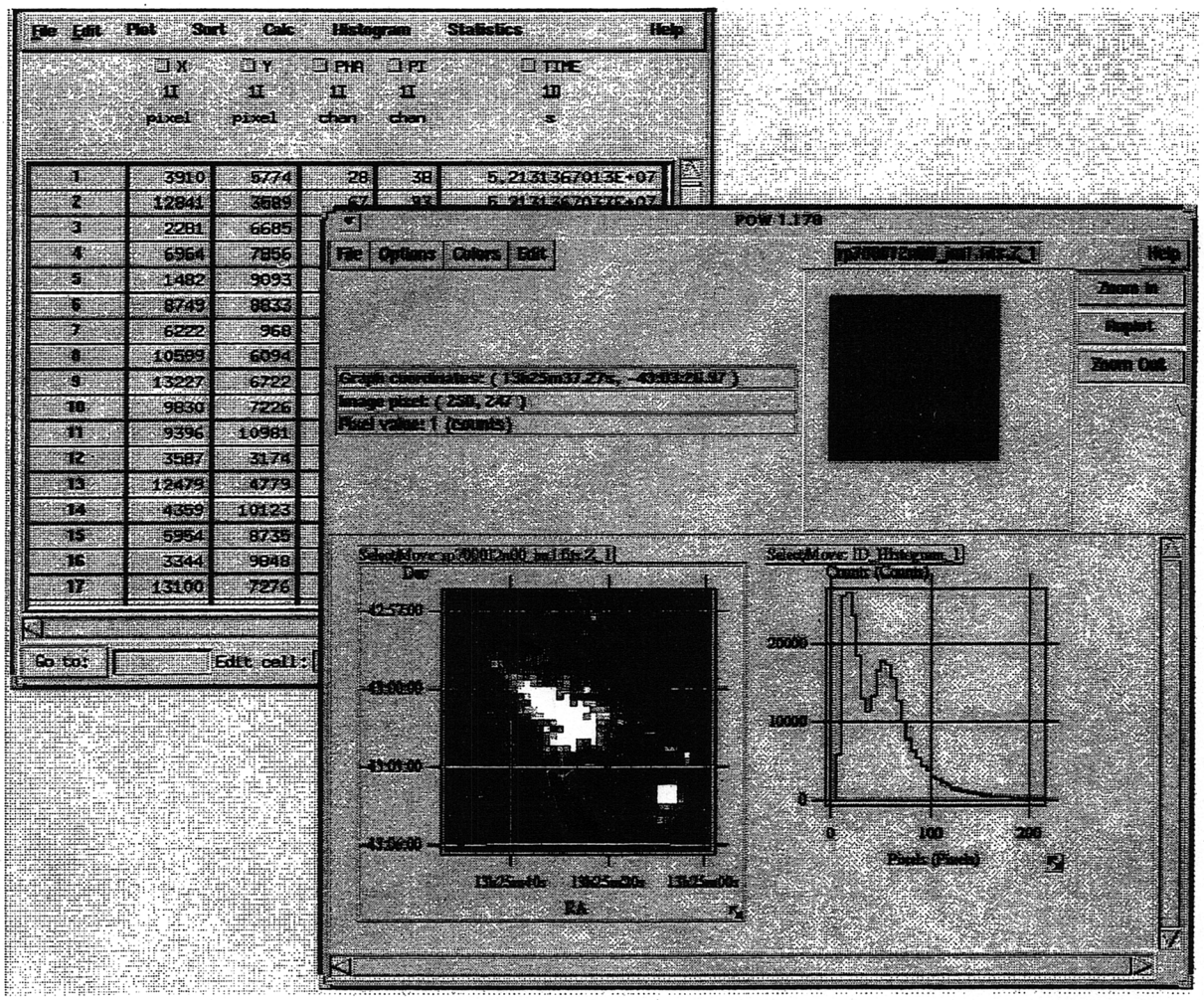
Legacy

The Journal of the High Energy Astrophysics Science Archive Research Center

Number 7 **June 1998**



NASA
Goddard Space Flight Center

HEASARC

AXAF Data and Data Manipulation Software: The ASC Data Model

J. McDowell, M. Noble, M. Elvis
SAO

1. Introduction

AXAF, the Advanced X-ray Astrophysics Facility, is scheduled for launch in December 1998. The AXAF Science Center (ASC), located in Cambridge, Massachusetts at SAO and MIT, is developing software for the analysis of AXAF data. While this software can be used as a set of FTOOLS-like programs or as IRAF tasks, it also has built into it additional infrastructure that makes data analysis easier, especially for the rich data cubes that the AXAF CCD Imaging Spectrometer (ACIS) will provide. The ASC software infrastructure builds on heritage from the IRAF-based PROS analysis system and on ideas from the FTOOLS package. Programs access this infrastructure via a new data I/O library, the 'ASC Data Model' (ASCDM) library.

The goals of the ASCDM library are:

* to provide an easy way to filter and bin data files at runtime, extending the functionality provided in the PROS system.
* to provide a common view of multiple file formats, including FITS, IRAF/QPOE, and ASCII files, and isolate the details of those formats from the calling program.
* to provide a programming interface to the data which is higher level than basic I/O libraries like CFITSIO, but is still generic.
* to support encoding a greater level of self-description in the data files while retaining back compatibility with existing conventions.

These goals may seem rather abstract, but the following sections try to give a feel for why they are useful, as well as some detail on how they are achieved. First we discuss filtering and binning, then the ability to use multiple file formats. Next we step back to explain how the data model operates by creating an abstract representation of the data, in fact a model of the data - a 'data model'

2. ASCDM Filtering

The idea behind ASCDM run-time filtering is that, in any program using the ASCDM interface, you can pass a 'virtual file' where an input file is expected. For instance, suppose that a source detection program, DETECT, expects a 2-dimensional image (counts versus x and y):

```
detect input=im.fits output=result.fits
```

but your full resolution 32k square HRC image is too big to fit in memory. You can instead pass it a subset of that file with only x values in a certain range, without first creating any intermediate disk file. The program sees the 'virtual' file you have specified:

```
detect input=im.fits[x=100:200] output=result.fits
```

Or, you can create the image on the fly from an event list, selecting a particular pulse height range at the same time, with

```
detect input=evt.fits[events][pha=20:120][bin x,y] output=result.fits
```

In fact the ASCDM will support the full spatial 'region' filtering used in PROS:

```
detect input=evt.fits[events][(ra,dec)=circle(14:07:23 -00:14:23 3')][bin x,y]
    output=result.fits
```

This selects only photons whose celestial positions lie within 3 arcmin of the quoted coordinates.

The virtual file is implicitly created from a true disk file by filtering or binning, but does not get created on disk or even formatted into a Unix pipe. Instead, I/O calls in the program return records of the virtual file by internally accessing the actual disk file and figuring out which data passes the filter.

The full syntax of a data model filter is a filename followed by a series of optional qualifiers in square parentheses:

```
filename[block][filter][columns][binning]
```

The block qualifier specifies which section of the file to use; in FITS, it specifies an HDU (Header Data Unit). The FTOOLS syntax of [n], where n is the HDU number, is accepted, but the name (EXTNAME value) of the HDU is equally valid and in many cases will be more intuitive. If the block qualifier is omitted, the first 'interesting' block will be used - the first FITS HDU for which NAXIS is nonzero.

The filter qualifier specifies which rows of a table or pixels of an image to use, by filtering on the values of columns or axes. A more complicated filter example is [pha=3:20,time=100:200,250:300,450:500,ccd_id=3]. The filter on each variable is expressed as a series of acceptable ranges. Logical operators (&, <, etc) may also be used, as described in the detailed ASCDM documentation.

The columns qualifier specifies which columns from a table should be used; it cannot be used with images. It can also be used to rename columns, in case you have a program which expects columns with particular names.

The binning qualifier makes an image from a table, binning on any collection of columns, optionally specifying the ranges and binning factors:

```
[bin x=100:612:4,y=512:1024:4,time=8441014.3:8441420.8:20.2]
```

specifies making a 3-dimensional x,y,time cube in which each x,y pixel is 4 input pixels, and each time pixel is 20.2 seconds.

The ASCDM will come with a set of basic file manipulation tools, of which the most important is dmcopy. This tool, in the spirit of Unix's cat, performs a variety of important tasks: it copies a virtual file to a real file, optionally changing the kernel (file format). It thus provides filtering and binning capability, and also format conversion.

3. Multiple File Format Support

Although FITS files will be used for all archival and user data distribution purposes, for run time analysis, and for compatibility for users working in the IRAF environment, we want our software to support the use of QPOE event files and IRAF IMH images. We do this by making a common abstraction of the data, and deciding which structures in QPOE and FITS map to various parts of the abstraction. This abstraction is what we mean by a 'data model'. The various supported file formats are called 'kernels'. The ability to use one set of tools to read multiple formats removes the need for conversion tools and greatly improves interoperability.

ASCDM abstraction	FITS	QPOE/IMH
Dataset	File, may have many HDUs	1 or more files
Block	HDU	Single file
Table	BINTABLE	QPOE
Image	IMAGE	IMH
Column descriptor	TTYPE etc	QPOE structure
TIME subspace	GTI HDU	deffilt keyword
Other subspace	keywords	keywords
Coord descriptor	TCRVL keys	Opaque binary MWCS

Table 1. Examples of ASCDM abstractions in different formats.

Even if an application expects only to use FITS, the differing details of how to store coordinate systems for images, table columns, images in table columns, etc., are of interest only to the true FITS aficionado (who should see Pence et al 1995, *http://legacy.gsfc.nasa.gov/docs/heasarc/ofwg/docs/general/wcs_keywords/wcs_keywords.html* in this particular case). We believe that others will be happy to have a uniform interface to handle and shield them from such information, and make their programs portable over a variety of formats.

The third kernel we expect to support is an ASCII kernel, which we anticipate will be most useful for importing user tabular data. The exact format has not been finalized, but will probably involve optional free-format FITS-style header keywords and will also support the 'rdb' Unix database format. We will also provide a null 'template' kernel which can be used as a starting point for those who wish to add further formats.

For another data format to be compatible with the ASCDM, we need it to support storage of tabular and image data, and have the ability to store arbitrary header metadata for each table and image. Conversely, more complicated structures (e.g. recursive hierarchical structures) may not be fully mapped to the present ASCDM design.

4. The ASCDM view of data

The ASCDM interface works in terms of a certain description of the structure of the data independent of the data format. We refer to this as the ASCDM's abstraction of the data.

Fundamentally we support only two types of data - images and tables. An image can have 1, 2, 3 or N dimensions but is always an array of values of a well defined size. So a sky intensity image, a pulse height spectrum and a Fourier spectrum versus energy versus time are all 'images'. A table, instead, is a set of data points (rows) with a bunch of properties (columns). An event file, a catalog of stellar magnitudes, or for that matter a listing of individual census returns are all tables. A raw image or table is not much use however without two types of supporting data: a header, which contains any extra information needed to interpret the data, and a data subspace. The data subspace, a new concept, describes in a general way how the data in the table or image were selected. It unifies the concepts of Good Time Intervals and of filters on other quantities. We introduce the word 'block' to refer to a unit of data which may be either a table or an image together with its supporting header and data subspace. This block usually corresponds to a single FITS HDU.

The conceptual 'data' section of a block comprises the actual image or table data, including information about the table columns or image axes which in the FITS file is stored in header keywords.

The ASCDM 'header' contains any extra information (metadata) not directly describing the main data, and corresponds to those FITS header keywords which don't impart information about the structure of the binary data section. For example, the TTYPE keyword of a FITS header is not part of the ASCDM block header, because it describes the structure of the block in a format-specific way. The implication of this is that in our interface you control TTYPE values in a file by changing the properties of the table block columns, never by writing anything to the ASCDM header. The INSTRUME FITS keyword, however, is simply treated as auxiliary data, and corresponds to an INSTRUME keyword in the ASCDM block header.

When writing the INSTRUME keyword to a file using ASCDM, you use an explicit keyword writing routine (dmKeyWrite_c). The ASCDM keywords are generalized versions of the usual FITS keywords, as we describe below (Section 6.1). To make a TTYPE keyword appear in a FITS file, you instead use a dmColumnCreate routine. It's

important to make this distinction, because in another file format the names of columns may be stored in some other way, so the fact that FITS uses a TTYPE header keyword to do this job shouldn't appear in a high level program.

Another kind of information that FITS stores mostly in header keywords is the description of how the data were selected, the data subspace (imagined as specifying a subset of the infinite-dimensional space of all data). This is particularly important for event data: a list of events is not very helpful if you don't know what the range of accepted times, pulse heights, and detector positions were.

This information is stored in heterogenous ways in X-ray FITS data, with GTI (Good Time Interval) tables for time, CHANMIN/MAX keys for pulseheight, and no standard method for spatial regions. The ASCDM provides a generic interface to say 'the valid ranges on this arbitrary quantity are the following' - this is the data subspace, which is conceived of as a table of filters, each column of which is a particular quantity being filtered on. In particular, programs that access a data set that knows how it was selected can use that information to choose appropriate calibration information, such as the correct point spread function to calculate the enclosed energy.

Because the ASCDM is designed to support multiple formats, the preferred way to handle an event list with a GTI table is to access it as an element of the event table's data subspace rather than opening it explicitly as a table. In IRAF's QPOE format, the GTIs are stored as strings in the QPOE header rather than as a separate table. Because the GTIs are in some sense an intimate part of the event data, it is plausible that they also might not be stored in separate tables in future formats. Therefore, the ASCDM interface allows you to get and set the GTI values with a call that does not explicitly require there to be a separate table. Nevertheless, in order to permit full access to writing header keywords to the table, the GTI may also be accessed directly as an ASCDM table block when using the FITS kernel.

5. Subroutine Interface

The basic ASCDM subroutine interface is written in ANSI C. A set of Fortran wrappers is also provided. The routines fall into the following categories:

- Dataset routines, including getting basic file properties and positioning within a file.
- Block routines, including opening and creating tables and images, and positioning in the block header.
- Table routines, positioning in tables and reading and writing whole rows or groups of rows.
- Column routines, creating and manipulating table column structures.
- Key routines, handling I/O of header keys.
- Coordinate routines, handling storage and use of linear transformations and 2-D spherical projections.
- Descriptor routines, getting generic descriptor properties and providing a uniform way of reading and writing column values, header keys, and coordinate values.
- Image routines, supporting operations with image data and image axes.
- Subspace routines, allowing you to store and use information about how the data have been selected.
- Kernel routines, supporting selection of underlying disk formats and use of special features of those formats (e.g. selecting ASCII rather than binary tables in FITS, a detail not modelled at the ASCDM level).

Our initial experience with the interface shows a significant reduction in the number of lines of code needed for typical operations compared to our earlier use of lower level data I/O libraries.

6. Special FITS keywords used in the ASCDM

The ASCDM's generalization and abstraction of the information written to the data files requires some extra FITS keywords. In all cases, if the new keywords are ignored, the file is less fully self-describing but still meets the existing HEASARC standards and will be correctly recognized by FTOOLS and other HEASARC-compliant programs.

The ASC has contacted other groups in the XMM and Integral projects and at GSFC who together have been investigating the possibility of adding further abstraction to FITS files. A 'common data model' email discussion was held in 1997 to propose some new keyword conventions. While no overall agreement was reached, the choice of the header and group descriptor ASCDM keywords described below reflects the main thrust of that discussion.

6.1 Header keys

The ASCDM has support for extra header key information. The traditional FITS header key has simply a short (8 characters or less) name, a value, and a comment. Instead, the most general ASCDM scalar header key allows a long name, a datatype and a unit. We support this structure in FITS using multiple FITS header keywords, but retain a single keyword where this is possible.

- DTYPEn is used to store the long name of the ASCDM key, which is not restricted to 8 characters.
- DUNITn is optionally used to store the unit of the key (e.g.cm, keV, W).
- The unit is also written in square parentheses at the beginning of the comment of the value keyword, thus

  ENERGY = 6.4 / [keV] Line energy

- DVALn is used to store the value of the key if the name is more than 8 characters. If the name is less than 8 characters, the traditional 'name=value' syntax is used instead, for simplicity and back compatibility. Here is an example:

```
COMMENT Full style
DTYPE1 = 'Acceleration'    / DM key name
DVAL1  =          9.81     / [m/s^2] Value for DTYPE1
DFORM1 = 'E       '        / Single precision implied
DUNIT1 = 'm s^{-2} '       / Unit for DTYPE1

COMMENT Example with short name
DTYPE2 = 'Speed'           / DM key name
SPEED  = 299792458.0       / [m/s]  Value for DTYPE2
DUNIT2 = 'm/s     '        / Unit for DTYPE2

COMMENT Short style, no DTYPE used
POSN   =       4.8         / [m] Position
```

In practice we only use the DTYPE keys for keyword names which are longer than 8 characters, and recommend these be used sparingly.

6.2 Group descriptors

Another abstraction of the ASCDM is to treat 2-dimensional positional columns (e.g. DETX, DETY) as a single 'group descriptor'. Similarly, a column of values and a column of corresponding uncertainties can be grouped together as a different kind of group descriptor. Also, World Coordinate System (WCS) names can be grouped together, e.g. EQPOS = (RA,DEC). The user interface will allow filtering on the group quantities, with the syntax 'EQPOS = CIRCLE(14:03:20 -00:23:11 5')'.

We use the keywords MTYPEn, MFORMn and METYPn to describe these groupings.

- MTYPEn gives the name of the grouped descriptor.
- MFORMn gives the component names of the group: the names of the individual columns, axes or coordinates, separated by commas.
- METYPn indicates the interpretation of the group. Allowed values include 'V' (value), 'R' (range), 'VU' (value with uncertainty). The default is 'V', indicating that the group is to be interpreted as a point in n-dimensional space, each column of the group representing a different axis. 'R' indicates two columns which represent a bin lower and upper end. We also plan to support METYP values which indicate a bin defined by its lower or central value, generalizing the HEASARC's TIMEPIXR keyword, and values which indicate upper limit flags.

Two examples of the use of these columns are given below. In the first example, we define a grouped quantity SKY and a grouped pair of coordinates EQPOS. The data model makes explicit the fact that a tangent projection is only meaningful on a pair of columns. Its internal coordinate transform object links the group SKY to the group EQPOS, rather than linking the individual X,Y, and RA,DEC quantities which are not independent.

```
TTYPE6 = 'X       '                / Column name
TTYPE7 = 'Y       '                / Column name
MTYPE1 = 'SKY     '                / Grouped descriptor name
MFORM1 = 'X,Y     '                / Group definition
TCTYP4  ='RA---TAN'                / Transformation and name
TCTYP5  ='DEC--TAN'                / Transformation and name
MTYPE2  ='EQPOS   '                / Group name: equatorial position
MFORM2  ='RA,DEC'                  / Group of coordinates
TCDLT4  =   -1.361E-04             / Deg per pixel (RA decreasing)
TCDLT5  =    1.361E-04             / Deg per pixel
TCRPX4  =       4096.0             / Center of sky field
TCRPX5  =       4096.0             / Center of sky field
TCRVL4  =    322.34824             / RA of field center (deg)
TCRVL5  =    -21.00023             / Dec of field center (deg)
```

In the second example, we encode in a good times interval extension the fact that START and STOP are really a set of TIME bins, mapping to the column called TIME in any associated EVENTS file.

```
TTYPE1 = 'START   '                / Column name
TTYPE2 = 'STOP    '                / Column name
MTYPE1 = 'TIME    '                / Group name
MFORM1 = 'START,STOP'              / Group definition
METYP1 = 'R       '                / START and STOP form a TIME bin
```

6.3 Data subspace keywords

The simplest data subspace filter is the logical AND of the restrictions on several columns. More complex logical expressions, like 'this data was taken when (X1=A AND X2=B) OR (X1=C AND X2=D)' may be rare in practice, but we have such a case with AXAF's ACIS CCD imager, where we want to make a single event list for the multiple chips, but due to dropouts and saturation effects the GTIs may be different for each chip. The ability to OR such filters is also useful for combining data from different missions. The spirit of the choices HEASARC has made for multiple coordinate system keywords is carried over into the ASC implementation of multiple filters.

We record the data subspace in a block with the keywords DSTYPn, DSVALn, and DSREFn.

- DSTYPn indicates the nth data subspace quantity. It may be the same as a column name, but does not have to be.
- DSVALn is a string containing ranges of allowed values. For single columns, the ranges are expressed as min:max pairs separated by commas. The format for storing 2-dimensional regions has not yet been finalized. The special value DSVALn = 'TABLE' is also recognized, indicating the values are stored in a separate table.
- DSREFn is used to indicate the relative path to the external table when DSVALn = 'TABLE'. At the moment we only use tables within the same FITS file, denoted by the EXTNAME value with a leading colon, e.g. DSVAL1= ':GTI '.

This first example is a simple data subspace with only AND operations between the different columns (in this case, PHA and TIME).

```
EXTNAME= 'EVENTS'          / This stuff is in the EVENTS table
DSTYP1 = 'PHA  '           / Quantity restricted
DSVAL1 = '20:30,40:140'    / Allowed ranges
DSTYP2 = 'TIME '           / Quantity restricted
DSVAL2 = 'TABLE'           / Values are in a separate table
DSREF2 = ':GTI'            / whose EXTNAME is 'GTI'
```

The second example supports a more complicated case with (TIME=GTI0 AND CCD_ID=0) OR (TIME=GTI1 AND CCD_ID=1).

```
DSTYP1 = 'CCD_ID '         / Quantity restricted
DSVAL1 = '0'               / Allowed ranges
DSTYP2 = 'TIME '           / Quantity restricted
DSVAL2 = 'TABLE'           / Values are in a separate table
DSREF2 = ':GTI0'           / whose EXTNAME is 'GTI0'
2DSVAL1= '1'               / CCD no. 1
2DSVAL2= 'TABLE'           /
2DSREF2= ':GTI1'           / Values are in GTI1
```

7. Summary

The ASCDM is now a well defined set of software and conventions. The first ASC internal use of the ASCDM library was in late 1997, and our development group has been phasing in use of the library since then. The advanced functionality of coordinate systems, grouped descriptors, and region filtering is being phased in over a series of upgrade releases prior to the AXAF launch. We hope that other missions will consider using either the ASCDM or its FITS conventions for their needs.

We thank Arnold Rots, Clive Page, Julian Osborne, Bill Pence, Morton Barfoed, Don Jennings and Koji Mukai for contributing to the discussion on data models and common data model keywords.

The full documentation for the ASCDM will be available with the forthcoming release of the ASC software, it will also appear on the official ASC web site at *http://asc.harvard.edu*