   16 bytes
Mini-header records                              2
Data        records                            256
Full header records                             73
------------------------------------------------------------------------

(J) BITPIX       =            8
(J) NAXIS1       =           16
(J) NAXIS2       =          256
(J) TFIELDS      =            4
(C) FILENAME     = prova_spectrum
(C) OBSERVER     = SAX TEAM (OP_884
(C) OBJECT       = Crab
```

omissis...

```
(C) PACKET       = m1dir002
(J) NPACKET      =         3235
(D) EXPOSURE     =  4554.62890625
(C) HISTORY      = saxhaccum prova m1dir002 PHA 0 255 1 1996 9 6 19 51 9 1996 9 6 21
(C) HISTORY      = + 17 20 27 60 0 255 0 255
------------------------------------------------------------------------
For a total of     48 keywords
```

14.2 Listing file content

It is possible to list part (i.e. a range of records) of the content of a XAS tabular file (i.e. a spectrum, time profile or photon list, but not an image) at the terminal using the command

`tlist [ file records`

which produces an ASCII listing of the binary content of all columns in the file in an appropriate format.

As a particular convenience, in the case of time profiles, the TIME column (which is stored internally as a double precision number of seconds since midnight of a reference day) may be presented in the more usual form `hh:mm:ss.ff` if the following environment variable is set

`xasset hms Yes`

14.3 Editing the file header

The keywords in a XAS file header are normally set by the program which generates the file, and may be altered (added or modified) by other programs which manipulate the file, or inherited by children files. In particular a sequence of `HISTORY` keywords keep track of the manipulations done.

It is however possible to edit one header keyword at a time also manually using the appropriate program. This command can be invoked in two ways :

- `header_edit [ file keyword value(s)`

  is the form used to edit an existing keyword. If the keyword is an array of numeric values, one can supply as many values as the original array contains. If the keyword is a character string, one can supply a string as long as the original value. In no case a keyword can be extended beyond the creation length, or changed of type.

- `header_edit [ file keyword value keytype`

  is the form used to add a new keyword. New keywords are always scalar quantities. One must supply also the appropriate keyword data type.

Note that "duplicatable" keywords (typically `HISTORY` and `COMMENT`) are always interpreted as new keywords. It is not possible to alter the previous values of an existing duplicatable keyword. It is also not possible to assign as a value a string containing imbedded blanks.

15. Generating response matrices

Generation of *response matrices* can be handled via a front−end program like other accumulations or calling directly the appropriate instrument−dependent program. Matrices can be produces in XAS format or directly in OGIP (XSPEC−compatible) format.

As prerequisites one shall ensure that

- setting of the appropriate environment variables is complete
- one knows the instrument configuration and the parameters used for accumulating spectra (THIS IS YET SUBJECT TO EVOLUTION AND WILL BE FINALISED LATER)

A separate description explains what is an RMF or an ARF, and when either is produced.
There are a few ways of controlling the behaviour of the accumulation program.

- **accumulating a matrix (normal form)**

  the simplest form of issuing the command is

  ```
  accumulate matrix [rmffile arffile {instrument−dependent−args}
  ```

- **accumulating a matrix (short form)**

  in the case one wishes to accumulate many matrices, one can default the kind of accumulation setting the context to matrix, after which one can issue an arbitrary number of accumulation commands in the short form (if one wishes to interleave with another kind of accumulation it is sufficient to use the the normal form

  ```
  xasset context m[atrix]
  accumulate rmffile [arffile {instrument−dependent−args}
  ```

- **invoking instrument dependent programs**

  one can also invoke specific programs directly by name

15.1 Environment setting for matrices

Before invoking the "accumulate matrix" command one must have defined the current instrument (this is not necessary if one invokes the instrument dependent program directly). This is achieved by the following command :

- `xasset instrument instrument`

Moreover, one can control the format of the output matrix file (default is OGIP format) via the following command :

- `xasset matform Ogip`
- `xasset matform Xas`

15.2 Instrument dependent matrix programs

The accumulate matrix command will in turn invoke one of the following instrument−specific programs. Such programs may have additional arguments which are different case by case [TO BE COMPLETED] :

- **MECS**

  ```
  mecsmaccum [rmffile arffile meunit {other−args}
  ```

  where the ME unit argument is compulsory in all cases, while the other arguments (TO BE CHANGED) are required only if an ARF is requested.
  See here for a guided tour to the MECS matrix

- **Other**

  TO BE WRITTEN

15.3 RMFs and ARFs

An RMF (*Response Matrix File*) is logically a 2−d array which gives for each input photon energy the probability of assigning the photon to a given PHA channel.
An ARF (*Ancillary Response File*) is logically a 1−d array, indexed on input photon energy, which contains all other contributions to the effective area. This may need to be calculated more often, since it may depend on instrument setting, position in the FOV, etc.
In practice the RMF file can be a pure RMF (contains an adimensional probability), or can contain the *product* of the RMF and the ARF.

XAS users have the option of producing separately RMFs, ARFs or both. This interacts with the choice of matrix format as follows :

- *if the OGIP format is selected*
  - the RMF (always a pure RMF) is produced only if rmffile is not equal to "none"
  - the ARF is produced only if arffile is not equal to "none"
  - both are produced if both filenames are given
- *if the XAS format is selected*
  - the rmffile argument must always be specified
  - no separate ARF is produced in any case
  - if arffile is equal to "none", the matrix file in XAS format contains the pure RMF.
  - if arffile is any other value, its name is ignored, but the matrix file in XAS format contains the product of the RMF, the ARF and the input energy bin width (i.e. has a dimension of cm2 keV)

15.4 Matrix file formats

XAS users have the options to produce response matrix data in either of two formats, either directly or later using the fromogip and toogip conversion commands.

According to the format selected the following files are produced :

- **XAS format**

  `rmffile.matrix`
  is a XAS 2−d image containing the matrix (adimensional RMF or cm2 keV according to the choice made. This file can be viewed and plotted as a normal image.
  `rmffile.histo`
  the associated histogram file is a XAS 1−d image containing the input energies

- **OGIP format**

  `rmffile.rmf`
  is a FITS table containing the compressed matrix according to OGIP specifications.
  `arffile.arf`
  is a FITS table containing the effective area (ARF)

---

16. Exporting XAS files to other packages

XAS data files can be used natively with SAOimage and IDL, and can be exported for use with a variety of other packages (including MIDAS, IRAF and XSPEC.

The following sections contain direction for use or export of XAS files with different packages.

- using XAS images with SAOimage
- using XAS files in IDL
- exporting XAS files as ASCII files
- converting XAS files to/from plain FITS files
- converting XAS files to/from OGIP FITS files
- converting XAS files across different operating systems

16.1 Using XAS images with SAOimage

XAS images (including response matrices) can be visualized using CfA SAOimage, in addition to the XAS display command. This is done as follows :

Issue command

`saodisp [ ifile`

which returns the SAOimage command string appropriate to the given image size, then invoke SAOimage copying the given string (either retyping it or using mouse cut−and−paste)

Unix users may want to alias the saodisp command as `‘eval saodisp‘` so that it will invoke SAOimage directly.

16.2 Using XAS files in IDL

Any XAS data file can be read directly into IDL, where it can be manipulated and displayed. It is also possible (limited) rewrite of the output of a manipulation in XAS form.

The interface between XAS and IDL is *not part of the XAS core package*, but is available separately as contributed software (this will not be formally supported or mantained as the XAS core). There are two variants of the IDL interface :

- a vanilla interface (xasread) which allows to read a XAS file (inclusive of all its header keywords) as an IDL structure (and also to rewrite a modified structure to an existing XAS file). This has been developed by Lucio Chiappetti
- a fancier interface (xasplot) which allows sophisticated plotting options using IDL widgets. This has been developed by Luciano Nicastro
- a parallel to the vanilla interface (xasasc) is used to read "XAS ASCII" files into IDL for plotting and manipulation. This might be used in conjunction with the export of XAS files into ASCII to access smaller parts of XAS data files.

16.3 Exporting XAS files as ASCII files

XAS tabular files (or part of them) may be easily converted to ASCII files, which can be used as an interface to any program, in particular to plotting programs like QDP or Supermongo or even IDL itself.

Command tlist is normally used to produce on the terminal an ASCII dump of (part of) a XAS tabular file (any data file not in image form). This output can be captured into an ASCII file using standard operating system input/output redirection facilities, e.g. in Unix

`tlist [file records > asciifile`

16.4 Converting XAS files to/from plain FITS files

All kinds of XAS files can be converted one−to−one to plain FITS files (in the form either of FITS images or FITS binary tables). This provides an effective way of importing XAS data into any astronomical package. The burden of converting to the conventions used in the target package (e.g. column names or order, keyword names) is left to the user.

Conversely a plain FITS file (FITS image, single−extension FITS binary table) can be converted into a XAS file. This is assumed mainly to re−import files manipulated in other packages.

See elsewhere for conversions using OGIP conventions.

There are three commands to handle XAS <−−> FITS conversion :

- **conversion from XAS to FITS**

  `tofits [file`

  converts a XAS file (the filetype part of the name may be omitted if it can be derived from the context) into a FITS file. The name of the output file is obtained replacing the filetype dot with an underscore and adding a `.fits` extension (thus e.g. `pinco.image` gives origin to `pinco_image.fits`. XAS image files (images, pseudoimages and response matrices) are converted to FITS images, all other files to FITS binary tables.
  It is care of the user to rename the file if so is required by the target package.

- **conversion from FITS to XAS**

  `fromfits [fitsfile logflag commentflag`

  converts a FITS file (whose name MUST include any filetype extension) into a XAS file. The name of the output file is derived "inverting" the naming convention used by tofits, if at all possible, otherwise it will do its best to assign a XAS file type (e.g. `pinco_image.fits` recreates `pinco.image`, but an image with name `pinco_panco.fits` creates `pinco_panco.image`)

- **finalising the conversion from FITS to XAS**

  It is possible that fromfits does not recognise the content of a tabular FITS file, and therefore names it using a `.generic` extension, and flagging it as a generic XAS table. Of course renaming is always possible, but it may be desirable to flag the file as one of the standard XAS types (if its content is compatible). This is achieved with

```
degeneralize [file xastype
```

Beware that the conversion from/to FITS is handled using the FITSIO library, and therefore it uses that library's error conventions (in particular an existing FITS file is never overwritten, contrary to XAS usage) and codes.

16.5 Converting XAS files to/from OGIP FITS files

Some kind of data files (although formally FITS) have very peculiar formats according to the OGIP conventions used by some X−ray astronomy packages. Dedicated programs exist to handle these peculiarities (however only mandatory items are handled; any optional keyword must be added/edited care of the user using target package's facilities).

Note that response matrices can be accumulated directly in OGIP form, without need of conversion.

There are two commands to handle XAS <−−> OGIP conversion :

- **conversion from XAS to OGIP**

  `toogip [file {extraargs}`

  converts a XAS file (the filetype part of the name may be omitted if it can be derived from the context) into a FITS OGIP file. The name of the output file is obtained replacing the filetype dot with an underscore and adding a `.fits` extension.
  `extraargs` may be asked if the conversion process requires naming additional files.
  It is care of the user to rename the file if so is required by the target package.

  The XAS file types handled so far are response matrices (`.matrix`) and spectra (`.spectrum`).

- **conversion from OGIP to XAS**

  `fromogip [fitsfile {extraargs}`

  converts a FITS OGIP file into a XAS file. The file types handled so far are response matrices and spectra.
  In the case of matrices if the FITS file is a "raw" RMF, `extraargs` have to be used to supply the name of an associated ARF. The resulting XAS matrix will be the product of the two. If `extraargs` is `none` just the RMF will be converted to XAS format.

Beware that the conversion from/to FITS is handled using the FITSIO library, and therefore it uses that library's error conventions (in particular an existing FITS file is never overwritten, contrary to XAS usage) and codes.

16.6 Converting XAS files across different operating systems

XAS data files use the *native binary representation* of the operating system on which they were created. Since these representation may differ, a XAS data file may not be usable on another operating system.

The creation system is recorded in the file header. XAS programs will recognize whether the creation system is consistent with the current operating system (e.g. Ultrix and OSF/1 are consistent, SunOS and HP−UX are consistent, but OSF/1 and SunOS are not consistent because of byte swap, and no Unix is consistent with VMS). If it is not will give a warning inviting you to run the following command.

```
localize [file newfile

localize [file INPLACE
```

The conversion may take place either creating a new file, or *modifying* the original file in place. Note that this will render the file unusable on the original system, unless it is `localize`d back.

`localize` must always be run on the *target* system, not on the origin system.

17. Displaying XAS files graphically

XAS provides facilities for simple standard plotting and visualization of data. These facilities are not intended for publication quality plots, for which one may export XAS files to one’s favourite package.
See in particular elsewhere for usage of SAOimage for image display and of IDL for general plotting.

Plotting is done by the user by just a couple of "client" commands which use an underlying graphic server. There are two main kind of servers, X−window and PostScript.

One proceeds as follows :

- create a server instance
- set the appropriate environment variables
- invoke the `display` command to plot a file in a fresh frame
- invoke the `overtrace` command to plot a file over the current plot
- one may additionally control the plotting behaviour of the server.
- at the end one terminates the server.

17.1 Kind of graphics servers

The basic idea behind the scene is that there is only one program (the *server*) permanently active which knows how to plot in a device−dependent manner, and that all user plotting commands (the *clients*) just communicate with the server via a communication channel using a limited set of standard (binary) commands corresponding to graphics primitives.

There are two main flavours of servers currently available in XAS, each of which may exist in several instances.

- **X−window server (xw)**

  The X−window (X11) server is the one commonly used for interactive plotting. Each server instance (e.g. `xw1`, `xw2`, etc.) keeps alive on the screen a graphics window. Both vector plots and images can be displayed. The user may interact with the graphics using the mouse.

- **PostScript server**

  There are three sub−flavours of PostScript servers (they actually share a single executables and differ only in the PostScript macros invoked for some operations). These servers are not interactive, and have been tested and developed only in a limited way. They write to a cyclic pool of files `file00.ps` to `file99.ps` in the current printdir.

  - **Black & White PostScript server (bw)**

    This server (whose instances are named `bw1`, `bw2`, etc.) produces black and white (including gray scale images) plots.

  - **Colour PostScript server (cp)**

    This server (whose instances are named `cp1`, `cp2`, etc.) produces colour plots using PostScript Level 1 features.

  - **Colour Level 2 PostScript server (c2)**

    This server (whose instances are named `c21`, `c22`, etc.) produces colour plots using PostScript Level 2 features (essentially this means that colour images can be three times smaller).

17.2 Creating a server instance

Before issuing any plotting command, one shall have started the graphic server to which all plotting will be directed.
In principle one can create several instances of the same kind of server. Each instance is characterized by a sequence number.

One may then direct plotting to the wished server issuing command

`xasset plotter servinst`

The default plotter is `xw1`.
Beware that switching among plotting server uses a single graphics environment which is not reset when one switches back !

The command used to start a new instance of a given type of server is

`createserver [server instance {extraargs}`

where the additional `extraargs` are currently defined only for the X−11 server as:

- `xcorner ycorner`
  the pixel coordinates of the corner of the window to be created (whether and how these are obeyed depends on the particular operating system X11 implementation).
- `xsize ysize`
  the size in pixel of the window to be created

A server is normally used from a single session (i.e. is associated with the terminal window from where it was started). See however directions for using the same server from more than one session.

17.3 Graphics environment variables

There is a wide range of environment variables which can be used to control the appearance of plots.
Most of them are assigned a values with the usual xasset command.
Note however that all variables which have a couple (or more) of numbers as values (i.e. axis−related) shall be assigned EXCLUSIVELY using command xasplot.

We provide here a list of all graphics related variables under user control (there are others which are "hidden" for internal program use), and we refer to the complete list of variables for full details. An asterisk before the variable name identifies variables which MUST be set using xasplot.
All such variables have suitable defaults.

- `plotter` : current plotting server
- `overwrite` : used internally
- `overcount` : used internally
- `* viewport` : part of window/page used for plotting ...
- `* xyaxis` : ... and relevant world units (for internal use)
- `* xaxis` : X−axis range in world units
- `* yaxis` : Y−axis range in world units
- `xscale` : whether X−axis is linear or log
- `yscale` : whether Y−axis is linear or log
- `zscale` : scaling method for images
- `nlevel` : number of levels for image display
- `replotaxes` : reserved
- `pen` : generic pen colour
- `bkgpen` : colour for background
- `axispen` : colour for axes
- `datapen` : colour for data points
- `textpen` : colour for text labels
- `datastyle` : whether error bars, solid lines etc. are plotted
- `plotcomment` : amount of textual comments in window/page
- `scroll` : reserved
- `zoom` : reserved
- `colorbar` : reserved
- `zfile` : reserved
- `fulldisplay` : reserved
- plus other used internally

To know the setting of all graphics related variables issue command

```
xasplot query
```

There are in additions other variables which are not shown by `xasplot query` and can be used for purposes more related to the semantics of the plots.

- `timeaxis` : label time axis in sec or hours
- `tquantity` : choice of column in time profile plots

17.4 The xasplot command

This command MUST be used to set all graphics a XAS environment variable whose value is not a single number or string.
This currently applies to variables like the axis extrema `xaxis` and `yaxis`.

- **Setting a variable**

  `xasplot variable value value ...`

  This form is used to set any XAS variable whose value is a couple (or two) of numbers.

- **Querying all values**

  `xasplot query`

  This form is used to know the values of all graphics related variables.

- **Querying a single values**

  Use the standard xasset command

- **Deleting (unsetting) a value**

  Use the standard xasunset command.

17.5 The display command

This command is used to display a XAS data file in a new window/page, i.e. the window is cleared and a new set of axes and annotations is plotted using the current colours.

This command is used for all kind of displayable data files (i.e. spectra of any type, time profile of any type and images of any type including response matrices).

The command can be invoked with slightly different syntaxes.

- **specifying explicitly what has to be plotted**

  the simplest form of issuing the command is one of

  ```
  display image [ ifile
  ```

  ```
  display spectrum [ sfile
  ```

  ```
  display time [ tfile
  ```

  ```
  display window [ wfile
  ```

  one does not have to add the file type to the file name, if it is the default `.image`, `.spectrum`, `.time`, `.window`, while one MUST add the file type if it is non−standard

- **using the file type**

  another simple form relies on the file type (which MUST be one of the standard ones

  ```
  display [ ifile.image
  ```

  ```
  display  [ sfile.spectrum
  ```

  ```
  display  [ tfile.time
  ```

  ```
  display  [ wfile.window
  ```

- **using the context**

  For convenience a shortcut form is possible if one works with only one kind of files, in this case ona can set the context to the wished type and let the program default to that.

  ```
  xasset context context
  display [ file
  ```

Finally note that the syntax `display window` is provided for compatibility, but actually overtraces the time windows on the current page.

17.6 The overtrace command

This command is used to display a XAS data file over the current plot (which must be of the same type). It is of course care of the user to ensure that one is plotting data which is in the same x and y range as the data on the screen !

The colour of the plot for the "xw" server, unless the user arranges otherwise, is obtained cycling through the fundamental colours (since the first plot defaults to white, one than has green, blue, cyan, magenta, yellow, white, red and again).

The command can be invoked exactly with the same syntax as the display command.

17.7 Controlling the server

In principle the user shall be able to do things which change the appearance of plots after the server is started. Not all features described below are implemented at the present time.

- **positioning the plotting area in the page**

  By default there is a single plotting area in the page (although in principle one can try and xasplot the viewport environment variable), which occupies a region controlled xassetting the plotcomment environment variable, as follows
  - the top half in default mode
  - almost all the window if `plotcomment short`
  - all the window if `plotcomment none`

  The remainder of the plot area is used for standard annotations.

- **changing plot axes**

  One can let the plot set the extrema of the axes automatically, or set them explicitly xasplotting the xaxis and yaxis environment variables. One may also explicitly xasset the xscale and yscale environment variable to `lin` or `log`.

- **changing plot colours**

  There are several environment variables which control the colour of various pens (for the X−11 and colour PostScript servers). There are 8 fundamental "plot" colours always available for plotting , in addition, by using a negative colour number, one may refer to "image" colours in the colour lookup table described below. The fundamental colours are :
  - 0 black
  - 1 white
  - 2 red
  - 3 green
  - 4 blue
  - 5 cyan
  - 6 magenta
  - 7 yellow

- **handling colour tables**

  There are commands to load the RGB colour tables used for image displays. The graphics servers use two sets of colours, the above listed fundamental colours for vector plotting, and a colour table (Look−Up Table, LUT) for

image display.

The size and content of the LUT are (partially or fully) under user control.

- The black−and−white PostScript server default LUT is a 256−level gray table (and of course it does not much sense to edit it).
- The colour PostScript server default LUT is also a 256−level table, initialized to gray, but is under complete user control (which may use and redefine up to 256 colours).
- The X−11 server LUT could in principle be as long as 256 colours, but in practice it is limited by the number of colours already allocated by the workstation X server (on a Motif session this is usually about 30). One may allocate n colours from a start location j in the workstation default colourmap, provided that there are n consecutive locations free. One possibility is to allocate a small (n=16,32) number of colours to each server instance. Exclusive allocation is needed only if one wants to redefine colours. Otherwise one can share the LUT of another, non−XAS application, and just tell XAS to "start from j" guessing where the other application's LUT starts.

The following two commands represent a PROVISIONAL interface.

`readlut [ startc ncol`

prints on the terminal screen the RGB values of the `ncol` colours starting at location `startc`. If `ncol` is positive, it is assumed that `startc` is relative to the current server instance start "j", if `ncol` is negative, one conventionally assumes that `startc` is absolute in the workstation X colourmap.

`writelut [ startc ncol rgb ...`

usually loads into the graphics server LUT `ncol` colours, each one specified as an RGB triplet `rgb`, from given location `startc`. However this command can be used to perform different tasks :

- `writelut 0 0` prints on the terminal screen the current start "j" as the particular graphics server instance thinks it is.
- `writelut j 0` redefine the current start to the absolute location "j".
- `writelut 0 n` loads n colours in a new XAS LUT from beginning.
- `writelut k n` loads n colours from the k−th location in the current LUT.

This command is practical only to edit the content of one or a few colours. IT IS PLANNED to have a dedicated command to load entire LUTs. In the meanwhile a command file can be used in conjunction with writelut to load long tables.

- **resizing the plot window**

  In principle the screen window used by the X11 graphics server can be resized using the mouse. The way this is done is dependent on the particular window manager. The graphics server will do its best to cope with this at the next plot, but, since it is not ICCCM compliant, unpredictable results are possible.
  We remind also that the backing store of an already plotted window when obscured (partially or totally) is demanded to the window manager, which may cease to mantain it at any time.

- **changing page size and orientation**

  Currently the PostScript server uses only portrait orientation on A4 paper. It is planned to have a `controlserver` program in the future for the handling of page (or window) size and orientation changes.

- **disposing of print files**

  Currently the PostScript server opens a new file each time a new page is plotted upon. At the same time the previous file is properly closed. The last file is closed when terminating the server. There are currently no actions to dispose of (e.g. print) the files ; this is left to the care of the user. It is planned to have a `controlserver` program in the future capable of disposing of plot files.

- **choice of fonts**

  All graphics servers choose the font in which text is plotted using a conventional font number (default is number 1). The mapping between font numbers and actual fonts can be customized by copying the files `xwfont.list` (for the X−window server) and `psfont.list` (for the PostScript server) from the `$XASTOP/include` directory to one of the following directories (they will be consulted in this order of precedence) :
    - the current working directory (for customization done personally by each user for a particular purpose)
    - the user home directory (for customization done personally by each user for all of hir sessions)
    - `$XASTOP/local` (for customization done by the XAS installer for all users on a given system)
    - `$XASTOP/include` will be used if no customization was applied. The files in this directory shall NEVER be modified.

  Note that the choice of the font size in points is kept separate from the choice of the typeface for the PostScript server, while standard X11 names must be used for the X−window server.

17.8 Using a server from more sessions

If one user is running XAS from more than one terminal window (on the same machine), and starts an instance of a graphics server this particular server is registered in the XAS environment for that particular session. Although communication channels may exist at an higher level, they can be accessed only in the given session.

To allow to use a server also from a session different from the one in which it was started, it is necessary to register it with command

```
registerserver [server instance
```

Of course one shall also xasset the plotter variable and beware of the other setting of the environment as described elsewhere.

17.9 Terminating a server

It is care of the user to terminate each graphics server before logging out, in order not to leave hanging processes.

This is achieved from any session in which a server is registered with command

`deleteserver [server instance`

18. Analysis programs

The core XAS package does not provide any "scientific" analysis facilities, but just essential data reduction tools. It is of course possible to write analysis programs which access and use XAS files, and a number of these exists already. They have however to be considered as contributed software and will never be formally supported or mantained as the XAS core.

A list of the main contributed programs is presented in Appendix.

Appendix A : List of XAS environment variables

This list is in alphabetical order. There is also an aid to search this list by subject

---

- **accummode**

  This variable may be used to force accumulations to use non−default quantities on their X and Y axes ("pseudoaccumulations") or to revert to "normal" mode, in which the program uses the default x and y quantities.
  It may assume one of the following values :

  ```
  Normal Pseudo
  ```

- **ask**

  This variable is by default `OFF`. In this case any error while processing the answer to a prompt in the XAS standard user interface will let the program continue or terminate, but will never re−ask for a correction.
  Setting `ask` to `ON` should be intended to enable re−asking.

- **askterminal**

  This variable may be set to a single character, to change the "terminal escape" character (default `!`, an exclamation mark). The terminal escape character is used in XAS command files to force a request from the terminal.

- **axispen**

  A standard XAS colour to be used for plotting axes.
  It defaults to the current `pen` value.

- **bkgpen**

  Reserved for a standard XAS colour to be used for the background of the plot window.

- **calfact**

  This variable is a fractional number (0.0 to 1.0, current default 0.5) used by the gain history generation program to determine the range of channel in

which the fit is performed. This range will be all channels around the peak which have a count level higher than `calfact` times the peak value.

- **caljump**

  This variable is a number of channels (current default 3) used by the gain history generation program to determine the PHA range scanned by the fitting procedure : the procedure will step the peak position (and look for the minimum chi−square) in a range starting from `caljump` channels on the left of the data peak, for `calstep` steps of size `calpass`.

- **calpass**

  This variable is the step (in fraction of a channel, current default 0.05) in the peak position used by the best fit procedure of the gain history generation program.

- **calstep**

  This variable is the number of steps (current default 100) in the peak position used by the best fit procedure of the gain history generation program

---

- **caltype**

  Reserved to set the default filetype extension for calibration files if not specified. Current default `.calib` (not used).

- **cmdstat**

  It may assume the value `PERManent` to allow usage of a given command file with more that one program.

- **command**

  This variable contains the current command file name (the type `.command` is assumed by default if not specified otherwise). The variable retains its value only for a single execution of the first called program (unless cmdstat is set to `permanent`).

- **context**

  This variable specify the current default context for accumulation and display

programs. Its main purpose is to default the type of accumulation and the filetype.
Its value can be one of the following (the first letter is sufficient) :

`Image Spectrum Time Photon`

- **correction**

  This variable is used to enable accumulations to make instrument specific corrections.
  It may assume one of the following values :

  `Disabled Enabled`

- **countaxis**

  This variable is used by the spectral and (HK) time profile accumulations to determine whether the data are expressed as count rates (cts/s, default) or raw counts..
  It may assume one of the following values :

  `Rate Counts`

- **datadir**

  This variable points to the XAS pathname absolute (TBV) or (relative to rootdir) of the subdirectory where one wants to store data files, i.e. in case of SAX, the products of an accumulation (spectra, images, etc.), and all ancillary files (time windows etc.).
  In fact the actual directory name may be a subdirectory of `datadir` constructed from other environment variables (target, date and instrument, according to current order.

- **datapen**

  A standard XAS colour to be used for plotting data.
  It defaults to the current `pen` value (or cycles according to `overcount` in the case of overtraced plots)

- **datastyle**

  This variable determines the "style" of plotting, to be choses among:

- `Solid` all contiguous data points are connected by a solid line, which is broken only at gaps.
- `Histogram` all contiguous data points are plotted as horizontal bars connected by vertical lines, with a "binned" appearance. The line is broken at gaps.
- `Errorbar` all points are plotted with horizontal and vertical error bars. The size of the vertical error bar is the data error in the file, the size of the horizontal error bar is the binsize.
- a diamond error mode is planned
- a marker mode is planned

In all cases the bin x−coordinate in the file is the start of the bin, while the reference point on the plot is the middle of the bin.

- **date**

  This variable is used (optionally) to point to a subdirectory name, intended in the case one wishes to keep data separate by date. The full name of the directory is constructed from rootdir, a "data class" like datadir, fotdir or printdir, and optional `date`, target and instrument specification, and an user−selected order

---

- **echo**

  This variable is by default `ON`. In this case the XAS standard user interface issues prompts to the terminal, and echoes the answer to prompts even when this comes from the runstring or a command file.
  Setting `echo` to `OFF` will disable prompts and echoes.

- **filespectra**

  If this variable is set to `YES` the gain history generation program outputs also a file for each spectrum it is fitting. These spectra have sequential names of the form `temp_`*`nnn`* with a 3−digit sequence number.

- **fitplot**

  Used internally by contributed front−ends to the spectral plotting program to discriminate a "best fit result" plot.

- **flight**

  This variable is provisionally used to toggle between flight (FOT) data format and ground (calibration) data format for ground calibration purposes. It may

assume the values `ON|OFF` (current default) or `OFF` (reserved for instrument team use).

- **fotdir**

  This variable points to the XAS pathname absolute (TBV) or (relative to rootdir) of the subdirectory where one wants to store "telemetry" files, i.e. in case of SAX, the files coming from a FOT.

- **fotorder**

  This variable is used to specify that the name of the FOT directory is built from rootdir, fotdir and eventually target, date, etc. bypassing the standard order. The value of this variable may be of the same form as those of `order`.

- **gainhistory**

  This variable is set to the name of a gain history file to be used by accumulation programs to perform any correction for the dependency of gain with time.

- **hkconvert**

  This variable may be used to disable conversion of HK parameters to engineering units or to revert to the default conversion enabling.
  It may assume one of the following values (first letter is sufficient) :

  ```
  Enable Disable
  ```

- **hms**

  If this variable is set to `Yes` the `tlist` program will display times as formatted `hh:mm:ss.ff` instead of the default raw format.

- **instrument**

  This variable selects which of the SAX instruments is being processed, and directs accordingly the location of the calibration directory.
  It may also be used in conjunction with other environment variables to construct the path name for data storage according to current order.

- **lastcomment**

  Used internally by plotting programs to keep temporarily the status of plotcomment.

- **lastfit**

  Used internally by contributed fitting programs and front−ends to the spectral plotting program to store the name of the last fitted spectrum.

- **lastplot**

  Used internally by plotting programs to store the name of the last data file plotted in a fresh frame.

- **matform**

  This variable selects the format of response matrix files
  It may assume one of the following values (first letter is sufficient) :

  ```
  Ogip Xas
  ```

- **mycaldir**

  The path name of an alternate private directory where an user keeps private copies of calibration files. If set, such files will be used in preference to those in the standard calibration directory. If a file is not present in the private directory, the standard one is used anyhow.

- **nlevel**

  Used to set the number of colour levels used by the image plotting program (default is 16, it shall be less or equal than the number of colours in the current colour table).

- **obschain**

  This variable contains the current observation chain and is set exclusively via the `concatenate` command

---

- **observer**

  This variable is used, if set, to generate an `OBSERVER` keyword in the file headers different from the default (which is the PI name read from a FOT tape directory). This variable may be set to :
  - any string (without imbedded blank) containing the wished name
  - `OVERRRIDE` to use your system login name

- **oldtimeref**

  Used only internally by the time profile plotting program to keep track of the `TIMEREF` reference time (midnight) of the last plotted file, in order to allow overplotting of files with a different reference date. <

- **order**

  This variable is used to control the way directory names are assembled from rootdir, a "data class" like datadir, fotdir or printdir, and optional date, target and instrument specifications.
  It may assume a string value of one to four characters, to be chosen among the following, and given in the desired order :
  - `*` "don’t care" placeholder
  - `C` data class
  - `I` instrument
  - `T` celestial target name
  - `D` observation date

- **overcount**

  Set internally by plotting program to keep a count of how many data files have been plotted in the current frame. It is used to drive the cycling of colours for data overplotting (as such it may be altered willingly by a knowledgeable user).

- **overwrite**

  A transient variable used internally by plotting program to handle overplotting.

- **pen**

  A standard XAS colour to be used as default for all kind of plotting.

- **placeholder**

  This variable may be set to a single character, to change the "placeholder" character (default `.`, a dot). The placeholder character is used in the standard XAS user interface to replace a runstring argument to be defaulted (asked to terminal). Two commas around a null or blank value are equivalent to a placeholder.

- **plotcomment**

  This variables determines the extent of the annotations used on a plot (and indirectly the extent of the plotting region).
  It may assume one of the following values (first letter is sufficient) :

  ```
  Long Short None
  ```

- **plotter**

  This variable is used to direct all plots to a given instance of a plotting server. If not set, it may default to a program−specified default, or fall back to the default `xw1` server.

- **printdir**

  This variable points to the XAS pathname absolute (TBV) or (relative to rootdir) of the subdirectory where XAS will put log files and other ancillary output files intended for printing or looking at.

- **printorder**

  This variable is used to specify that the name of the print directory is built from rootdir, printdir and eventually target, date, etc. bypassing the standard order. The value of this variable may be of the same form as those of `order`.

- **quiet**

  This variable is used to enable additional, moderately verbose, terminal output from several programs. It may assume one of the following values (first letter is sufficient) :

  ```
  Yes No
  ```

- **result**

  This is never set explicitly but only as a result of the execution of some programs, and provides a generic way to pass back some (numeric) results for manipulation in procedures.

- **rootdir**

  This variable points to the XAS pathname of the directory under which all XAS "logical directories" are rooted.

  If one wants to store all data (FOT data, reduced data etc.) under one's own common subdirectory (on the same filesystem) one can use notation

  ```
  xasset rootdir /mydir/mydir/...
  ```

  If one instead wants to use subdirectories on different file systems one MUST use notation

  ```
  xasset rootdir /
  ```

- **selectfile**

  This is either the name of a XAS image (map) or of a SAOImage−style region file to be used for spatial region selection, according to the setting of the selection method.

- **selectmethod**

  This variable selects the method for the spatial region selection.
  It may assume one of the following values (first letter is sufficient) :

  ```
  Map Region
  ```

- **servers_***xx*

  This is never set explicitly but only when starting or registering a graphics server and contains the current list of registered servers of type *xx*.

- **slew**

  This variable is used to select the part of the current observing period to be analysed (remember that observations are numbered 1 to n in EACH part), and may assume the values :
  - `Normal` for normal pointing data
  - `Ingoing` for the ingoing (initial) slew
  - `Final` for the outgoing (final) slew

- **spacecraft**

  This variable is set to the current satellite mission and is used to access calibration data files and/or specific programs. It does not need to be set by the user, the first SAX specific program will set it to `SAX`.

- **target**

  This variable is used (optionally) to point to a subdirectory name, intended in the case one wishes to keep data separate by celestial object. The full name of the directory is constructed from rootdir, a "data class" like datadir, fotdir or printdir, and optional `target`, date and instrument specification, and an user−selected order

---

- **textpen**

  A standard XAS colour to be used for plotting text.
  It defaults to the current `pen` value.

- **timeaxis**

  This variable controls the annotation of the x−axis in plots, when the x quantity is time. The annotation may be in elapsed seconds from midnight, or in hours (hh:mm:ss).
  It may assume one of the following values (first letter is sufficient) :

  `Seconds Hours`

- **timeunits**

  This is used by accumulation programs to determine the units used for time when outputting time profiles (times are always elapsed since the midnight of observation start).

It may assume one of the following values :

`Second MILlisecond MICrosecond`

NOTE THAT ONLY USAGE OF "seconds" HAS BEEN TESTED.

- **timewindow**

  This variable contains the name of the current time window file.
  Unset it (or set it to blank) to disable time windowing (as default).

- **tquantity**

  This variable is used to select an alternate quantity to be plotted, when plotting time profiles. Simple time profiles contain only a scalar `DATA` column, and it is not necessary to do anything.
  In case of more complex time profile this variable may be used to select alternate columns as follows :
  - `Deadtime` to plot the deadtime column if present
  - `Other` to plot the "other data" column if present
  - `n` to plot the n−th element of a multidimensional `DATA` column

---

- **verbose**

  This variable is used to enable additional, extremely verbose, terminal output from sa few programs. It may assume one of the following values (first letter is sufficient) :

  `No Yes`

- **viewport**

  This is a xasplot variable, whose values is a comma−separated list of four numbers in the range 0.0−1.0 :
  - the lower X coordinate of the plotting viewport in normalized coordinates (i.e. as a fraction of the plotting page)
  - the upper X coordinate of the plotting viewport
  - the lower Y coordinate of the plotting viewport
  - the upper Y coordinate of the plotting viewport

- **XASTOP**

  This is NOT a XAS environment variable, but a system environment variable

which must be set to the top directory holding the XAS distribution tree. This setting shall be done (e.g. in a login file) before using any XAS command.

- **xaxis, yaxis**

  These are xasplot variables, whose values is a comma−separated list of two numbers, corresponding to the extrema of the X or Y axis in the current world units.
  It may also assume the value `AUTO` (default) to let the program set the extrema automatically, or to `NICE` for a better (rounded) automatic choice.

- **xquantity**

  This variable indicates the mnemonic name of the quantity to be used for accumulation on the x−axis (for images) or the only axis (for spectra).

- **xscale, yscale**

  This variable determines whether the X or Y axis is linear or logarithmic. It may thererfore assume one of the following values :

  ```
  LIN LOG
  ```

- **xyaxis**

  This is a xasplot variable, whose values is a comma−separated list of four numbers. It is used internally to keep track of the X− and y−axis extrema in the last plot.

- **yaxis**

  See `xaxis`.

- **yquantity**

  This variable indicates the mnemonic name of the quantity to be used for accumulation on the y−axis (for images).

- **yscale**

  See `xscale`.

- **zscale**

  This variable is intended to give the user a choice of the way the "Z" scale is chosen when displaying an image. The foreseen values are :
  - `Histogram` equalization mode (only current form)
  - `LIn`ear scale
  - `LOg`arithmic scale
  - `User` selected scale
  - `File` : scale taken from an external file

Appendix B : List of XAS environment variables

This list is ordered by subject. We provide here only LINKS to the full description in the main list in alphabetical order.

General purpose

These variables are used to control the XAS user interface

- command : name of a command file used instead of runstring or terminal
- cmdstat : controls if command file setting is retained
- quiet : may allow no or moderate verbosity of output
- verbose : allows large verbosity of output

The following user−interface variables are used seldom.

- echo : enables/disables echo of prompt and answers
- ask : enables reasking in case of errors
- askterminal : changes terminal escape character
- placeholder : changes runstring placeholder character

- result : used to return calculation results to the shell
- hms : allows times to be displayed in hh:mm:ss format
- observer : sets the observer name in the file headers
- flight : reserved for use with ground calibration data

File and directory location

These variables are used to determine the file name or type, or the name of the program to be run, or to locate the file.

- context : set default filetype extension
- spacecraft : satellite mission name
- instrument : current instrument being analysed
- slew : select part of observing period

These variables are used to build the directory names where files are searched or created.

- rootdir : the top logical root
- order : the default order of components used to build path names
- fotdir : the second level path for FOT data storage
- fotorder : an alternate order of components used to build FOT path names
- datadir : the second level path for reduced data storage
- date : a lower level path for data arranged by date
- target : a lower level path for data arranged by source name

- printdir : the second level path for ancillary printouts
- printorder : an alternate order of components used to build print path names

- mycaldir : an alternate private directory for calibration data
- caltype : default filetype extension for calibration files

Accumulation control

(see also file and directory location)

These variables are of common use, and are used to select part of the data, or to control the kind of processing done.

- obschain : stores the current chain of observations
- timewindow : the current time window file
- correction : whether instrument−specific corrections are enabled
- gainhistory : the current gain vs time correction file
- selectfile : the current spatial selection file (if any)
- selectmethod : whether the above file is a region or image file
- countaxis : whether bins contain count/sec or raw counts
- matform : whether response matrices are in XAS or OGIP format

These other variables are used less often to allow variants in accumulations

- accummode : to use other than X,Y or PHA for images and spectra
- hkconvert : control HK conversion to engineering units
- xquantity : what is on the X axis of an image or spectra
- yquantity : what is on the X axis of an image
- timeunits : select unit for times

These are used for fine tuning of the gain history program(s)

- calfact : select an intensity cut of the spectrum
- caljump and
- calpass and
- calstep : are used together to select the fitting range
- filespectra : allows output of individual spectra

Graphics

These are used very often for general settings.

- plotter : choose current plotting server
- plotcomment : amount of plot annotations
- timeaxis : time axis labels in seconds or hours
- tquantity : which data column to plot vs time

These are used to control the plotting axes.

- xaxis and
- yaxis are the ranges of X− and Y−axis (xasplot)
- xscale and
- yscale controls whether the X− and Y−axis are linear or logarithmic

These are used for colour selection.

- axispen : the colour for axes
- bkgpen : the colour for the background
- datapen : the colour for data
- textpen : the colour for text
- pen : the default colour for everything not defined otherwise

- nlevel : the number of levels for image displays
- zscale : the kind of scaling for image displays

The following variables are used internally

- overcount : the number of overplotted data files
- overwrite : transient flag for overplotting
- servers_xx : list of registered graphics server
- viewport : current plotting viewport
- oldtimeref : reference time of last light curve plotted
- lastcomment : annotation of last plot
- lastplot : data file of last plot
- fitplot : flag for best fit plotting
- lastfit : last spectrum fitted

See also elsewhere for a list of reserved variables (this page has restricted access).

Appendix C : List of XAS commands

This list is in alphabetical order. There is also a list by subject.
For each command the link points to the page (or section) where the appropriate command syntax is given.

We list here all commands, including hidden commands generally not called by the general user : these are listed indented, outside of alphabetical order, after the command from which they are called. Note that the `HISTORY` keyword may refer to the hidden command name.

- `accumulate` front end accumulation program to:
  - `mecsgainaccum` MECS gain history
  - `mecsmaccum` MECS response matrix
  - `pdsmaccum` PDS response matrix
  - `saxiaccum saxgaccum` spatial and generic (pseudo−)images
  - `saxsaccum saxhaccum` spectra and (generic) histograms
  - `saxhkaccum` HK time profiles
  - `saxpaccum` photon lists
  - `saxtaccum` time profiles
- `check expconf` experiment configuration summary
- `concatenate` observation concatenation
- `createserver` start one of the following graphic servers :
  - `bwserver` front−end (black & white) to PostScript server
  - `c2server` front−end (colour Level 2) to PostScript server
  - `cpserver` front−end (colour) to PostScript server
  - `psserver` back−end PostScript server
  - `xwserver` X−window (X11) server
- `degeneralize` assign XAS type to generic tabular file
- `deleteserver` terminate graphics server
- `display` front−end to graphical display of:
  - `idisp` images
  - `splot` spectra
  - `tplot` all kinds of time profiles
  - `wplot` time windows
- `fotfile` (fotfile_2 fotfile_3) FOT filing program (in 3 steps)
- `fromfits` conversion of FITS files to XAS
- `fromogip` conversion of OGIP FITS files to XAS
- `header_edit` edit XAS file header keywords
- `hlist` list XAS file header
- `importback` (Unix only) import XAS environment into Unix environment
- `insertboundary` (contributed) insert channel boundaries in keV into a XAS spectrum
- `iwindow` generate intensity windows
- `localize` convert XAS files between different operating systems
  mergewindow merge time windows

- `overfit` (contributed) shortcut to overplot a fitted spectrum
- `overres` (contributed) shortcut to overplot residuals of fit
- `overtrace` graphical display on the current plot
- `psdgrppha` rebinning of PDS spectra
- `psdoprspe` operations on PDS spectra
- `plotdata` (contributed) shortcut to plot last fitted count spectrum
- `plotfit` (contributed) shortcut to plot last fitted photon spectrum
- `plotres` (contributed) shortcut to plot residuals
- `readlut` read content of current colour table
- `registerserver` register a graphic server in current session
- `saodisp` produce runstring for displaying an image with SAOimage
- `saxauxcalc` generate auxiliary quantity time profiles
- `tappend` (contributed) append one time profile after another
- `tlist` list content of XAS data file
- `tofits` conversion of XAS files to FITS
- `toogip` conversion of XAS files to OGIP FITS
- `twindow` generate time windows
- `writelut` write (part of) colour table
- `xaccumulate` front end cross−accumulation program to:
  - `xiaccum xgaccum` spatial and generic (pseudo−)images
  - `xsaccum xhaccum` spectra and (generic) histograms
  - `xpaccum` photon lists
  - `xtaccum` time profiles
- `xasplot` assign values to some graphics XAS environment variables
- `xasset` assign values to XAS environment variables
- `xasunset` delete XAS environment variables

Appendix D : List of XAS commands

This list is ordered by subject. There is also a master list in alphabetical order to which the reader is referred for details.

Environment handling

- `xasset` assign values to XAS environment variables
- `xasunset` delete XAS environment variables
- `xasplot` assign values to some graphics XAS environment variables
- `importback` (Unix only) import XAS environment into Unix environment

---

FOT filing

- `fotfile` (fotfile_2 fotfile_3) FOT filing program (in 3 steps)
- `check expconf` experiment configuration summary
- `concatenate` observation concatenation

Data accumulation

- `check expconf` experiment configuration summary
- `accumulate` front end accumulation program to:
  - `mecsgainaccum` MECS gain history
  - `mecsmaccum` MECS response matrix
  - `pdsmaccum` PDS response matrix
  - `saxiaccum saxgaccum` spatial and generic (pseudo−)images
  - `saxsaccum saxhaccum` spectra and (generic) histograms
  - `saxhkaccum` HK time profiles
  - `saxpaccum` photon lists
  - `saxtaccum` time profiles
- `saxauxcalc` generate auxiliary quantity time profiles

- `twindow` generate time windows
- `iwindow` generate intensity windows
- `mergewindow` merge time windows

- `xaccumulate` front end cross−accumulation program to:
  - `xiaccum xgaccum` spatial and generic (pseudo−)images
  - `xsaccum xhaccum` spectra and (generic) histograms
  - `xpaccum` photon lists
  - `xtaccum` time profiles

Data manipulation

- psdgrppha rebinning of PDS spectra
- psdoprspe operations on PDS spectra
- insertboundary (contributed) insert channel boundaries in keV into a XAS spectrum
- tappend (contributed) append one time profile after another

---

Data handling

- header_edit edit XAS file header keywords
- hlist list XAS file header
- tlist list content of XAS data file

Graphics

- createserver start one of the following graphic servers :
  - bwserver front−end (black & white) to PostScript server
  - c2server front−end (colour Level 2) to PostScript server
  - cpserver front−end (colour) to PostScript server
  - psserver back−end PostScript server
  - xwserver X−window (X11) server
- deleteserver terminate graphics server
- registerserver register a graphic server in current session

- display front−end to graphical display of:
  - idisp images
  - splot spectra
  - tplot all kinds of time profiles
  - wplot time windows
- overtrace graphical display on the current plot
- xasplot assign values to some graphics XAS environment variables
- saodisp produce runstring for displaying an image with SAOimage

- readlut read content of current colour table
- writelut write (part of) colour table

- plotdata (contributed) shortcut to plot last fitted count spectrum
- plotfit (contributed) shortcut to plot last fitted photon spectrum
- plotres (contributed) shortcut to plot residuals
- overfit (contributed) shortcut to overplot a fitted spectrum
- overres (contributed) shortcut to overplot residuals of fit

---

Format conversion

- `localize` convert XAS files between different operating systems
- `saodisp` produce runstring for displaying an image with SAOimage
- `degeneralize` assign XAS type to generic tabular file
- `fromfits` conversion of FITS files to XAS
- `fromogip` conversion of OGIP FITS files to XAS
- `tofits` conversion of XAS files to FITS
- `toogip` conversion of XAS files to OGIP FITS

Appendix E: List of program arguments

This is an alphabetic list of all possible program arguments using the abbreviation name used in the main help pages.

- **abspath**

  This notation points to an absolute XAS pathname of the form

  `/dir1/dir2/dir3`

- **arffile**

  is the name of an ancillary response file, o "none" if no ARF generation is requested. Usually no filetype is requested (if specified overrides the default setting of ".arf"). This argument has no effect in the case of XAS matrix format.

- **binning**

  The way the binning factor for a quantity can be specified depends on the quantity itself :

  `n`

  for any quantity other than `TIME` one just gives a binning factor (zoom factor), 1 to mantain original channels/pixels, n>1 to have bins which are n channels wide

  `binsize`

  if the quantity is `TIME` one gives a binsize in seconds, expressed as a REAL number. This value must be a multiple of the proposed default. If the user does not comply, the value is adjusted.

- **cfile**

  This notation indicates the name of a command file. Usually one provides only the filename and lets the filetype default to `.command`, and the path default to the current directory.

- **colour**

  A XAS colour may be one of the fundamental colours used for vector plotting (in practice a positive number from 0 to 7), or a conventionally negative number, which points into the current colour table (e.g. −12 means the 12th location of the current table).

- **commentflag**

  This flag controls whether FITS inline comments are translated into XAS comment keywords or are omitted. It may take either values

  `Yes No`

- **context**

  This argument specifies the current default context for accumulation and display programs. Its main purpose is to default the type of accumulation and the filetype.
  Its value can be one of the following (the first letter is sufficient) :

  `Image Spectrum Time Photon`

- **datatypes**

  For the purpose of the `fotfile` program one this field shall be a plus−separated concatenation of file classes and types, with optional negation, e.g. of the form

  `class+class+type+NOclass+NOtype`

  where class refers to any of the following classes of file types

  `A`

  the `tapedir` file

  `B`

  the ephemeris, attitude and OBT−UTC conversion files

  `C`

  all spacecraft HK files common to all instruments

`D`

all instrument directories (`instdir, obsdir, expconf`)

`E`

all instrument HK files

`F`

all instrument science data files

`ALL = A+B+C+D+E+F`

all of the above

`DEF = A+B+D+E+F`

default configuration

and type refers to any filetype as shown in the `tapedir`.
The above shall allow selective constructs like :

```
ALL+NOB
B+NOephemeris
D+E+F+obt_utc
```

- **file**

  A generic file name. In case of XAS data files this may be an image (or a response matrix), spectrum, time profile (of any sort), or photon list.

- **fitsfile**

  A data file in the FITS format.

- **gainfile**

  This notation indicates the name of gain history time profile. Usually one provides only the filename and lets the filetype default to `.time`, and the path default to the standard XAS data path.

- **ifile**

  This notation indicates the name of a XAS image file. Usually one provides only the filename and lets the filetype default to `.image`, and the path

default to the standard XAS data path.

- **include**

  This notation is used for photon accumulation to specify if a given quantity has to be included in the output file or not. It may take either values

  `Yes No`

- **instance**

  This argument is a sequence number n, indicating the n−th instance (copy) of a given graphics server program.

- **instrument**

  One of the SAX instrument codes :

  - `lecs`
  - `mecs`
  - `hpgs`
  - `pds`
  - `wfc`

- **instrument**

  For the purpose of the `fotfile` program the SAX instrument code is one of the following two−letter codes :

  `XX`

  "any" instrument, i.e. the files for "all" instruments (indicated as ** in the tape directory), plus the files for the first instrument appearing on tape whatever it is

  `LE`

  LECS

  `ME M1 M2 M3`

  MECS, respectively all units, or only unit 1, 2 or 3

  `HP`

  HP−GSPC

`PD P1 P2 P3 P4`

PDS, respectively all units, or only unit 1, 2 , 3 or 4

`W1 W2`

WFC1 or WFC2 respectively

- **inslewobs**

  An extended observation range of observations in the initial (ingoing) slew manoeuvre

---

- **hkname**

  This corresponds to the mnemonic code of an housekeeping (HK) parameter. Mnemonic codes are listed in the instrument PCF (Parameter Characteristics File).

- **keyword**

  The name of a file header keyword is a string of no more than 8 characters.

- **keytype**

  The type of a file header keyword is coded as follows:
  - `C` for a character string
  - `I` for a 16−bit integer (INTEGER*2, use discouraged)
  - `J` 32−bit integer (INTEGER*4)
  - `R` 32−bit floating point (REAL*4)
  - `D` double precision floating point (REAL*8)

- **logflag**

  This flag controls whether an additional log file is produced with some information on the data conversion. It may take either values

  `Yes No`

- **meunit**

  The number (1,2 or 3) of the MECS unit.

- **ncol**

  The number of colours for colour table manipulation. It is a number with absolute value from 0 to 256 (or less, according to LUT size). A positive value conventionally refers to the particular graphics server's LUT, while a negative value conventionally refers to the full X colourmap.

- **ncounts**

  The number of counts to be accumulated in each spectrum.

- **obsrange**

  is an "extended" range of observation numbers i.e. the concatenation of
  - single observation numbers (e.g. 1, 3)
  - observation intervals in dash−separated form (e.g. 5−7 for 5 to 7)

  using a "plus" as concatenation operator, e.g. to concatenate the above

  ```
  1+3+5−7
  ```

  An observation chain of "0" means "none", and "1−99" means "all".

- **outslewobs**

  An extended observation range of observations in the final (outgoing) slew manoeuvre

---

- **outtype**

  This argument specifies the output file type for the check_expconf command and can be one of the four values (all can be abbreviated to the first letter)

  - `Ascii`
  - `Ps`
  - `Colour`
  - `Graphics`

  corresponding respectively to a named ASCII file, a named black&white

Postscript file, a named colour Postscript file or a graphic X−window server instance

---

- **path**

  This notation points to XAS pathname which can be either absolute or relative, i.e. of the form

  ```
  dir1/dir2/dir3
  ```

- **packet**

  This corresponds to the code of a SAX FOT telemetry packet. This argument needs to be supplied ONLY if more than one packets are eligible for the particular accumulation requested, otherwise it is resolved internally by the program.

- **pfile**

  This notation indicates the name of a XAS photon list file. Usually one provides only the filename and lets the filetype default to `.photon`, and the path default to the standard XAS data path.

- **quantity, xquantity, yquantity**

  This indicates the names of quantities present for an event, and which can be used for accumulations (e.g. on the x and y axes). The names are mnemonic codes, contained in the packetcap files, like :

  ```
  X Y PHA TIME BURSTLENGTH ...
  ```

- **range**

  These arguments specify the range for the non−interesting quantities of an accumulation, that is, the ones used for selection.
  In the case of photon lists, this applies to all quantities, interesting and uninteresting.
  The notation for ranges is described below.

- **range**

  The way a range for a quantity can be specified depends on the quantity itself :

  `lowvalue highvalue`

  for any quantity other than `TIME` one just gives the low and high values in channels or pixels

  `starttime endtime`

  if the quantity is `TIME` one gives start and end times in UT, using the full date (year, month number, day) and time (hours, minutes, seconds), with notation

  `yyyy mm dd hh mm ss`

- **records**

  A range of records (start and end) in a file.

- **rfile**

  This notation indicates the name of SAOimage−style region file. Usually one provides only the filename and lets the filetype default to `.region`, and the path default to the standard XAS data path.

- **rgb**

  An RGB colour specification is a triplet of real numbers (each one in the range 0.0 to 1.0) indicating the level of Red, Green and Blue.

- **rmffile**

  is the name of a response matrix file, o "none" if no RMF generation is requested. Usually no filetype is requested (if specified overrides the default setting), and the appropriate one (".rmf" or ".matrix") is constructed according to the file format selected.

- **servinst**

  This argument fully identifies a particular graphic server instance, and is formed by a two−letter code (corresponding to the server type) and a number n (corresponding to the n−th instance or copy of a server) without intervening

blanks.
E.g. `xw2` is the 2nd instance of an X−window server, `bw1` is the 1st of a black&white Postscript server, etc.

- **server**

  This argument is a two−letter code indicating the type of graphics server. The currently defined values are

  - `xw`
  - `bw`
  - `cp`
  - `c2`

  corresponding to the X−Window server, the Black&White Postscript server, the Colour Postscript server, and the Colour Level 2 Postscript server.

- **sfile**

  This notation indicates the name of a XAS spectral file. Usually one provides only the filename and lets the filetype default to `.spectrum`, and the path default to the standard XAS data path.

- **startc**

  The start colour for colour table manipulation. It is a number from 0 to 256 (or less, according to LUT size).

---

tape

This is a system−dependent tape device name (currently corresponding to a local tape drive attached to the system where the `fotfile` program is run) e.g.

```
/dev/nrmt0l  /dev/nrst0  MUA0:
```

- **targetobs**

  An extended observation range of observations in the normal pointing on the celestial target being observed

---

- **tbin**

  This indicates a binsize in seconds, expressed as a REAL number. This value must be a multiple of the proposed default. If the user does not comply, the value is adjusted.

- **tfile**

  This notation indicates the name of a XAS time profile. Usually one provides only the filename and lets the filetype default to `.time`, and the path default to the standard XAS data path.

- **trange**

  A time range is always indicated as

  ```
  starttime endtime
  ```

  giving start and end times in UT, using the full date (year, month number, day) and time (hours, minutes, seconds), with notation

  ```
  yyyy mm dd hh mm ss
  ```

- **value**

  The value of a XAS environment variables can be any string, provided it does not contain imbedded blanks. An array of value can be represented as a comma−separated list (but this is not accessible in the normal way). A numeric value is also passed as a string. A blank value is equivalent to a null value (i.e. to unset the variable)

- **value**

  The value of a XAS file header keyword can be
  - a string of no more than 68 characters not containing imbedded blanks
  - one or more integer numeric values
  - one or more (single precision) floating point numeric values
  - one or more double precision numeric values

- **variable**

  The name of a XAS environment variables can be any string (excepted the word `FROM`)

- **xastype**

  For the purpose of most programs valid XAS data types are of the the form
  - image
  - spectrum
  - photon
  - time

- **xbin, ybin**

  These arguments specify the binning factor for the interesting quantities (along x and eventually y) of an accumulation.
  The notation for binning is described elsewhere

- **xfile**

  A XAS environment file which can be fed into `xasset`, consisting of a number of lines, each one of the form

  `variable value`

- **xrange, yrange**

  These arguments specify the range for the interesting quantities (along x and eventually y) of an accumulation.
  The notation for ranges is described elsewhere

- **xquantity, yquantity**

  See quantity.

- **wfile**

  This notation indicates the name of a time window file. Usually one provides

only the filename and lets the filetype default to `.window`, and the path default to the standard XAS data path.

Appendix F : Contributed additional software

IT IS PLANNED TO list here additional XAS programs and additional software related to XAS which is not part of the core package, is not formally supported or mantained with or as the core package, and is not distributed within the core package.

- **IDL vanilla interface**
  - xasread and xaswrite, a small group of IDL procedures to
    - read XAS files (data and keywords into an IDL structure
    - rewrite the IDL data structure into an existing XAS file
    - update a XAS file header
  - xasasc a csh script which adds an header to an ASCII tabular file, and an IDL procedure to read it into an IDL structure
- **IDL xasplot widget GUI**

  This plotting environment has been developed at ITESRE and the relevant documentation (if any) will be mantained there.
- **Image reduction**
  - `centroid` image barycentre
  - `psf` PSF and LSF computation
  - `smooth_square`, `smooth_gauss` and `smooth_gauss2` image smoothing programs
- **Spectral analysis**
  - `insertboundary` inserts keV channel boundaries in a spectrum
  - `scount` cursor−aided integration of a spectrum
  - `integra` integration of analytic spectral form
  - `soperate` algebra on spectral files
  - `gausspeak` fitting of Gaussian lines
  - `convolve`, `fit` and `grid`, a set of spectral convolution and fitting programs
  - `plotdata`, `plotfit` , `plotres`, `overfit` and `overres`, shortcut commands to plot the results of the above fits with the usual XAS graphics
- **Timing analysis**
  - `tappend` appends time profiles one after the other
  - `varies` mean and variance of time profile
  - `dtaccum` and `dtaccum` produce a time profile or a statistics of SAX packet header time information

- `fourdeem` a Direct Fourier Transform program (Deeming's method, other programs will be converted later).

Printing remark

The PostScript version of this manual has been obtained as a snapshot of the HTML files at `http://sax.ifctr.mi.cnr.it/Xashelp/` using a custom shell script `makeps`. This script drives NCSA X−Mosaic to display all pages in the appropriate order. One then uses X−Mosaic Save As ... PostScript facilities manually to convert HTML to PostScript into a scratch file. The script then uses a combination of awk and sed programs to remove the buttons at the end of each page and change the pagination numbers. Finally the psutils by Angus Duggan are used to combine two pages into one to save paper.

Explicit liber XAS

Si LV ponatur
tunc CIV iungatur
scriptor iste nominatur
