FITS Keyword Conventions in CXC Data Model files
SDS-7.3

Jonathan McDowell

November 10, 2002

Contents

1 Introduction

This document describes some of the FITS keywords used in Chandra data files. The summary lists them and the remainder of the document describes some of them in more detail.

2 Summary of keywords

Data model keywords:

CNAMEn	Override CTYPEn image coordinate axis name
CPREF	List of 'preferred cols' for making image from table
DSTYPn	Data Subspace column name
DSFORMn	Data Subspace data type name (optional)
DSUNITn	Data Subspace unit name (optional)
DSVALn	Data Subspace filter list
iDSVALn	Same as DSVALn for component i
DSREFn	Data Subspace table pointer
iDSREFn	Same as DSREFn for component i
DTYPEn	Name for composite long-named keyword (cf. CFITSIO HIERARCH)
	Also used to define array keywords
DUNITn	Unit for composite keyword
DVALn	Value for composite keyword
HDUNAME	Gives a name to an HDU (defaults to EXTNAME/EXTVER)
METYPk	Type of composite column (not yet supported)
MFORMk	Comma-separated list of column names making up composite col (with MTYPE)
MTYPEk	Composite column name (paired with MFORM)
TCNAMn	Override TCTYPn table coordinate axis name
TDBINn	Default binning factor for table column
TDNULLn	Allows for floating point null value other than NaN

Also note use of CTYPEnP to represet a 'physical' linear coordinate system mapping blocked (rebinned) to original pixels.

Also note the special values CTYPEi = 'LONG-TAN', CTYPEj = 'NPOL-TAN' defining a nonstandard WCS with the latitude equal to zero rather than 90 at the pole, useful for representing a telescope off-axis angle.

The following CXC standard header keywords have also been introduced:

MISSION	Grouping of related telescopes
OBS_ID	Observation id
SEQ_NUM	Sequence number

ASCDSVER	Processing system revision
DEFOCUS	Defocus distance of instrument in mm rel to best
FOC_LEN	Telescope focal length in mm
OBS_MODE	Pointing or slewing or ground cal
DATAMODE	Configuration of on-board processing
READMODE	Configuration of instrument
DATACLAS	Observed or simulated
ONTIME	Sum of GTIs
DTCOR	Dead time correction, 0.0 - 1.0
LIVETIME	ONTIME times DTCOR
GAINFILE	CALDB file for gain correction
GRD_FILE	CALDB file for grade correction

MISSION was introduced to group files that included both flight data for Chandra (TELESCOP='CHANDRA') and ground calibration data for instruments and test mirrors, with different values of TELESCOP.

OBS_ID and SEQ_NUM label the observations - SEQ_NUM refers to an individual exposure and OBS_ID refers to a group of exposures to be processed together (although in practice it's very rare for there to be more than one exposure per OBS_ID).

We also use the CFITSIO convention keywords LONGSTRN, CHECKSUM, DATASUM and the HEASARC keywords TIMESYS, MJDREF, TIMEZERO, TIMEUNIT, CLOCKAPP, TSTART, TSTOP, TIERRELA, TIERABSO, TIMVERSN, TIMEPIXR, TIMEDEL, RA_NOM, DEC_NOM, ROLL_NOM, RA_TARG, DEC_TARG.

3 CXC Data Model Conventions

The CXC data model implementation uses a number of new FITS header keyword conventions. The guiding principle used in their interpretation is to select defaults so that FITS files without such keywords will still be correctly interpreted by the data model. The new keywords are as far as possible chosen to be analogous to existing FITS conventions.

3.1 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 in CXC files 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.

3.2 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; CXC DM 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).

3.3 Extra information for compound columns

At the CXC'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.
- MTYPEn (string) gives the name of the composite descriptor defined by MFORMn.
- 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. As of Jan 2003, METYPn support has not yet been implemented.

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.

3.4 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.

3.5 Extra information for header keys

On reading a FITS header, all the mandatory FITS keywords and the keywords defining the BINTABLE/IMAGE and overlying DM 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, as per CFITSIO, is

```
FOO   =  value  / [unit]  desc
```

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
DFORMn =   'datatype' /                    (CDM not implemented)
```

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
```

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 that CFITSIO uses a different convention, called HIERARCH).

- 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.
- MTYPEn and MFORMn and METYPn keywords may also be used to group keys.

3.6 Array keywords

Our software provides 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

```
DTYPE3 = 'COEFF* '
```

This tells the software that the COEFFn keys should be read in as a single array object rather than as a bunch of different scalar objects with similar names.

- 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.
- 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.

3.7 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.

3.8 Coordinate Systems

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)
```

CDELT and CROTA have been deprecated in favor of the CD matrix, but we use them 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.

3.9 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 include:

-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 CXC DM we introduce the following extra keyword:

- CNAMEn (TCNAMn for tables) 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.)

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

3.10 Keywords for recording filters

This section describes the keywords used by the CXC DM 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
```

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 ...}
```