CORONAL DIAGNOSTIC SPECTROMETER

SoHO

CDS SOFTWARE NOTE No. 4

Version 1.1 30 December 1994

IDL SOFTWARE FOR FITS BINARY TABLES

W. Thompson
Applied Research Corporation
NASA Goddard Space Flight Center
Laboratory for Astronomy and Solar Physics
Code 682.1
Greenbelt, MD 20771, USA

William.T.Thompson.1@gsfc.nasa.gov
pal::thompson

This software package was developed to support reading and writing FITS binary table extensions. The primary source for a description of the FITS binary table extension is the document "Binary Table Extension to FITS: A Proposal" by W. D. Cotton and D. Tody, hereafter referred to as C&T. This information is also included almost verbatim in the draft standards document "Implementation of the Flexible Image Transport System (FITS)" from the NASA/OSSA Office of Standards and Technology (NOST). As well as routines that are directly related to binary tables, there are also supporting routines which allow the user to read and write primary FITS header and data units (HDUs), and to copy FITS files to and from tapes.

1 Primary FITS Header and Data Units

Every FITS file must contain a primary Header and Data Unit (HDU). This part represents the most basic kind of FITS file. It allows for the storage of an array with NAXIS dimensions, but one can also have a primary HDU which only consists of a header without any data (signalled in the header by setting NAXIS=0). All FITS extension HDUs, must then follow the primary HDU.

The following routines are for writing primary FITS header and data units. They are listed in the order they would be used to create a basic FITS file to which binary table extension could then be appended.

FXHMAKE: Creates the primary FITS header based on the array passed as an optional parameter. If no data array is passed, then a header array consistent with no data in the primary HDU is created. When appending FITS extensions, such as binary tables, to the file, FXHMAKE should be called with the /EXTEND qualifier to generate a header record that reads "EXTEND=T" (true).

FXADDPAR: Adds keyword=value records to the primary header, or modifies those that are already there. The use of this routine is entirely at the user's discretion—the required keywords are inserted by FXHMAKE.

FXWRITE: Writes the primary header and (optional) data array to a FITS file on disk.

There are a number of routines available to read and interpret FITS primary HDUs. For example, the IDL Astronomy User's Library, available via anonymous ftp from idlastro.gsfc.nasa.gov, contains a number of routines including READFITS. The following lists two routines included in this package to support this function.

FXREAD: Reads a basic FITS file, or the primary HDU from a FITS file with extensions. This routine can either read the entire array, or a subarray. It also gives the user the option of reducing the amount of data read either by reading every nth pixel, or by averaging together $n \times n$ pixels.

FXPAR: Extracts values from FITS headers.

2 Creating FITS binary table files

The processes of creating a FITS binary table file is fairly easy. However, there are enough individual steps involved in creating the file, that it is anticipated that it will be carried out primarily using procedure files, rather than interactively from the keyboard.

The routines used in writing FITS binary tables files are:

FXBHMAKE: Starts the definition of the FITS binary table header. At this time the number of rows in the table is defined, but no information about the individual columns is as yet included. Optionally, the extension is given a name.

FXBADDCOL: Add the information about the columns to the binary table extension header. This routine is called once for each column that will be in the table.

FXBCREATE: Opens the FITS file that the binary table will be appended to and writes the header.

FXBWRITE: Writes the data into the binary table. A separate call to FXBWRITE is made for every row and and every column in the table.

FXBFINISH: Finishes up and closes the file containing the newly created binary table.

The basic steps involved in creating a FITS binary table extension file are as follows:

- First the primary FITS data unit must be created. At the very minimum, this will include a FITS header with the keyword "EXTEND" set to T(rue). It may also have data associated with it, or it may simply have "NAXIS" set to zero to signal that there is no primary data array. This step is carried out through the FXHMAKE and FXWRITE routines.
- Next, the binary table extension header must be created, and the various columns to be used have to be defined. The routines FXBHMAKE and FXADDCOL take care of this.
- Then, the extension file must be opened, and the header written out. FXBCREATE takes care of this.
- The next step is to actually write the data arrays themselves into the table. This is done using multiple calls to FXBWRITE.
- Finally, the table file is closed with the FXBFINISH command.

The following IDL statements demonstrate how to use these routines to create a simple binary table with three columns and five rows. Some test arrays are generated to write into these columns, and are slightly modified for each row, to make the test more complete.

```
;  Create the data to write to the binary table.
;
a = intarr(10,100)
b = reverse(intarr(20,100),2) - 100
c = fix(dist(50))
;
;  Create a primary header and write it out.
;
fxhmake,header,/initialize,/extend,/date
fxwrite,'sample.fits',header
```

```
;
;  Create a binary table extension header for a table with 5 rows.
;
fxbhmake,header,5,'TESTEXT','Test binary table extension'
;
;  Create the columns for the a, b, and c arrays.
;
fxbaddcol,acol,header,a,'Column 1'
fxbaddcol,bcol,header,b,'Column 2'
fxbaddcol,ccol,header,c,'Column 3'
;
;  Write out the extension header.
;
fxbcreate,unit,'sample.fits',header
;
;  Write out the data.  For each row, multiply the test arrays by the row
;  number.
;
for i=1,5 do fxbwrite,unit,a*i,acol,i
for i=1,5 do fxbwrite,unit,b*i,bcol,i
for i=1,5 do fxbwrite,unit,c*i,ccol,i
;
;  Close the binary extension.
;
fxbfinish,unit
end
```

The primary FITS header created by this routine is very simple, and looks like this

```
SIMPLE  =                    T /Written by IDL:  30-Jan-1992 11:19:34.00
BITPIX  =                    8 /
NAXIS   =                    0 /
EXTEND  =                    T /File contains extensions
DATE    = '30/01/92'           /
END
```

And the binary table extension header looks like this

```
XTENSION= 'BINTABLE'           /Written by IDL:  30-Jan-1992 11:35:49.00
BITPIX  =                    8 /
NAXIS   =                    2 /Binary table
NAXIS1  =                11000 /Number of bytes per row
NAXIS2  =                    5 /Number of rows
PCOUNT  =                    0 /Random parameter count
GCOUNT  =                    1 /Group count
TFIELDS =                    3 /Number of columns
EXTNAME = 'TESTEXT '           /Test binary table extension
```

```
TFORM1  = '1000I   '          /Integer*2 (short integer)
TTYPE1  = 'COLUMN 1'          /Label for column 1
TDIM1   = '(10,100)'          /Array dimensions for column 1
TFORM2  = '2000I   '          /Integer*2 (short integer)
TTYPE2  = 'COLUMN 2'          /Label for column 2
TDIM2   = '(20,100)'          /Array dimensions for column 2
TFORM3  = '2500I   '          /Integer*2 (short integer)
TTYPE3  = 'COLUMN 3'          /Label for column 3
TDIM3   = '(50,50) '          /Array dimensions for column 3
END
```

One thing that sometimes confuses first time users is that UNIT is an output parameter of FXBCREATE, not an input parameter. One does not specify the logical unit number for the file; instead FXBCREATE assigns one via a call to GET_LUN. A similar situation exists in regard to the column number argument to FXBADDCOL. Column numbers are assigned in the order in which FXBADDCOL is called. In the above example, the values of acol, bcol, and ccol will be returned by FXBADDCOL as 1, 2, and 3 respectively.

3 Reading FITS binary table files

The process of reading FITS binary tables is much simpler than writing them, since the structure of the table is already set. There are only three basic steps in reading a FITS binary table file:

- The file is opened, and the binary table extension selected, with the routine FXBOPEN.
- Data is read from the table with the routine FXBREAD. A particular row and column can be read, or an entire column can be read into a single array (except for columns containing variable-length arrays).
- The file is closed with the routine FXBCLOSE.

For instance, the binary table created in the above example could be read with the following statements.

```
IDL> FXBOPEN,UNIT,'sample.fits','testext',header
IDL> FXBREAD,UNIT,A,'Column 1'
IDL> FXBREAD,UNIT,B,'Column 1'
IDL> FXBREAD,UNIT,D,'Column 1'
IDL> FXBCLOSE,UNIT
IDL> HELP,A,B,C
A               INT       = Array(10, 100, 5)
B               INT       = Array(20, 100, 5)
C               INT       = Array(50, 50, 5)
```

Note that, because the entire columns were read in, the arrays A, B, and C each have an extra last dimension of 5. Also, the same comment in Section 2 above about FXBCREATE and UNIT also applies to FXBOPEN.

The following routines support reading FITS binary table extensions:

FXBOPEN: Opens a FITS binary table extension for reading. One can open several binary tables at once, either in the same or different files, by specifying different variable names for FXBOPEN to store the logical unit number into.

FXBREAD: Reads data from a column in a FITS binary table. One can read an entire column, a range of rows within a column, or a single row and column combination.

FXBTDIM: Parses keywords from binary tables with a TDIM-like format. See Section 5 for more information.

FXBHELP: Prints to the screen a simple table giving information about each of the columns in the binary table.

FXBFIND: Finds the values of keywords in the header associated with the binary table columns. For example, the command

```
fxbfind, header, 'TTYPE', columns, values, n_found
```

would return an array containing the values of the keywords TTYPE1, TTYPE2, etc., the columns for which they were found, and how many were found.

FXBCOLNUM: Returns the column number of a FITS binary table specified either as a number or by name.

FXBHEADER: Returns the header of a FITS binary table opened with FXBOPEN. (Note that the header can also be returned as an optional parameter of FXBOPEN.)

FXBISOPEN: Returns whether or not a logical unit number points to a FITS binary table that is open for read.

FXBSTATE: Similar to FXBISOPEN, but returns a state variable denoting whether or not a logical unit number points to an open FITS binary table, and if so whether that table is open for read or for write.

FXBCLOSE: Closes a FITS binary table that had been opened with FXBOPEN.

4 Tape I/O

The following routines are for writing and reading FITS files to and from tape. Each has a menu driven interface, so that the user only has to enter the name of the routine itself, and everything else is prompted for. Currently, these routines are only supported under VMS.

FXTAPEREAD: Reads FITS files from tape to disk. This routine is for standard FITS tapes, i.e. raw tape files separated by filemarks, with a blocking factor of $N \times 2880$ bytes, where N is between 1 and 10. The selected files are copied from tape to disk as a byte stream, without any conversion being applied.

FXTAPEWRITE: Writes FITS files from disk to tape. This routine writes standard FITS tapes. Optionally a user-selected keyword can be inserted into the primary header as written to tape to store the filename of each FITS file. Otherwise, the selected files are copied from tape to disk as a byte stream, without any conversion being applied.

5 Multidimensional Array Facility

The routines described here provide direct support for the Multidimensional Array Facility described in C&T. This convention uses keywords TDIMn of the format

TDIMn = '(D_1, D_2, ...)'

to define the dimensions associated with column n. For an example of the use of this keyword, see the sample binary table header in Section 2 above.

Support for this convention is automatic—FXBADDCOL inserts TDIMn keywords into the header, and FXBOPEN interprets any found in the header—unless the /NO_TDIM keyword is used. Values of TDIMn can also be overridden with the DIMENSIONS keyword in the FXBREAD routine.

In addition to the keywords described in the binary tables extension proposal, several additional keywords are supported by FXBADDCOL. These keywords have a one-to-one correspondence with standard keywords used in primary FITS headers, i.e.

Additional Keyword	Standard Equivalent
TDMINn	DATAMIN
TDMAXn	DATAMAX
TDESCn	CTYPEm
TCUNIn	CUNITm[1]
TROTAn	CROTAm
TRPIXn	CRPIXm
TRVALn	CRVALm
TDELTn	CDELTm

The anticipated use of these keywords is such that TDMINn and TDMAXn would have a ordinary FITS record format, no different from their standard equivalents, and that the rest would have a format similar to TDIMn.

6 Variable-Length Array Facility

These routines also support the Variable-Length Array Facility described in C&T. Variable-length array columns are defined by using FXBADDCOL with the /VARIABLE keyword. Other than that, support for variable-length arrays is automatic. Some operations, such as reading entire columns, and the multidimensional array facility described above, are not allowed with variable-length arrays.

Ordinarily, the default THEAP value (NAXIS1 $\times$ NAXIS2) is used to write the variable-length arrays. However, a different THEAP value can be used by using FXADDPAR to insert the desired value into the binary table header before calling FXBCREATE.

[1]Note that this keyword is not strictly standard, but has been proposed by some to express the units of the axis of an array.

7 IEEE Not-a-Number (NaN) Special Values

Data dropout in FITS binary table arrays are signalled in one of two ways. Dropouts in integer arrays are signalled with values specified by TNULLn keywords. However, dropouts in floating point arrays (including single or double precision, and real or complex) are signalled with standard IEEE NaN (not-a-number) special values. The routine FXBREAD will optionally translate these NaN numbers into a user-specified value, given by the NANVALUE keyword. Conversely, the same keyword, when used with the FXWRITE or FXBWRITE routines, will write out NaN for any points in the array with the value of the NANVALUE keyword.

At present, there is no support for IEEE ±Infinity and -0 special values. However, it would be a simple matter to add support similar to IEEE NaN.

8 Bit, Logical, and Double-precision Complex Arrays

Although IDL does not support a data type corresponding to double-precision complex (type "M" in FITS binary tables), these routines do allow reading and writing this data type as ordinary double-precision arrays with an extra first dimension of two. Support for this is automatic when reading binary tables. Columns can be defined as type "M" when writing binary tables if the /DCOMPLEX keyword is passed to FXBADDCOL, and the data array is complex with the first dimension being of size two.

Bit arrays (type "X" in FITS binary tables) are treated in IDL as byte arrays with approximately 1/8th the number of elements. Support for this is automatic when reading binary tables. Columns can be defined as type "X" when writing binary tables if the BIT keyword is passed to FXBADDCOL giving the number of bits, and the data array is of type byte. Dimension information is ignored for bit arrays, since the dimensions apply to the bits, and not to the bytes that IDL processes.

Logical arrays (type "L" in FITS binary tables) are treated in IDL as byte arrays. Support for this is automatic when reading binary tables. Columns can be defined as type "L" when writing binary tables if the /LOGICAL keyword is passed to FXBADDCOL, and the data array is of type byte.

9 Virtual Columns

It is possible to treat keywords in binary table headers as if they were columns in the table, with the same value replicated for every row. This virtual column convention allows the user to have a unified view in a table regardless of whether the information is stored in a table column and thus capable of varying from row to row, or stored in the header and thus the same for every row.

To use the virtual column convention, the user must call FXBREAD with the /VIRTUAL keyword, and must also reference the desired information by name rather than by column number. FXBREAD will then look first for a column with that name. If it doesn't find one, it then looks for a keyword with that name in the header.

10 Associated routines

The remaining routines are mainly used internally by the other routines mentioned above.

FXHCLEAN: Remove obsolete keywords—called by FXHMAKE, FXHBMAKE.

FXPARPOS: Find position in FITS header—called by FXADDPAR.

FXBFINDLUN: Find LUN in FXBINTABLE—called by FXBCREATE, FXBOPEN.

FXBPARSE: Parse binary table header—called by FXBCREATE, FXBOPEN.

FXBTFORM: Parse TFORM column descriptor—called by FXBPARSE.

FXHREAD: Read FITS header—called by FXBOPEN.

FXFINDEND: Find the last FITS record—called by FXBCREATE.

WHERENAN: Find points equal to IEEE NaN—called by FXBREAD.

BOOST_ARRAY: Resize array, and append another array.

STORE_ARRAY: Resize array, and insert another array.

DETABIFY: Removes tabs from strings.

PRODUCT: Calculates total product of all elements of an array.

FXTPIO_READ: Reads a file from a tape—called from FXTAPEREAD.

FXTPIO_WRITE: Writes a file to a tape—called from FXTAPEWRITE.

11 Implementation notes

The routines in this directory also make use internally of other routines from the SDAS, FITS, and MISC directories from the Astronomy User's Library. In some cases these files are distributed along with the routines described here.

The file "fxbintable.pro" is an include file containing the definition of the IDL common block FXBINTABLE. This file must be in one of the directories pointed to by the IDL search path parameter !PATH. Normally, this is ensured by keeping this file in the same directory with the IDL procedures found here.