SCHOOL OF PHYSICS AND ASTRONOMY
UNIVERSITY OF LEICESTER
PROG/002
Asterix Data Analysis
Programmer Note 002

TJ Ponman
25 July 1991

ASTERIX Standard Structures and Conventions

Contents

1 Introduction

ASTERIX files are based on the Starlink standard structures defined in SGP38; however this defines only certain basic constructions, and is free of astronomical content and instrumental detail. It has therefore been necessary to extend it considerably by defining new structures. The purpose of this document is to define these standard ASTERIX structures so that they can be consistently utilised by programmers, and also to specify certain conventions relating to the interpretation and manipulation of data.

There are two main types of datasets within ASTERIX: binned datasets and event datasets. The former are used to represent binned data of any dimensionality (up to 7) and are based on the Starlink NDF construction (see SGP38) in order that they should be accessible by standard Starlink software. Event datasets represent the various properties of a set of individual (unbinned) photon events; this is done by a set of parallel LIST structures, one for each property represented. Each LIST is in fact an NDF in its own right (though it lacks any axis information) and so can be processed by many analysis applications, which can be useful.

2 BINNED DATASETS

Here we list the structure of a general N-dimensional binned dataset, and then consider the various substructures in turn.

```
DATASET              <#D>                See comment below on
                                         TYPE
  TITLE                <_CHAR*80>
  DATA_ARRAY(m,n,...) <_REAL>            Could be any numeric
                                         type
  LABEL                <_CHAR*80>
  UNITS                <_CHAR*80>
  VARIANCE(m,n,...)    <_REAL>
  QUALITY              <QUALITY>
     QUALITY(m,n,...)     <_UBYTE>
     BADBITS              <_UBYTE>
  AXIS(N)
     DATA_ARRAY           <p_array>
     LABEL                <_CHAR*80>
     UNITS                <_CHAR*80>
     NORMALISED           <_LOGICAL>
     WIDTH                <s_array>
  MORE                 <EXTENSION>
     ASTERIX              <EXTENSION>
        HEADER               <EXTENSION>
        INSTRUMENT           <EXTENSION>
           SORT                 <EXTENSION>
        PROCESSING           <EXTENSION>
           BGND_SUBTRACTED      <_LOGICAL>
           CORRECTED            <EXTENSION>
```

```
            DEAD_TIME          <_LOGICAL>
            VIGNETTING         <_LOGICAL>
            BARYCENTRIC        <_LOGICAL>
      LIVE_TIME          <EXTENSION>   Only for data binned
                                       up from event data
      PSF                <EXTENSION>   Only for image data
      ENERGY_RESP        <EXTENSION>   Only for spectral data
HISTORY            <HISTORY>
```

Note that the only component which is mandatory for a binned dataset is the DATA_ARRAY, though some particular applications will be unable to function sensibly without further information. In general software should default as many of the other components as possible, where they are not present.

A package of subroutines is available to allow easy access to components in standard binned datasets. See ASTERIX document PROG_003 for details.

2.1 TYPEs Of Binned Dataset

Dataset TYPE could be '1D', '2D' etc in the general case, but a range of standard TYPEs is also defined, as listed below. Note that the order of the axes in multidemensional cases is not generally fixed, though some software may insist on a particular order, as discussed in section 6.

```
SPECTRUM             - count rate vs. channel, energy or
                       frequency
IMAGE                - count rate vs. x,y position
TIME_SERIES          - count rate vs. time
SPECTRAL_SET         - set of concurrent spectra from
                       multiple detector instrument (2D)
                       Spectral axis must be dimension 1
SPECTRAL_SERIES      - time series of spectra (2D array)
IMAGE_SERIES         - time series of images (3D array)
SPECTRAL_IMAGE       - 3D dataset representing a set of
                       spatially resolved spectra
SPEC_IM_SERIES       - time series of spectral images
HARD_SERIES          - hardness ratio vs. time
FOLDED_SERIES        - 1D dataset folded at some period
DISTRIBUTION         - frequency distribution
PROJECTION           - projection of some object of higher
                       dimensionality
SCATTERGRAM          - scattergram relating two sets of
                       quantities
POLAR                - polar profile, surface brightness vs.
                       radius, azimuth
POWER_SPECTRUM       - power vs. frequency, may also contain
                       a DATA_PHASE array
AUTOCORRELATION      - autocorrelation function vs. lag
CROSS_CORR           - cross-correlation function vs. lag
```

```
CROSS_SPECTRUM        - cross-spectrum, includes coherency
                        and phase array structures
DYN_SPECTRUM          - dynamical power spectrum, 2D dataset
                        giving a time sequence of power
                        spectra (calculated from successive
                        data segments)
GRAFIX                - graphics display object
```

A brief description of the nature of a dataset can be included in the TITLE component. This is available for user information, and will also be displayed by graphics sofware, unless explicitly overridden by a graphics legend.

2.2 The DATA_ARRAY And Associated Components

In principle we could use any of the Starlink ¡ARRAY¿ types for DATA_ARRAY; in practice software will only be expected to support primitive arrays, at least for the time being.

2.2.1 Units

For consistency the following standard abbreviations and syntax rules, mostly adapted from SGP38, should be used. The principal units used in ASTERIX have been starred. Note that case can be important; in general lower case should be used and plurals avoided (e.g. 'count', not 'Counts').

Unit	Abbreviation
*count	count
*photon	photon
*second	s
minute	min
day	day
year	yr
*centimetre	cm
metre	m
kilometre	km
*arcsecond	arcsec
*arcminute	arcmin
*degree	deg
radian	rad
steradian	sr
gram	g
kilogram	kg
electron volt	eV
*kiloelectron volt	keV
*erg	erg
joule	J

```
watt                   W
herz                   Hz
angstrom               A
magnitude              mag
kelvin                 K
```

Note that 'channel', 'bin' and 'pixel' are understood (e.g. count/s implies counts per second per channel, for a spectrum) and need not be entered as explicit units.

In compound units, brackets should be used where necessary to avoid ambiguity. E.g. count/(arcmin**2*s), photon/(cm**2*s*keV).

2.2.2 Quality

Data quality is flagged using the QUALITY system described in SGP38. Each data value has an associated BYTE quality value stored in the QUALITY array. This is masked by preforming a logical AND with the BADBITS mask. If the resulting byte is non-zero (i.e. any 1 in the QUALITY byte matches a 1 in the mask) then quality is bad for that point.

The BADBITS mask is usually set to 11111111 (so that any non-zero QUALITY value will be taken as bad), except in the case of 'patched' data (see below).

The following conventions for QUALITY are used within ASTERIX:

```
a) QUALITY=00000000  => data good
b) QUALITY=00000001  => data missing
c) QUALITY=00000100  => arithmetic error occurred (e.g. zero
                        divide or log(-ve))
d) QUALITY=01000000  => patched data value (e.g. replaced by
                        interpolated value)
e) QUALITY=1*******  => IGNOREd data
   The leftmost quality bit is reserved for temporary flagging
   of data (e.g. by the IGNORE command) so as to allow selected
   points to be omitted from processing. This is done by toggling
   the bit, leaving any other quality code unaffected, so that
   the previous quality status can be recovered with RESTORE.
f) Other bits (2,4,5 and 6) are just generalised 'bad' at
   present.
```

Note that in the case of patched data (d), the BADBITS mask will be set by the patching software to 10111111, hence patched data are flagged, but will still be regarded as 'good' by subsequent software; they can be excluded by resetting BADBITS to 11111111 using the MASK application.

ASTERIX programmers can access the above standard definitions using the QUAL_PAR include file.

2.2.3 Magic Values

The system of "magic values" (in which certain special data values represent "bad" values) is less flexible than the QUALITY system, and is not supported by ASTERIX applications. However it does have the merit of reducing the size of data files (since no ancillary QUALITY array need be stored), and is employed in several other Starlink packages. It is possible to convert between NDFs employing the two systems using the ASTERIX applications 'MAGIC' and 'SETQUAL MAGIC', thus:

```
                 MAGIC
Quality -------------------> Magic

             SETQUAL MAGIC
Magic ---------------------> Quality
```

Further details are given in ASTHELP.

2.3 AXIS Information

AXIS values correspond to bin centres, and will generally be stored either as spaced arrays (for regularly spaced values) or primitive arrays. REGULARLY SPACED VALUES SHOULD BE STORED AS SPACED ARRAYS SO THAT SOFTWARE WILL BE ABLE TO RECOGNISE THEM AS SUCH. Software finding a primitive array of axis values should assume that it could be irregular and act accordingly (if it can't handle irregular spacing it must warn the user and either proceed assuming regular spacing, or exit).

NORMALISED=.TRUE. indicates that the values in the DATA_ARRAY have been normalised with respect to the axis values of the axis in question (e.g. count/s or count/arcmin**2). Note that if NORMALISED is not present it is assumed that the data are NOT normalised.

WIDTH is the bin width, which may be scalar (all bins of equal width) or vector (one value for each value in the axis array). The bins are assumed to be centred on the axis values. If we find a requirement for offset bins, or for bin durations which are not equal to bin widths, then we will have to make use of the proposed AXIS extensions, to represent the extra information.

2.4 Asymmetrical Error Bars

The Starlink NDF standard does not at present cater for asymmetrical errors on either data or axis values. In the absence of such a standard, the ERROR structure, and UWIDTH and LWIDTH components within the AXIS structure have defined for use within ASTERIX. At this stage these are intended solely to allow display of data with asymmetrical error bars, and are supported only by the ASTERIX graphics software (for 1D data only) and by IMPORT, which can be used to generate an NDF with asymmetrical errors from a text file. The additional components will not be recognised by ASTERIX analysis software, and since they fall outside the SGP38 standard, they will not be propagated by Starlink applications (including ASTERIX ones). However, since they are entirely additional components (not changes to existing ones) they will not interfere with the operation of other programs.

An example of a 1D dataset containing asymmetrical errors for both data and axis dimensions is:

```
DATASET
  DATA_ARRAY(m)
  ERROR
    UPPER(m)
    LOWER(m)
  AXIS(1)
    DATA_ARRAY(m)
    LWIDTH(m)
    UWIDTH(m)
  LABEL
  UNITS
  etc.
```

Note that,

- The graphics software will use ERROR in preference to VARIANCE if both are present. (Other software will see only the latter.)
- Similarly LWIDTH/UWIDTH will be used in preference to the standard WIDTH component by graphics only.
- Asymmetrical errors may be specified for either AXIS, or DATA, or both.

2.5 The MORE And ASTERIX Structures

All miscellaneous information present in an NDF must be put into the MORE structure to ensure that it is propagated by Starlink software. Strictly, items in MORE should be 'registered' with Starlink.

ASTERIX-specific components are located within the ASTERIX structure to separate them from any other components (possibly with clashing names) which might be defined by other Starlink packages.

2.6 The ASTERIX HEADER

HEADER contains standard header items describing an observation. These are drawn from the following list as relevant for a particular dataset.

```
TARGET         <_CHAR*40> - name of astronomical source
OBSERVER       <_CHAR*40> - name of owner
OBSERVATORY    <_CHAR*40> - e.g. 'EXOSAT'
INSTRUMENT     <_CHAR*40> - particular instrument (e.g.
                            'ME')
```

```
AXIS_RA        <_DOUBLE>  - RA in decimal degrees of
                            pointing axis (or more general
                            spatial reference point)
AXIS_DEC       <_DOUBLE>  - declination in decimal degrees
                            of spatial reference point
EQUINOX        <_INTEGER> - equinox (year) for RA and Dec
POSITION_ANGLE <_DOUBLE>  - Angle (degrees) from North
                            through East to +ve y axis
                            (i.e. standard astronomical PA
                            of +y axis)
FIELD_RA       <_DOUBLE>  - RA of field centre
FIELD_DEC      <_DOUBLE>  - declination of field centre
BASE_DATE      <_CHAR*11> - day of observation reference
                            time: dd-mmm-yyyy (This
                            contains the same information
                            as BASE_MJD, and is provided
                            for user convenience)
BASE_MJD       <_INTEGER> - MJD of observation reference
                            time (whole part only)
BASE_UTC       <_DOUBLE>  - UTC in seconds of the reference
                            time, measured from 0:0 on the
                            day of BASE_DATE/BASE_MJD
BASE_TAI       <_DOUBLE>  - Atomic time (TAI) since 0:0 UTC
                            on Jan 1st 1972, in units of
                            days. This is the standard
                            continuous reference time
                            system used by ASTERIX.
OBS_LENGTH     <_REAL>    - total observing time in
                            seconds, making no allowance
                            for dead time losses etc
EXPOSURE_TIME  <_REAL>    - actual exposure time in seconds
                            after allowing for any data
                            losses
```

The contents of HEADER and of the PROCESSING structure described below can be conveniently inspected using the HEADER application.

2.6.1 The Time Reference Frame

All time values in a dataset are expressed relative to a 'reference time'. This will usually correspond to the observation start, but need not do so. The familiar system of MJD and UTC (supplemented by the redundant BASE_DATE for ready comprehension) is used, but it suffers from the serious disadvantage that UTC contains a discontinuity at each leap second.

Hence a continuous atomic time, BASE_TAI, is also defined. This starts from 0:0 on Jan 1st 1972 (since this is the point from which leap seconds are included in the Starlink routine SLA_DTA). BASE_TAI is expressed in days, so that

```
BASE_TAI = (no. of seconds elapsed from 0:0 Jan 1st 1972
            to the observation reference time) / (24*60*60)
```

Hence the TAI (in seconds) corresponding to a timetag T (in seconds) will be (24*60*60*BASE_TAI+T), whereas the corresponding UTC will NOT necessarily be (BASE_UTC+T), because of the possibility that a leap second may have occurred in the interval of T seconds since the reference time.

Software which enters or revises the base time (mostly the instrument interface software) must take care that leap seconds are allowed for in arriving at the BASE_UTC value. Interconversion between MJD/UTC and TAI can be achieved using the ASTERIX subroutine TIM_MJDUTC2TAI. This employs the Starlink SLALIB routine (see SUN67) SLA_DAT, which contains details of all leap seconds announced, and is regularly updated. The relationship between the base values is:

```
  BASE_TAI=OBS_MJD+((SLA_DAT(OBS_MJD)-10.0D0)/86400.0D0)
                                 -41317.0D0
where
        OBS_MJD=DBLE(BASE_MJD)+BASE_UTC/86400.0D0
```

2.6.2 The Spatial Reference Frame

In a rather similar way, spatial position is given as x,y offsets relative to a reference frame defined by AXIS_RA, AXIS_DEC and POSITION_ANGLE. These might define the frame of the instrument during the observation, but in general (e.g. for a scanning instrument) they just specify some convenient reference frame.

Note that FIELD_RA and _DEC do NOT define the reference frame for the position information; they just contain the coordinates of the field centre.

2.7 Instrument-specific Information

The INSTRUMENT extension contains instrument-specific information, which is defined separately for each instrument supported by ASTERIX. For details see the relevant instrument interface documents (e.g. USER_002 for EXOSAT).

One important component commonly present in the INSTRUMENT structure is SORT. This contains information about the ranges in the data dimensions (time, energy, x, y) over which data have been accumulated to form the binned dataset. These ranges are initially established in the event sorting, or are defined by the instrument itself (if it generates ready binned data). Some of the data dimensions may have disappeared in the binning process, so that preservation of the range information may be important for subsequent software, or for user information. The structure of SORT varies in detail from one instrument to another.

2.8 The PROCESSING Structure

This component flags basic operations which may have been performed on the data. This currently includes background subtraction, correction for instrument vignetting and dead-time, and barycentric correction. See section 4 for further discussion.

2.9 Instrument Response Information

2.9.1 Point Spread Function

The PSF component contains information allowing the point spread function of an imaging instrument to be recovered by software. Due to the fact that the PSF may be best represented in quite different forms for different instruments and may be dependent on energy and epoch, the name and arguments of a subroutine to calculate values from a point spread function are stored, rather than storing the values themselves. The structure is:

```
PSF                    <EXTENSION>
   ROUTINE_NAME           <_CHAR*20>
   LIBRARY_NAME           <_CHAR*20>
```

Software then recovers a value from the PSF by calling the subroutine whose name is given in ROUTINE_NAME, which is located in the shareable library with logical name given in LIBRARY_NAME. A suite of PSF_* subroutines to perform these operations is described in document PROG_011.

2.9.2 Energy Response

ENERGY_RESP represents the instrument response matrix for a spectral instrument. Its structure is as follows:

```
ENERGY_RESP               <EXTENSION>
   VERSION                   <CHAR*80>
   ENERGY                    <LIST>
      DATA_ARRAY(N)             <_UBYTE, _UWORD or _INTEGER>
      ENERGY_SPEC(NEN)          <_REAL>
      ENERGY_BOUNDS(NEN+1       <_REAL>
   CHANNEL                   <LIST>
      DATA_ARRAY(N)             <_UBYTE, _UWORD or _INTEGER>
      CHANNEL_SPEC(NCH)         <_REAL>
      CHANNEL_BOUNDS(NCH+1)     <_REAL>
   RESPONSE                  <LIST>
      DATA_ARRAY(N)             <_REAL>
```

Basically, this is a set of three lists: ENERGY, CHANNEL and the RESPONSE value which maps between the two, together with a VERSION string which defines the epoch or version number of the response (important in case it changes). The ENERGY and CHANNEL lists are stored, for compactness, as indices, with translation into actual values specified by the ENERGY_SPEC and CHANNEL_SPEC arrays. (i.e. ENERGY index 1 corresponds to energy ENERGY_SPEC(1)). ENERGY_BOUNDS specifies the boundaries of the energy space channels, and CHANNEL_BOUNDS the energy values corresponding to the instrument channel boundaries.

Response values map spectral fluxes in photon/(cm**2*s) per energy bin onto count/s per channel - they therefore correspond to effective areas as a function of energy for each channel.

In the case of a SPECTRAL_SET, which contains a set of parallel spectra, the ENERGY_RESP is an array of structures (one for each spectrum in the set) each component having the structure shown above.

2.10 HISTORY

This is the standard HISTORY system which Starlink has adopted from ASTERIX; it is described in SGP38. All applications which create or modify a standard file should insert a HISTORY entry detailing the program version, the date and time of execution and the important program parameters involved. A package of subroutines is available to assist this - see PROG_006. The history of a file is easily inspected by invoking the ASTERIX application 'HISTORY'.

2.11 An Example - An IMAGE Dataset

A 'live' example of an ASTERIX binned dataset. Only the first component of structure arrays have been displayed to save space.

```
      Component            Type              Contents
------------------------------------------------------------
DATASET              <IMAGE>
   DATA_ARRAY(512,512)   <_REAL>        0,0.4,1.5,0,0.9,...
   TITLE                 <_CHAR*80>     'EXOSAT LE'
   LABEL                 <_CHAR*80>     'Intensity'
   UNITS                 <_CHAR*80>     'count/s'
   AXIS(2)               <AXIS>
      Contents of AXIS(1)
      LABEL                 <_CHAR*80>      'X_CORR'
      UNITS                 <_CHAR*80>      'deg'
      DATA_ARRAY            <ARRAY>
         VARIANT               <_CHAR*6>      'SPACED'
         BASE                  <_REAL>        -1.634579
         SCALE                 <_REAL>        6.3975E-03
         DIMENSION             <_INTEGER>     512
   MORE                  <EXTENSION>
      ASTERIX               <EXTENSION>
         LIVE_TIME             <EXTENSION>
```

```
         ON(2845)             <_REAL>       -2.46,2.4...
         OFF(2845)            <_REAL>       2.40,9.15...
         DURATION(2845)       <_REAL>       4.80,6.75...
      HEADER               <EXTENSION>
         TARGET               <_CHAR*20>    '1820-303'
         OBSERVER             <_CHAR*20>    'Ponman...
         OBSERVATORY          <_CHAR*6>     'Exosat'
         INSTRUMENT           <_CHAR*2>     'L1'
         AXIS_RA              <_DOUBLE>      275.1606
         AXIS_DEC             <_DOUBLE>      -30.3572
         FIELD_RA             <_DOUBLE>      275.1608
         FIELD_DEC            <_DOUBLE>      -30.3572
         EQUINOX              <_INTEGER>     1950
         POSITION_ANGLE       <_DOUBLE>      -89.2496
         BASE_DATE            <_CHAR*11>     '16-Apr...
         BASE_MJD             <_INTEGER>     46171
         BASE_UTC             <_DOUBLE>      43657.8...
         BASE_TAI             <_DOUBLE>      4854.50...
         OBS_LENGTH           <_REAL>        39101.78
         EXPOSURE_TIME        <_REAL>        16590.63
      INSTRUMENT           <EXTENSION>
         PIXEL_SIZE           <_REAL>        1.1111E-03
         FILTER               <_CHAR*50>     'Al/Par...
         DETECTOR             <_CHAR*12>     'CMA'
         OBC_MODE             <_CHAR*7>      'LDIR2'
         TAPE_VOLID           <_CHAR*6>      '802385'
      PSF                  <EXTENSION>
         LIBRARY_NAME         <_CHAR*25>     'PSFLIB'
         ROUTINE_NAME         <_CHAR*25>     'PSF_EXOLE'
HISTORY                 <HISTORY>
   CREATED                 <_CHAR*18>     '26-JUN-89 14...
   EXTEND_SIZE             <_INTEGER>     10
   CURRENT_RECORD          <_INTEGER>     2
   RECORDS(10)             <HIST_REC>
      Contents of RECORDS(1)
      DATE                    <_CHAR*30>    '26-JUN-89 ...
      COMMAND                 <_CHAR*30>    'Exolesort V...
```

3 EVENT DATASETS

Many of the ancillary structures in event datasets (e.g. the MORE box and its contents) are the same as for binned datasets, and are described above. However the event data itself is stored in a set of parallel LIST structures: one for each event property represented.

Structure of an event dataset:

```
DATASET             <EVENT_DATA>
   TITLE
```

```
Set of lists, e.g.
X_CORR              <LIST>
Y_CORR              <LIST>
ENERGY              <LIST>
TIMETAG             <LIST>
QUALITY             <LIST>
MORE                <EXTENSION>
   ASTERIX             <EXTENSION>
      HEADER              <EXTENSION>
      INSTRUMENT          <EXTENSION>
      LIVE_TIME           <EXTENSION>
      SPATIAL             <EXTENSION>
HISTORY             <HISTORY>
```

3.1 LISTs

The LISTs contain a set of properties for each event (or for each primitive instrument bin, see below). Their structure looks like:

```
X_CORR              <LIST>
   DATA_ARRAY(N)       <_REAL>
   QUANTUM             <_REAL>    Scalar or vector
   UNITS               <_CHAR*80>
   FIELD_MIN           <_REAL>
   FIELD_MAX           <_REAL>
   DECREASING          <_LOGICAL>
```

Of these components, only DATA_ARRAY is mandatory, though FIELD_MIN and FIELD_MAX are required by binning software. Data type does not have to be _REAL, but it is wise to make the types of the numerical components consistent. Note that each LIST is in fact an NDF (containing only DATA_ARRAY, and possibly UNITS), hence a LIST can be used as input to many analysis applications, which will default the missing components such as axis information.

The contents of the various components are:

```
DATA_ARRAY - the actual list values.
QUANTUM    - width of bins used to locate the data array values
             (i.e. quantum of measurement). Can either be the same
             for all events (scalar value) or given separately for
             each event (vector of values). Where QUANTUM is not
             specified it will be assumed by binning software to
             be 1 for integer data types, and 0 (i.e.
             infinite instrumental precision) for _REAL data.
UNITS      - char*80 string giving data units.
FIELD_MIN  - minimum and maximum values of field
FIELD_MAX    over which the list data have been collected. (Note
             that this is not the same as the observed data range).
```

```
             Binning software requires these to be present.
DECREASING - This is a flag which tells binning software to bin this
             axis in decreasing order (e.g. azimuth angle is usually
             treated in this way).
```

Note that FIELD_MIN and _MAX values correspond to the CENTRE values of the extreme primitive bins, not to their boundaries. Unlike ranges entered by users (see section 6) both _MIN and _MAX values are INCLUSIVE.

For some LISTs (e.g. TIMETAG, X_CORR, Y_CORR), the DATA_ARRAY values MUST be expressed as offsets from information in the HEADER. In this case the FIELD_MAX and _MIN values should also be expressed as offsets.

For example, consider a TIMETAG LIST. The values in DATA_ARRAY are offsets from BASE_TAI (in the HEADER). Hence if the observation starts at the time given by BASE_TAI and lasts for a total of OBS_LENGTH seconds, with a measurement quantum of size QUANTUM, then (remembering that FIELD_MIN and _MAX refer to the CENTRES of the extreme bins)

```
FIELD_MIN = QUANTUM/2
FIELD_MAX = OBS_LENGTH - QUANTUM/2
```

3.2 Binned Instrument Data

Some instruments (such as the EXOSAT ME) deliver binned data rather than individual events. Such data cannot be appropriately expressed in LIST form, and should be dumped directly into an NDF by the instrument interface software.

3.3 Standard Names For LISTs

These are the same as those used in ASTERIX. New LISTs may be defined for a particular instrument as required and software should be sufficiently flexible to cater for this (e.g. if it fails to find a standard list the user should be asked to nominate one).

```
X_RAW           - x offset from instrument axis in pixels.
                  LIST type usually _WORD.
Y_RAW           - y offset from instrument axis in pixels.
                  LIST type usually _WORD.
X_CORR, Y_CORR  - x,y positions after instrumental
                  corrections.
PULSE_HEIGHT_CH - instrument pulse height channel number.
                  LIST type usually _UBYTE, _UWORD or
                  _INTEGER, depending upon number of
                  channels.
PULSE_HEIGHT    - this may be used for corrected p.h. values.
ENERGY          - energy in absolute units (e.g. keV).
RAW_TIMETAG     - collection time (usually referenced to
```

```
                  observation start) in seconds. LIST type
                  usually _REAL.
TIMETAG         - used in place of RAW_TIMETAG if any
                  corrections have been applied.
DETECTOR_ID     - detector identification. LIST type usually
                  _UBYTE.
QUALITY         - data quality list. Applies where quality of
                  individual events is supplied by the instrument.
                  Note that this is a LIST, containing its values
                  in a DATA_ARRAY, its structure therefore differs
                  from the QUALITY component in binned datasets,
                  but the quality values should conform to the
                  same convention, as defined in section 2.2.2.
                  LIST type _UBYTE.
```

3.4 Spatial Indexing

The SPATIAL component may contain an optional spatial index to allow faster extraction of events in a given spatial region. This facility has not yet been implemented, and its structure is still under discussion. Currently, software should assume that events are not ordered in any particular way.

4 LIVE_TIME AND INSTRUMENTAL CORRECTIONS

Event data will not have been corrected for instrumental effect like dead time and vignetting. This is done after the binning stage by an instrument-specific correction program (applications running on event data will have to make their own arrangements, using the same correction subroutines). However it will frequently be the case that the instrument interface has done some of the work required for correction (notably extracting the relevant instrument on-times) in the course of assembling the event dataset. Some of this information can be stored in the dataset to allow the subsequent correction stage to be performed more quickly. This is the function of the LIVE_TIME structure:

```
LIVE_TIME      <EXTENSION>
   ON(N)          <_REAL>
   OFF(N)         <_REAL>
   DURATION(N)    <_REAL>                Optional
```

This would be attached to the event dataset when it is created. It is basically just a list of on and off times (leaving aside the last component for the moment). For spatially resolved event data these times refer to live time for the WHOLE FIELD. Live time for a spatial subset may in general be some subset of this overall live time (depending upon telescope motion). The live time values are referred to the same reference time (specified in the HEADER) as the data timetags.

An instrument-specific correction program takes any binned dataset (e.g. image or time series) and uses the LIVE_TIME information, together with HK and aspect data as required (which will generally be in files referenced in the INSTRUMENT extension) to normalise to count/s, correcting for dead-time and vignetting. (It then sets the logical flags in the PROCESSING structure to indicate what has been done.) In the case of an imaging instrument, the corrections may vary with position. If the spatial information has already been removed by binning into some other domain (e.g. a time series has been formed) then the correction application should take the spatial position from the FIELD_RA and FIELD_DEC components in HEADER, and will perform the corrections as if dealing with a single spatial point.

The overall processing scheme for event data looks like this:

```
Instrument interface       Outputs event lists (uncorrected)
('Event Processing')       with LIVE_TIME for whole field
        |
        |
  EVENT DATASET
        |
        |
Event selection            Select a subset of the events based on
        |                  spatial, spectral, time etc criteria
        |
Smaller EVENT DATASET      (e.g. small region centred on source)
        |
        |
Binning (EVENTBIN)         Bins data without normalisation, enter
        |                  FIELD_RA/DEC in HEADER if necessary
        |
  BINNED DATASET
        |
        |
Correction program         Uses LIVE_TIME, field centre, HK and
(instrument-specific)      aspect files to normalise data,
        |                  correcting for dead time, vignetting etc.
        |
Normalised, corrected
 BINNED DATASET
```

Software operating on event datasets (notably EVSUBSET and EVMERGE) must maintain the components required for the subsequent correction process - i.e. LIVE_TIME must be updated if the time ranges are altered and FIELD_RA/DEC must be updated if the spatial field is changed.

Binning software simply bins the data up and propagates the LIVE_TIME, it does not attempt to normalise the data in any way.

Note that software operating on binned data will NOT be expected to altered and FIELD_RA/DEC must be updated if the spatial field is maintain LIVE_TIME or FIELD_RA/DEC. This means that if normalisation is required, it should be performed on the freshly binned data. Manipulation (e.g. subsetting) of the data after binning but before normalisation could lead to incorrect normalisation being applied. In general it would be best if binned data software which affects these components were to delete them, to avoid confusion.

Where one has the simpler case of a non-imaging instrument, the dead time correction can be handled by including the DURATION component in LIVE_TIME. The correction program would then normalise the data by dividing by the collection time and then correcting for dead time by multiplying by (OFF-ON)/DURATION (either summed up, for a spectrum, or as a function of time for a time series).

5 CONTROL OF NEW COMPONENTS

New names and types can be invented by users and programmers as the need arises, but where these have general applicability the X-ray applications programmer should be informed, so that this document can be updated if necessary. This will help to avoid a proliferation of local standards. Where the inclusion of new software in ASTERIX involves the introduction of new names/types this should always be reported.

Note that new components should always be located within the MORE structure if it is intended that they should be propagated by software.

6 ASTERIX CONVENTIONS

Below are listed a number of conventions relating to ASTERIX data, whose consistent application throughout the system is important, so that users know what behaviour to expect, and so that incompatibility between the practices of different pieces of software is avoided.

- Location of a bin - a bin will be located by its centre value. Hence, for example when a subset of a binned dataset is selected it is the bin centre (not the boundaries) which will be tested against the specified range to decide whether a given bin qualifies for inclusion. The same applies for event data, where 'primitive bins' (of width specified by QUANTUM) may already be implicitly present. Minimum and maximum values, as recorded in FIELD_MIN/MAX also refer to (primitive) bin centre values, not to boundaries.

- Behaviour on boundaries - in binning and selection processes the ASTERIX convention is that lower bounds are inclusive and upper bounds exclusive. This rule leads to rather 'unnatural' behaviour in some circumstances (e.g. if one specifies a range of axis values 1 to 10 in a case where the values are integral, then one gets only the values 1 2 ... 9), however it has the great merit that one can select out subsets based on contiguous ranges (e.g. 0-10, 10-20, 20-30 etc.) without any risk of the same data being included in more than one subset. If there is a good reason for breaking the rule of the exclusive upper bound in some particular instance then the user should be explicitly warned.

  One result of the 'upper exclusive' rule is that where software offers a default to the user corresponding to the full data range, it must (if the default is accepted) raise its upper bound slightly (AFTER the selection) in order to ensure that the uppermost point is not left out.

- Regular spacing - regular axis values should always be represented by a ¡SPACED_ARRAY¿ so that software can recognise its regular nature.
- Some axes (e.g. RA) are conventionally stored in reversed form. In such cases data ranges (for example, when subsetting) should be given in increasing index (i.e. decreasing value) order, for example 270:210. The first value will be inclusive and the second exclusive.
- Some applications may require a particular order of axes in a binned dataset. In this case the application should warn the user of this fact. AXFLIP and AXSWAP utilities are provided to allow axis order to be easily changed.
- Normalisation - the NORMALISED component in the AXIS structure in principle allows software to handle binned data in both normalised and unnormalised forms. However, it will be less confusing for users if a consistent practice is adopted with regard to normalisation. Normal practice is for binned data to be normalised with respect to time, but not with respect to the spectral dimension. In the spatial domain it is common to encounter both normalised and unnormalised data (e.g. counts per square arcmin and counts per pixel) so software will probably have to cope with both cases, however the 'normal' case will be taken as unnormalised.

Axis	Typically
x	unnormalised
y	unnormalised
time	normalised
energy	unnormalised

- Default units - the following units should be assumed by default by ASTERIX applications:

Dimension	Default unit	Comment
position	degree	relative to AXIS_RA/DEC/PA
time	second	relative to BASE_TAI
energy	keV	

7 REFERENCES

ASTERIX documents

USER_002: EXOSAT Raw Data Handling in ASTERIX
PROG_003: Binned Dataset Access Routines
PROG_006: New HISTORY Routines
PROG_011: The PSF subroutines

Starlink documents

SGP38: Starlink Standard Data Structures
SUN67: SLALIB - a library of subprograms