FITS Keyword Conventions in ASC Data Model files
SDS-7.2

Jonathan McDowell

January 11, 1999

Contents

1 Introduction **3**
1.1 FITS Implementation 3
1.2 Notes on existing FITS special cases 3
1.3 The name of an HDU 3
1.4 Extra information for compound columns 4
1.5 Support for preferred columns or axes 5
1.6 Extra information for header keys 6
1.7 Array keywords 8
1.8 Images 8
1.9 Coordinate Systems 9
1.10 Coordinate systems on image block axes 10
1.11 Coordinate Data Descriptors: columns 11
1.12 Coordinate descriptors for array column axes 12
1.13 Keywords for recording filters 12
1.13.1 Special cases 15

2 Systematic algorithm for creating descriptors **16**
2.1 Existence of a descriptor on read 16
2.2 Descriptor names 16

2.3 Descriptor component names 17
2.4 Element dimensionality 18
2.5 Descriptor units 18
2.6 Descriptor values 18
2.7 Data type 19
2.8 Descriptor desc 19
2.9 Display format 19
2.10 Element type 20
2.11 Array dimensionality 20
2.12 Array sizes 20
2.13 Legal range 21
2.14 Transform type 21
2.15 Transform values 21

3 Proposed future conventions **21**
3.1 Future enhancement for ASC 'element types' 21
3.2 Element types 23
3.3 Extra keys for header keywords 23
3.4 Data Subspace keys 24

1 Introduction

This document introduces FITS header keyword conventions for use with the ASC data model. The guiding principle used is to select defaults so that existing FITS files should be correctly interpreted by the data model. The new keywords are as far as possible chosen to be analogous to existing FITS conventions.

This document supersedes the earlier SDS-7.1 and reflects the implementation current as of January 1999. It is intended for those already familiar with HEA (GSFC) FITS conventions.

1.1 FITS Implementation

1.2 Notes on existing FITS special cases

1. Zero-width columns (e.g. 'TFORM3 = 0I') are forbidden.

2. In header keywords, NaNs should be converted to a keyword with a blank value field:

   FOO = /

   In floating point binary table columns, IEEE NaNs are fine.

3. Use of TSCAL and TZERO is currently deprecated except for the special case of specifying unsigned integer types. Recommend use of TCRVL and TCDLT instead, with the corresponding linear coordinate transform machinery which gives clearer information on the intent.

1.3 The name of an HDU

In our software, each FITS HDU is given a unique name as follows:

- If HDUNAME is present, its value is the HDU name, and we ignore EXTNAME.

- If there is no HDUNAME, but EXTNAME and EXTVER are present, the name is the value of extname concatenated with the value of extver. Example:

  ```
  EXTNAME = 'SPECTRUM'    / Spectral data
  EXTVER  =         3     / Version no; ASC name will be SPECTRUM3
  ```

- If there is no HDUNAME or EXTVER, but EXTNAME is present, the name is the value of EXTNAME. We recommend that EXTNAME values not end with digits, since on copying an HDU to another file we're likely to strip the trailing digits on the assumption they're meant to be an EXTVER.

- If there is no HDUNAME or EXTNAME, we name the HDU to be "HDUn", where n is the HDU number counting the primary array as HDU1. (In earlier releases, we called it HDU0).

1.4 Extra information for compound columns

At the ASC's high level Data Model layer, we define compound columns which may map to multiple FITS columns. This is reflected in the FITS file with extra keywords that tie the columns together. If these keywords are ignored, the columns are just seen as independent in the usual way.

We propose a new set of FITS header keywords to describe the extra structure on top of the raw BINTABLE. By analogy with keywords like TTYPEn, these new keywords are indexed keywords beginning with a common letter M (for Meta-column). The most important keywords are MTYPEn and MFORMn, defined by the Common Data Model (CDM) discussion list. MTYPE4 = 'SKY', MFORM4 ='X,Y', TTYPE13='X', TTYPE14='Y' defines a descriptor SKY composed of columns 13 and 14.

To parse a table, we use the following rules:

- The index subscript on the MTYPEn series of keywords does not impose an ordering. Descriptor order is imposed by the ordering of the TTYPEi keywords of the first element of each descriptor.
- Although the CDM does not require that descriptor (meta-column) components be adjacent TTYPEi columns, we will require this for the time being.
- Starting with TTYPE1, we examine the next TTYPEi which has not already been marked as a component.
- If the TTYPEi value appears as the first item in any MFORMn, we have a new compound column whose name is the value of the corresponding MTYPEn and whose component names are the comma-separated items in MFORMn. We identify the remaining component names with TTYPEi values and mark those TTYPEs as components. The element dimension and element type are inferred from the number of component names and the value, if any, of METYPn (see later discussion).
- If the TTYPEi value does not appear in an MFORMn, we have a new (non-compound) column whose name is the value of TTYPEi, and whose element dimension is 1 and element type (see later) is V.
- Continue until all TTYPEs have been dealt with.

The special keywords are:

- MFORMn (string) is a comma-separated list of names (at least one name; zero is an error) which defines a composite descriptor. Each name should be either the value of one of the TTYPEn keywords (i.e. a FITS column name) or the name of a FITS keyword. MFORMn is a CDM keyword.

- MTYPEn (string) gives the name of the composite descriptor defined by MFORMn. MTYPEn is a CDM keyword.
- METYPn (string) gives the Data Descriptor's element type. Initially supported types will be 'V' (value), 'VU' (value with one uncertainty range), 'R' (range, binned data'). 'REG' (2D region string descriptor). If absent, a default value of 'V' is assumed. METYPEn is an ASCDM keyword. As of Jan 1999, METYPn support has not yet been implemented.
- MDESCn (string) gives the Data quantity description (a comment for the compound column). If absent, the default value is the comment string following the / in the MTYPEn or, if that is absent, the TTYPEj keyword. We have not implemented MDESCn as of Jan 1999.

We note the following existing FITS keywords and their use:

- TFORMj is used to store the Data Descriptor's data type and the number of elements per cell, and also the string length if applicable.
- TDIMj is used to store the Array Specification axes.
- TUNITj is used to store the Data Descriptor's unit.
- TTYPEj, TTYPEj+1,.. are used to store the Data Descriptor Component Names when DCEDIMn is more than 1.
- TDISPj is used to store the Data Descriptor display format.
- TLMINj and TLMAXj are used to store the legal range of values. This is used by us and by HEASARC software for filtering and binning.

1.5 Support for preferred columns or axes

We expect to implement support for preferred axes prior to launch.

- CPREF (string) specifies preferred quantities: the most interesting axes, and the ones you should bin on if no axes are specified. Its format is

  ```
  CPREF =  'DETX,DETY'          / default axes to bin on
  CPREF =  'PHA(DETX,DETY)'     / default axes to bin on, with weighting function
  ```

  The optional weighting function is the name of a column to weight by, which must be a single FITS scalar column. The binning axes can include compound column names, but not array columns.

1.6 Extra information for header keys

On reading a FITS header, all the mandatory FITS keywords and the keywords defining the BINTABLE/IMAGE and overlying ASC TABLE structure are parsed. All remaining keywords are interpreted as block header keys.

We introduce a new set of header keywords analagous to the TTYPEn series, for attributes.

We have implemented two different forms of FITS enhanced keyword support (to store more info about each keyword) - the 'long form' and the 'short form'. In the short form, info is packed into the FITS comment keyword. In the long form, needed for long keyword names, separate keywords are used.

In all the following cases, string keywords with blank or default values should be omitted (i.e. DUNITn should not appear in the file if the unit is blank).

The short form is

```
FOO   =  value  / [unit] {type} desc
```

The unit convention is as per CFITSIO. The new `{type}` convention, to be implemented by us in Jan 1999, specifies an intended data type for a numeric keyword, allowing us to distinguish between float and double, or long and unsigned short, say. For most applications this is not important and can be ignored, but sometimes you want to preserve the information. The values within the parentheses are

```
'E'   4 byte real
'D'   8 byte real  (default for value containing decimal point)
'I'   2 byte integer
'J'   4 byte integer (default for value without decimal point)
'U'   2 byte unsigned integer
'V'   4 byte unsigned integer
```

We will usually omit the type information for the two default cases.

We map a DM header key FOO to the following set of (long form) FITS header keywords:

```
FOO    =   value  / [unit] desc           CDM
DTYPEn =   'FOO'   /                       CDM
DUNITn =   'unit'  /                       CDM
DDISPn =   'disp'  /                       CDM
DFORMn =   'datatype' /                    (CDM controversial)
```

On reading, we set the name to be FOO, the unit to be first the DUNITn, next the value in [] after the / in FOO, finally to blank if neither of the preceding are there. The comment is set to be whatever is after the / in FOO with the exclusion of any [] token.

For long keyword names, keyword FOO is replaced by DVALn:

```
DVALn  =   value  / [unit] desc            CDM
DTYPEn =   'LONG_KEY_NAME'   /             CDM
DUNITn =   'unit'  /                       CDM
DDISPn =   'disp'  /                       CDM
DFORMn =   'datatype' /                    (CDM controversial)
```

The values of n must be unique in a given HDU block, but need not be consecutive, although it would be nicer to keep them so.

NOTE: We have just heard (Jan 99) that Bill Pence is implementing a different scheme, using HIERARCH keywords introduced by ESO, in CFITSIO. I haven't seen these keywords discussed in HEA forums, so I'm awaiting more details.

- DTYPEn gives the name of the Data Descriptor. This keyword must be present if any of DUNITn, DVALn, DFORMn, DDISPn, DDESCn are present, otherwise it must be omitted.
- DUNITn (string) gives the unit for the Data Descriptor. If the unit is blank, it should be omitted. The unit should also be copied to the root keyword comment as specified by the new CFITSIO convention.
- DVALn (arbitrary type) gives the element value for the attribute. If the attribute name in DTYPEn is 8 characters or fewer, the attribute name will be used as the keyword name instead of DVALn. On reading, the data type for the Data Descriptor is inferred from the format of the element value.
- DFORMn (string), if present, gives the data type for the element, overriding the data type inferred from the formatting of the value header keyword and the short form type convention. Omit for strings and signed numeric types.
- DDISPn (string) gives the Data Descriptor recommended display format; this should be used very sparingly, and is not yet implemented.
- DDESCn (string) gives the Data Descriptor description. If absent, the default value is the comment string following the / in the keyword containg the value. DDESC has not yet been implemented.
- DLMINn, DLMAXn to record the legal range of a descriptor. Default is -Inf to +Inf; only applies to numeric data types. This has not yet been implemented.
- MTYPEn and MFORMn and METYPn keywords may also be used to group keys.

1.7 Array keywords

We provide limited support for 1-D array key descriptors. Traditionally related values such as coefficients of polynomials have been written using indexed keywords, e.g. COEFF1, COEFF2, COEFF3... This provides an obvious model for array valued keys. However, indexed keywords have also been used for other purposes, so on read we cannot assume the presence of a trailing digit indicates an array keyword. Also, NAXIS and NAXISn are both defined keywords, and if we used the naive interpretation both would be descriptors with name NAXIS.

- DTYPEn: We therefore require that array keys be written using the DTYPEn keyword with the special syntax

  ```
  \item DTYPE3 = 'COEFF* '
  ```

- Here the asterisk is used to imply a set of array keywords. The general format is DTYPEn = 'NAME*'; if NAME is less than or equal to 7 characters, the values will be stored in keywords NAME1, NAME2, ... NAMEm.

- iDVALn: When the 'NAMEi' exceeds 8 characters, iDVALn will be used instead. Example:

  ```
  DTYPE4 = 'COEFFICIENT*'     / Array keyword
  4DVAL1 =      0.001         / Coeff for n=1
  4DVAL2 =      3.4E-6        / Coeff for n=2
  4DVAL3 =    -14.328         / Coeff for n=3
  ```

- On read, the dimension of the array is equal to the largest value of i present as a NAMEi. Missing values of i are set to zero or blank; elements of the array must be all numeric or all string.

1.8 Images

For Image Data descriptors, the following are existing FITS keywords:

- BUNIT (string) Unit of image data values (B is for 'brightness')
- BITPIX (integer) coded value implies the data type.
- BSCALE, BZERO values used e.g. for unsigned data types; handled by CFITSIO.

We plan to introduce

- BTYPE (string) Name of image data array. If absent, default to value of HDU name.
- BFORM (string) as DFORMn, to impose a data type interpretation overriding the BITPIX value.

These have not yet been added.

1.9 Coordinate Systems

Relevant docs:

```
 FITS standard,
ftp://fits.cv.nrao.edu/fits/documents/standards/fits_standard.ps
ftp://fits.cv.nrao.edu/fits/documents/standards/bintable_aa.ps
the WCS draft document,
ftp://fits.cv.nrao.edu/fits/documents/wcs/wcs.all.ps
and the OGIP94-006 document,
http://legacy.gsfc.nasa.gov/docs/heasarc/ofwg/docs/summary/ogip_94_006_summary.html
```

(all of which are mirrored in /proj/jcm/ASC/FITS/docs)

We will store coordinate info as follows: The general transform supported by DM has the following parameters, named according to the FITS keywords used in the FITS IMAGE implementation...

```
Dimension n
Transform type (string): ctype
Number of transform function parameters m (depends on ctype, usually = zero)
Transform function parameters (doubles): prop1 to propm
Reference pixel: crpix1 to crpixn
Reference value: crval1 to crvaln
Reference scale: cdelt1 to cdeltn
Rotation angle:  crota (only used in 2D case)
Rotation matrix: cd(n,n)
```

The rotation angle is zero and rotation matrix is unity for all our data at the moment. Recently, CDELT has been deprecated in favor of the CD matrix, but we are continuing to use CDELT anyway.

We distinguish between the first transform on a particular descriptor, which is considered the principal transform, and subsequent transforms. Slightly different keywords are used for principal and other transforms. In addition, different keywords are used for transforms for the following descriptor cases:

```
1) the axes of an image data array (Axis number j)
2) a table scalar column (FITS column number i)
3) the axes of a table array column (FITS column number i, axis p); not yet supported.
4) values of an image data array (not yet supported).
```

For the principal transform: (these are HEASARC proposed keywords)

```
Case    1                   2                   3
```

ctype	CTYPEj	TCTYPi	pCTYPi
crpix	CRPIXj	TCRPXi	pCRPXi
crval	CRVALj	TCRVLi	pCRVLi
cdelt	CDELTj	TCDLTi	pCDLTi
crota	CROTAj	TCROTi	pCROTi
cd	CDjj	TCDii	ppCDi

For subsequent transforms: (case 3 not supported; these are ADASS FITS BOF proposed keywords)

Case	1 or 2
ctype	CTYPEjk
crpix	CRPIXjk
crval	CRVALjk
cdelt	CDELTjk
crota	CROTAjk
cd	CDjjk

In this case k is a single upper case letter from A to Z. We reserve the choice of the letter P to flag the physical coordinate transform (IRAF's LTM/LTV) which maps original pixels to current logical pixels.

1.10 Coordinate systems on image block axes

Traditional use of CTYPE: Construction of the CTYPE keyword (or TCTYP, etc): In a classic piece of broken design, we use CTYPE to store both the name of the coordinate descriptor quantity and the name of the projection. The hack is as follows: for now, we support only 1-D LINEAR transforms and 2-D WCS spherical projections. if the transform is not LINEAR, it must be one of the WCS projections. In this latter case, there are a pair of CTYPEs, CTYPEn and CTYPEm (hopefully with m = n + 1). The value of each of these is an 8 byte string; the first 4 bytes contain the axis name padded with trailing dashes, and the last 4 bytes contain the transform code padded with leading dashes. The only allowed value pairs for the axis names are:

RA--	DEC-	Equatorial
GLON	GLAT	Galactic
ELON	ELAT	Ecliptic
HLON	HLAT	Helioecliptic
SLON	SLAT	Supergalactic
PLON	PLAT	Planetary
XLON	XLAT	Generic latitude and longitude

We add the extra names

```
LONG     NPOL       Generic with north polar angle not latitude
```

This is used only with the TAN transform and is useful for a WCS for telescope off-axis angle and azimuth.

The allowed values for the transform type are:

-TAN, -AZP, -SIN, -STG, -ARC, -ZPN, -ZEA, -AIR, -CYP, -CAR, -MER, -CEA, -COP, -COD, -COE, -COO, -BON, -PCO, -GLS, -PAR, -AIT, -MOL, -CSC, -QSC, -TSC.

If the CTYPE value does not include the dash character '-' in byte 5, we may assume it is a LINEAR transform in which case the descriptor name is the full value of CTYPE.

For the ASC DM we introduce the following extra keywords:

- CNAMEn Name of axis (overrides value of CTYPEn, used in case where CTYPE is not a LINEAR transform to override the standard component names like RA and DEC; i.e. when XLON and XLAT are present in CTYPE.)
- CUNITn Unit of axis (FITS standard keyword)

We also support the use of MTYPEn, MFORMn for defining composite axes. Their use is entirely analogous to their use with table columns.

1.11 Coordinate Data Descriptors: columns

We note the following existing HEASARC FITS keywords and their use:

- TCTYPj (string) is used for the coordinate descriptor component name, (or descriptor name for a non-composite descriptor), and transform type.
- TCNAMj (string) Like CNAMEn for images.
- TCUNIj (string) is used for the coordinate descriptor unit.
- TCDLTj (real) is used for the Data Coordinate Transform scale.
- TCRPXj (real) is used for the Data Coordinate Transform reference pixel element value.
- TCRVLj (real) is used for the Data Coordinate Transform reference world element value.
- RADECj (string) and EQUINj (real) are used for RA, DEC column pairs to give the system ('ICRS', 'FK4', 'FK5') and equinox (2000.0, 1950.0, etc)

We also want to give a name to a composite (2D) coordinate descriptor. For this we introduce the ASC-defined keyword

- MCTYPn (string) is used for the Data Coordinate descriptor name for the primary coordinate system attached to composite column MFORMn. In other words, if MFORM4 = 'X,Y' and TTYPE3='X', TTYPE4='Y' then the coord descriptor name is the value of MCTYP4 and the coord descriptor component names are values of TCTYP3 and TCTYP4, with transform values given by TCRPX3/4 etc.

1.12 Coordinate descriptors for array column axes

The following keywords are all HEASARC-specified.

- iCTYPj is used for the Component Name for an Axis Group Coordinate quantity corresponding to the i'th axis.
- iCUNIj is the unit of the axis group coordinate quantity.
- iCRPXj is the reference pixel value for the axis group quantity in the axis group coordinate transform.
- iCRVLj is the reference world value for the axis group coordinate quantity in the axis group coordinate transform.
- iCDLTj is the transform scale in the axis group coordinate transform.

1.13 Keywords for recording filters

This section describes the keywords used by the ASCDM Data Subspace code.

Suppose we filter a file with the constraint

```
MASS = 14.2:230.1,GRADE=1:5,10:12,14:23
```

In the output FITS file this will be recorded as

```
DSTYP1 = 'MASS'      / Rest Mass
DSUNI1 = 'kg  '      / Unit for DSTYP1
DSVAL1 = '14.2:230.1' / Range for DSTYP1
DSTYP2 = 'GRADE'     /
DSVAL2 = '1:5,10:12,14:23'  / Ranges for DSTYP2
```

Note: The Data Subspace conventions are internal to ASC and are not agreed as part of the CDM.

The example GRADE above but with 30 values instead of 3 would be better stored as a table, as follows:

```
DSTYP2 = 'GRADE'          /
DSVAL2 = 'TABLE'          / Values are in a table
DSREF2 = ':GRADE_FILTER'  / Name of table
```

and in an HDU elsewhere in the file:

```
XTENSION='BINTABLE'
NAXIS1  =     8
NAXIS2  =    30
TFIELDS =     2
TTYPE1  = 'GRADE_MIN'
TFORM1  = '1J'
TTYPE2  = 'GRADE_MAX'
TFORM2  = '1J'
EXTNAME = 'GRADE_FILTER'
MTYPE1  = 'GRADE'
MFORM1  = 'GRADE_MIN,GRADE_MAX'
METYP1  = 'R'
```

similar to the GTI table given above. The colon before the table name was recommended as part of a broader scheme to specify URLs for FITS HDUs; I'm not sure how standard it will be.

When there is more than one DSS component, we need to generalize these keywords. We prefix abbreviated versions of the keywords with the DSS component number:

- iDSVALj instead of DSVALj
- iDSREFj instead of DSREFj
- The same filter (value of j) in components 2 onwards must share the same name (DSTYPj), unit (DSUNIj), and data type. So we don't need keywords for those.

The presence of an iDSVALj (or iDSREFj) keyword for any value of j implies the existence of component i. If iDSVALj exists but iDSVALk does not, the value of iDSVALk is assumed to be the same as DSVALk. The idea here is that components will often have many filters in common, and just a couple that are different.

Here is an example with components: it represents a merged spectrum list with different extraction radii for different energies.

```
DSTYP1  = 'ENERGY'    / Energy
DSUNI1  = 'keV  '     / Unit for DSTYP1
DSTYP2  = 'RADIUS'    / Extraction radius
DSUNI2  = 'pixel'     / Unit for DSTYP2
```

```
DSTYP3  = 'GAIN'      / Calibration gain
DSVAL1  = '0.1:2.0'   / Range for Energy
DSVAL2  = '14'       / Extraction radius
DSVAL3  = '2:'       / Range for gain
2DSVAL1 = '2.0:5.0,8.0:10.0'   / Energy range, 2nd component
2DSVAL2  = '30'      /  Extraction radius
```

This means that the data contains energies in the range 0.1 to 2.0 keV extracted in a radius of 14 pixels (around some point), and also energies in the ranges 2 to 5 and 8 to 10 keV, all extracted in a radius of 30 pixels. The data was also selected in all cases for a gain between 2 and infinity. (there is no 2DSVAL3 so the gain for the second component is assumed to be the same as for the second component, i.e. DSVAL3) Datasets like this usually arise from merging two datasets with a single component in their data subspace. One might write the above DSS as a logical expression:

```
{  [(ENERGY in 0.1:2) AND (RADIUS = 14)]     OR
   [(ENERGY in 2:5,8:10) AND (RADIUS = 30)] }
AND (GAIN >2 )
```

Another more realistic case is multiple GTIs for different ACIS chips:

```
DSTYP1  = 'CCD_ID'      / Chip number
DSTYP2  = 'TIME'        / Time
DSUNI2  = 's   '        / Unit for DSTYP2
DSTYP3  = 'PHA '        / PHA
DSVAL1  =         0     / Chip ACIS-I0
DSVAL2  = 'TABLE'      / DSTYP2 ranges are in BINTABLE HDUs
DSREF2  = ':GTI0'      /  Good times for chip 0
DSVAL3  = '2:1024'      / Good PHA range
2DSVAL1 =         1    /  Chip ACIS-I1
2DSREF2 = ':GTI1'      /  Good times for chip 1
3DSVAL1 =         2    /  Chip ACIS-I2
3DSREF2 = ':GTI2'      /  Good times for chip 2
4DSVAL1 =         3    /  Chip ACIS-I3
4DSREF2 = ':GTI3'      /  Good times for chip 3
5DSVAL1 =         6    /  Chip ACIS-S2
5DSREF2 = ':GTI6'      /  Good times for chip 6
6DSVAL1 =         7    /  Chip ACIS-S3
6DSREF2 = ':GTI7'      /  Good times for chip 7
```

Note no DSUNI1 keyword is written since Chip number doesn't have a unit. Logically this DSS translates to:

```
(PHA in 2:1024) AND {
 ( CCD_ID = 0 AND TIME = GTI0 ) OR (CCD_ID = 1 AND TIME = GTI1 )
OR ...}
```

1.13.1 Special cases

For back compatibility for non-compliant files, the following special cases are recognized on reading and writing: (only the first is currently done by our implementation).

1. On writing, if the axis name is TIME, write the data cell for the first DSS component as TABLE GTI. On reading, if a GTI extension exists, interpret it as a data subspace data cell on quantity TIME.

2. Also calculate the sum of the GTI intervals and store as an attached attribute ONTIME, and multiply by an attached attribute DTCOR if present, to generate another attached attribute LIVETIME. On reading, interpret the keywords ONTIME, DTCOR and LIVETIME as attached attributes to the TIME data subspace axis.

3. Recognize the HEASARC timing keywords SCSEQBEG, SCSEQEND, DATE-OBS, TIME-OBS, DATE-END, TIME-END, ONTIME, MJD-OBS, TSTART, TSTOP, MJDREF and their variations, all of which are associated attributes of the TIME data subspace axis. In the absence of a GTI record, use the start and stop times in mission time or, if no mission time is available, JD or MJD, as deduced from these keywords, to define a single time range element for the data subspace.

4. On writing, if the axis name is PHA, write the data cell as the indirect spec [MINCHAN:MAXCHAN]. On reading, check for the MINCHAN and MAXCHAN keyword pair and interpret as a data subspace data cell on axis PHA.

5. A coordinate system on a data subspace axis whose components are RA and DEC will have its reference world element repeated as header keywords RA_NOM and DEC_NOM. On reading, these keywords will be recognized as the component names for a 2D data subspace axis on sky pixel position.

Note: Writing data subspaces to FITS files could be complicated. Suppose we have a data subspace cell which needs to be written as a table (e.g. GTI). We normally would write all structural information first, before writing the rows of the main Table Data section, but now we need to write a separate BINTABLE extension. Perhaps the best solution is to make an initial pass through the data subspace first and write all data subspace extensions before beginning the main ASC Table extension.

2 Systematic algorithm for creating descriptors

This section was written to help the ASCDM implementors, but I'll include it in here in case it clarifies the earlier sections.

I'll describe this from the point of view of both writing and reading a header. The central attribute of any descriptor is its name, so we look for that first. I then give the order of priority for each piece of information. Thus the column descriptor name is given as 'MTYPE,TTYPE', meaning that on read you first look for MTYPE, and then if there is no MTYPE you look for TTYPE; on write you start from the other end and use TTYPE if you can, but if TTYPE is insufficient to encode the information (e.g. it's a vector column) you use MTYPE.

2.1 Existence of a descriptor on read

- Descriptors that may exist without explicit names in the file are the image data descriptor and image axis descriptors. Their existence is forced by the presence of an image block.
- Column descriptors must have corresponding TTYPE or MTYPE keywords
- Key descriptors must have a DTYPE keyword or an ordinary FITS header keyword that is not an ASCDM-reserved name.
- Coordinate descriptors must have a CTYPE (TCTYP, iCTYP) or CCTYP keyword.

2.2 Descriptor names

- Column descriptor: MTYPE if composite, else TTYPE
- Key: MTYPE, else DTYPE if needed, else keyword name
- Image data descriptor: BTYPE, else EXTNAME, else "IMAGE".
- Filter: DSTYP
- Coord attached to column: MCTYP, else TCTYP
- Image axis coord: MTYPE, else CCTYP, else CTYPE

Special case: if a column descriptor has an attached coordinate system TCTYP which is not a LINEAR transform, it must be a vector column of dimension 2, paired with some other column with a matching TCTYP, even if there is no MFORM keyword. For example if we have

```
TTYPE3 = 'X'
TTYPE4 = 'Y'
TCTYP3 = 'RA---TAN'
TCTYP4 = 'DEC--TAN'
```

we pretend that the keywords MTYPE1 = 'POS', MFORM1 = 'X,Y', MCTYP1 = 'EQPOS' were actually present. The column component names are 'X', 'Y' and the coordinate component names are 'RA', 'DEC'. The game is:

- I have a TTYPE; is it part of an MFORM?
- If not, does it have a TCTYP that needs a partner? RA—TAN needs DEC–TAN as its partner.
- If so, make up an MTYPE name and an MCTYP name for it (see below).
- If not, it is a simple column and TTYPE gives its name.

Now how will we assign these MTYPE and MCTYP names? We are going to recognize the following special cases:

Component names	Default composite name
X,Y	POS
X, *Y	*, for all *
RA,DEC	EQPOS
GLON,GLAT	GALPOS

and in all other cases do POSn, where n is some unique integer. This means that TDETX, TDETY maps to TDET.

2.3 Descriptor component names

- Column descriptor: TTYPE
- Key: DTYPE or keyword name
- Image dd: not supported
- Filter: DSCPT
- Column coord: TCTYP
- Axis coord: CTYPE

Here's an interesting question. For 2D filters, the natural thing is to specify a single DSTYP and DSVAL, with the DSVAL being a region string. But we need somewhere to put two (optional) component names and internally we need to point to two associated column descriptors. So do we go with the MTYPE paradigm and have MDTYP1 = 'POS', MDFOR1 ='X,Y', DSTYP1 = 'X', DSTYP2 = 'Y' and instead of having two DSVALs have an MDVAL with MDVAL1 = 'circ 2 30 2'? Or do we make DSTYP the thing that can be composite, and have instead DSTYP1 = 'POS', DSVAL1 = 'circ 2 30 2', DSCPT1 = 'X,Y' ? I think the latter is much more consistent with the rest of data subspace, and I like it better, so I'm going with it for now. Comments welcome.

2.4 Element dimensionality

- Column descriptor: infer from MFORM (and TCTYP) else 1
- Key: always 1 for now
- Image dd: always 1 for now
- Filter: infer from DSCPT, not to be implemented yet
- Column coord: infer from MFORM of parent
- Axis coord: infer from MFORM

2.5 Descriptor units

- Column descriptor: TUNIT
- Key: DUNIT, else Pence convention on keyword with the name
- Image dd: BUNIT
- Filter: DSUNI
- Column coord: TCUNI
- Axis coord: CUNIT

2.6 Descriptor values

- Column dd: current row and cell
- Key: DVAL or keyword value
- Image dd: image data

- Filter: DSVAL
- Column coord: via transform
- Axis coord: via transform

2.7 Data type

- Column dd: TFORM
- Key: DFORM or infer from format of value
- Image dd: BFORM, else BITPIX
- Filter: DSFORM, else type of assoc-col
- Column coord: infer from transform type, default double
- Axis coord: infer from transform type, default double

2.8 Descriptor desc

- Column dd: TDESC or / comment after name keyword
- Key dd: DDESC or / comment after value keyword
- Image dd: BDESC or / comment after BTYPE
- Filter: DSDSC or / comment after name keyword
- Column coord: TCDSC or / comment after name keyword
- Axis coord: CDESC or / comment after name keyword

2.9 Display format

- Column dd: TDISP
- Key: DDISP
- Image dd: not supported
- Filter: not supported
- Column coord: not supported
- Axis coord: not supported

2.10 Element type

- Column dd: METYP, default to V
- Key: METYP
- Image dd: METYP
- Filter: always R
- Column coord: inherit from parent
- Axis coord: inherit from parent

2.11 Array dimensionality

- Column dd: Infer from TFORM and TDIM
- Key: Always 0 for now
- Image dd: NAXIS
- Filter: Always 1
- Column coord: Always 0
- Axis coord: Always 0

2.12 Array sizes

- Column dd: Infer from TFORM and TDIM
- Key: n/a for now
- Image dd: NAXISn
- Filter: infer from DSVAL string
- Column coord: n/a
- Axis coord: n/a

2.13 Legal range

- Column dd: TLMIN/TLMAX
- Key: DLMIN, DLMAX
- Image dd: not yet supported
- Filter: n/a
- Column coord: not supported
- Axis coord: not supported

2.14 Transform type

- Column coord: Infer from TCTYP
- Axis coord: Infer from CTYPE

2.15 Transform values

- Column coord: TCRVL and TCRPX and TCDLT
- Axis coord: CRVAL and CRPIX and CDELT

3 Proposed future conventions

In this section I describe possible keywords that we might use in later releases; these aren't final but give an idea of the directions we're considering.

3.1 Future enhancement for ASC 'element types'

How many BINTABLE columns are used for a single ASC Table column? We will store a value element of dimensionality d in d separate columns of the table. A range element will require $2d$ columns, and a value with uncertainty will thus require $3d$ columns. A 2D region descriptor may be stored as a string in a single column; details of the implementation are to be worked out. The element type is stored in a special string keyword METYPn tied to the MTYPE/MFORM keywords. If it is absent, the default type V (value) is assumed. Thus, by knowing the element dimensionality d_i and element type t_i for each ASC Table column, we can assign the mapping to BINTABLE columns. The number of BINTABLE columns for ASC Table Column i is $d_i N_{BT}(t_i)$ as tabulated below:

t_i	$N_{BT}(t_i)$	Description
V	1	Value only
B	2	Bin
BF	1	Fixed width bin
S	2	Bin start
SF	1	Fixed width bin start
I	3	Value plus interval
R	2	Interval only
K	2	Scale range
KF	1	Fixed scale range
L	3	Two sided scale (log)
LF	1	Fixed two sided scale
U	2	One sided uncertainty
UF	1	Fixed one sided uncertainty
T	3	Two sided uncertainty
TF	1	Fixed two sided uncertainty
REG	1	2D region

Then the total number of BINTABLE columns required for the c columns of the ASC Table is

$$\text{TFIELDS} = \sum_{i=1}^{c} d_i N_{BT}(t_i)$$

and the starting BINTABLE column number j for ASC Table column i is

$$j(i) = 1 + \sum_{k=1}^{i-1} d_k N_{BT}(t_k)$$

The following proposed keywords are reserved for possible future ASCDM implementation.

- MITYPn (string) gives the interval type for elements of type VU or R. Possible values are '[]' (default), '()', '(]', '[)'.
- MULEVn (real) gives the uncertainty level, between 0.0 and 1.0, default 1.0.
- MDTYPn (string) gives the data type, overriding the default data type in TFORMj. This allows us to add data types which are not directly supported by the FITS standard. Values of MDTYPn are specified in the data model design document.
- MUMINn gives the uncertainty lower value for a fixed uncertainty. Default CUVALn.
- MUMAXn gives the uncertainty upper value for a fixed uncertainty. Default CUVALn.

- MUVALn gives the uncertainty value for a fixed uncertainty. Default zero.
- MZTYPn (string) gives the element type for the systematic zero point uncertainty.
- MSTYPn (string) gives the element type for the systematic scale uncertainty.
- MZMINn, MZMAXn, MZVALn give the systematic zero point uncertainty values.
- MSMINn, MSMAXn, MSVALn give the systematic scale uncertainty values.

3.2 Element types

To make a column of element type INTERVAL or RANGE, go as follows:

```
 dmColumnCreateInterval( table, name, type, unit, desc, cptNames, elementType )
where
 name = "TIME", type = dm_DOUBLE, unit = 's', desc = 'Spacecraft Time',
 cptNames = { "TSTART", "TSTOP" }, elementType = dmRANGE.
```

Suppose the next available FITS column is 4, and this is the second CDM grouped column. In the FITS file this should map to:

```
TTYPE4 = 'TSTART  '            / TIME
TFORM4 = '1D      '            / Data type for column 4
TUNIT4 = 's       '            / Unit for column 4
TTYPE5 = 'TSTOP   '            / TIME
TFORM5 = '1D      '            / Data type for column 5
TUNIT5 = 's       '            / Unit for column 5
MTYPE2 = 'TIME    '            / Spacecraft Time
MFORM2 = 'TSTART,TSTOP'        / CDM meta-column
METYP2 = 'R'                   / ASC Element Type is Range
```

3.3 Extra keys for header keywords

The following proposed keywords are reserved for possible future implementation

- DUMINn gives the uncertainty minimum value.
- DUMAXn gives the uncertainty maximum value.
- DITYPn (string) gives the interval type for elements of type VU or R. Possible values are '[]' (default), '()', '(]', '[)'.

- DULEVn (real) gives the uncertainty level, between 0.0 and 1.0, default 1.0.
- DARELn (string) gives the name of the 'parent' Data Descriptor to which the attribute is related - either the name of a column or of another attribute. This allows us to define attributes which belong to individual columns.

3.4 Data Subspace keys

The following proposed keywords are reserved for possible future implementation

- DSITYPj interval type for axis j.
- DSjLm, DSjUm: lower (L) and upper (U) interval values for a range. Implies

  ```
  DSj = (DSjL1:DSjU1),(DSjL2:DSjU2),...(DSjLn,DSjUn)
  ```

- DSCNAMn (string): Name of coord quantity for axis group n. This and other coordinate info defaults to the coordinate system on any table column mapping to the DSS axis.
- DSCTYPj (string): Name of coord quantity component for axis j.
- DSCDLTj (string): Transform scale for axis j.
- DSCRPXj (real): Reference pixel value for coord transform
- DSCRVLj (real): Reference world value for coord transform.
- DSCUNIj (string): Unit for coord quantity
- DSCj (string): Data Cell in world coord units
- DSCjLm, DSCjUm: Data Cell min and max element values (shares interval type for Data Descriptor, i.e. DSITYPj).

The data cell specification is parsed as follows: (this is a long term goal, not to be implemented for the time being):

1. If the first character is numeric, [or (, the spec is a range.
2. Otherwise, if the first (space-delimited) word is the special string TABLE, it is a table specification.
3. Otherwise, it is a 2D region specification.

4. For a range, we determine the interval type by locating the closing parenthesis. The allowed types are [],[),(],() representing open, closed and semi-open intervals. No parentheses means closed interval.

5. Within the parentheses, we seek a colon to parse the string as [a:b]. If no colon is present, interpret as [a:a]. If a or b is non-numeric, interpret them as the names of other header keywords (an indirect range specification). The purpose of the indirect range specification is to allow us to continue to write back compatible files.