RFC: Mapping FITS data to HDF5

Pedro Vicente Nunes, pvn@ncsa.uiuc.edu

Rationale

> Just as cars come in many makes and models, scientific data is available in a wide variety of formats. These range from the "home grown", project-specific formats to the standard formats of the "Big 3" (FITS, HDF, netCDF). Unlike cars, where there is rarely a need to transform Toyotas into Chevys, it is often desirable to convert data from one format into another for archival, transport and analysis purposes
> (Jennings *et all, 1995*).

FITS is a data format mostly used within the astronomy community, while HDF5 is a general purpose library and file format for storing scientific data, used in a variety of communities. This RFC proposes a software tool, named fits2h5, that aims to read a FITS file and generate a HDF5 file with the corresponding data. Some of HDF5 features that potentially could be useful in the translation are

- HDF5 allows data to be organized into *groups*, analogous to UNIX file folders
- HDF5 provides random (fast) access to its data objects
- Data compression
- Chunking. Chunking refers to a storage layout where a dataset is partitioned into fixed-size multi-dimensional chunks. This has many efficiency advantages when doing I/O to the file, particularly in very large files.
- An API that defines a standard for arrays that are to be interpreted as images, with palette and true color (RGB) support
- The ability to define fill values in the data
- A variety of free and commercial visualization packages that read HDF5.

The goals that fits2h5 tries to achieve are thus, to facilitate the transport of space and Earth science data between different science disciplines, data analysis packages, and hardware platforms.

The FITS data format

FITS (Flexible Image Transport System) is a data format widely used within the astronomy community for transporting, analyzing, and archiving scientific data files. FITS has a support office page. FITS is primarily designed to store scientific data sets

consisting of multidimensional arrays (images) and 2-dimensional tables organized into rows and columns of information. Like HDF and HDF5, the underlying goal of FITS is to provide a standardized, simple, and extensible means to transport data between computers of different architectures.

A FITS file is comprised of segments called Header + Data Units (HDUs), where the first HDU is called the `Primary HDU', or `Primary Array'. The primary data array can contain a N dimensional array of 1, 2 or 4 byte integers or 4 or 8 byte floating point numbers using IEEE representations.
Any number of additional HDUs may follow the primary array. These additional HDUs are referred to as FITS `extensions'. Three types of standard extensions are currently defined:

- Image Extensions contain a N dimensional array of pixels, similar to a primary array

- ASCII Table Extensions store tabular information with all numeric information stored in ASCII formats. While ASCII tables are generally less efficient than binary tables, they can be made relatively human readable and can store numeric information with essentially arbitrary size and accuracy.

- Binary Table Extensions store tabular information in a binary representation. Each cell in the table can be an array but the dimensionality of the array must be constant within a column.

Header Units

Every HDU consists of an ASCII formatted `Header Unit' followed by an optional `Data Unit'. Each header or data unit is a multiple of 2880 bytes long. If necessary, the header or data unit is padded out to the required length with ASCII blanks or NULLs depending on the type of unit.
Each header unit contains a sequence of fixed-length 80-character keyword records which have the general form:

KEYNAME = value / comment string

The keyword names may be up to 8 characters long and can only contain uppercase letters A to Z, the digits 0 to 9, the hyphen, and the underscore character. The keyword name is (usually) followed by an equals sign and a space character in columns 9 and 10 of the record, followed by the value of the keyword which may be either an integer, a floating point number, a complex value (i.e., a pair of numbers), a character string (enclosed in single quotes), or a Boolean value (the letter T or F). Some keywords, (e.g.,

COMMENT and HISTORY) are not followed by an equals sign and in that case columns 9 - 80 of the record may contain any string of ASCII text.

Each header unit begins with a series of required keywords that specify the size and format of the following data unit. A 2-dimensional image primary array header, for example, begins with the following keywords:

SIMPLE = T / file conforms to FITS standard
BITPIX = 16 / number of bits per data pixel
NAXIS = 2 / number of data axes
NAXIS1 = 440 / length of data axis 1
NAXIS2 = 300 / length of data axis 2

The required keywords may be followed by other optional keywords to describe various aspects of the data, such as the date and time of the observation. COMMENT or HISTORY keywords are also frequently added to further document the contents of the data file.
The last keyword in the header is always the `END' keyword which has blank value and comment fields. The header is padded with additional blank records if necessary so that it is a multiple of 2880 bytes (equivalent to 36 80-byte keywords) long. Note that the header unit may only contain ASCII text characters ranging from hexadecimal 20 to 7E); non-printing ASCII characters such as tabs, carriage-returns, or line-feeds are not allowed anywhere within the header unit.

Data Units

The data unit, if present, immediately follows the last 2880-byte block in the header unit. Note that the data unit is not required, so some HDUs only contain the header unit. The image pixels in a primary array or an image extension may have one of 5 supported data types:

- 8-bit (unsigned) integer bytes
- 16-bit (signed) integers
- 32-bit (signed) integers
- 64-bit (signed) integers
- 32-bit single precision floating point real numbers
- 64-bit double precision floating point real numbers

Unsigned 16-bit and 32-bit integers are supported by subtracting an offset from the raw pixel values (e.g., 32768 (2**15) is subtracted from each unsigned 16-bit integer pixel value to shift the values into the range of a signed 16-bit integer) before writing them to the FITS file. This offset is then added to the pixels when reading the FITS image to restore the original values.

The other 2 types of standard extensions, ASCII tables and binary tables, contain tabular information organized into rows and columns.

All the entries within a given column of an ASCII or Binary table extension have the same datatype. The allowed data formats for an ASCII table column are: integer, single or double precision floating point value, or character string. Binary table extensions also support logical (T/F), bit, and complex data formats.

Each entry, or field, in an ASCII table may only contain 1 scalar value. Binary tables are more flexible and allow N-dimensional arrays of data (either fixed length or variable length) to be stored within each field. Variable-length arrays are implemented by storing a pointer in the field of the table which defines the length and byte offset to the start of the data array which is located in the "heap" area that follows the table proper. The variable-length-arrays convention is not strictly part of the FITS standard but is widely used.

Mapping FITS to HDF5

The FITS file is read in a linear fashion using the concept of a HDU position. This is a 1 based index that identifies the position of a HDU in the file. There is an API function that returns this index

```
fits_get_hdu_num(fptr, &hdupos); /* Get the current HDU position */
```

After opening the FITS file, this function returns 1, identifying the position of the primary HDU. A cycle can be constructed, where in each iteration there is an attempt to move to the next HDU position using the API function

```
fits_movrel_hdu(fptr, 1, NULL, &status);
```

which moves to the next HDU in the file. At each HDU, there is a query to find the type of HDU (image or table) using the API function

```
fits_get_hdu_type(fptr, &hdutype, &status);
```

which returns the HDU type.

Writing the HDF5 file

At each HDU iteration an HDF5 object is created according to

Type of HDU	HDF5 object

Image	Dataset
Table	Dataset with compound type, with the number of fields corresponding to the number of columns in the table
Header	Attribute of the HDU group (see below)

Image HDUs

Each of the five FITS Image data types are converted to HDF5 datatypes using the following table

FITS Image data type	**HDF5 dataset datatype**
8-bit (unsigned) integer	H5T_NATIVE_UCHAR
16-bit (signed) integer	H5T_NATIVE_SHORT
32-bit (signed) integer	H5T_NATIVE_INT
64-bit (signed) integer	H5T_NATIVE_LLONG
32-bit single precision floating point	H5T_NATIVE_FLOAT
64-bit double precision floating point	H5T_NATIVE_DOUBLE

Table HDUs

Allowed values for the data type in ASCII tables are: TSTRING, TSHORT, TLONG, TFLOAT, and TDOUBLE. Binary tables also support these types: TLOGICAL, TBIT, TBYTE, TCOMPLEX and TDBLCOMPLEX. The following table shows the corresponding mapping to HDF5 datatypes:

FITS Table data type	HDF5 datatype
TSTRING	H5T_C_S1
TSHORT	H5T_NATIVE_SHORT
TLONG	H5T_NATIVE_LONG
TFLOAT	H5T_NATIVE_FLOAT
TDOUBLE	H5T_NATIVE_DOUBLE
TLOGICAL	TBD (to be determined)
TBIT	TBD
TBYTE	TBD

TCOMPLEX	TBD
TDBLCOMPLEX	TBD

Naming HDF5 objects

This RFC proposes a way to the FITS data to be organized in the HDF5 file.

For each HDU, a group is created with the name "HDU_N", where the "_N" refers to the index of the object read (1 based).

HDU index	Object (group)
1	HDU_1
2	HDU_2
3	HDU_3

Each group would contain either a header and image or a header and table or just a header.

The FITS data model does not define a name for its objects, which are instead identified by its position in the file. When creating the corresponding HDF5 objects the following naming convention is used.

Type of HDU	HDF5 object name
Image	FITS_IMAGE_N
Table	FITS_TABLE_N

FITS Headers

A FITS header is created in the HDF5 file as an attribute of the group HDU. For each FITS HDU an attribute named 'FITS_HEADER_N' is created, as a compound datatype with 3 fields: the keyword name, the keyword value, and the keyword comment (all of which are optional in the FITS header). The fields are of HDF5 string datatype (H5T_C_S1). The number of elements in the this array (rows) is as many as FITS header keywords.

For the previous keyword example, one would have this FITS_HEADER_N attribute, the form on the FITS file as

SIMPLE = T / file conforms to FITS standard

translated to the HDF5 attribute as

keyword name	keyword value	keyword comment
SIMPLE	T	file conforms to FITS standard
BITPIX	16	number of bits per data pixel
NAXIS	2	number of data axes
NAXIS1	440	length of data axis 1
NAXIS2	300	length of data axis 2

Usage

The fits2h5 usage is

./fits2h5 <filename>

where <filename> is the name of an existing FITS file. The output of the program is an HDF5 file that has the same name as <filename> with the extension .h5. If the input file name has an extension (usually .fits or .fit), that is removed. For example

./fits2h5 file.fits

produces a HDF5 file named file.h5

References

Convert: Bridging the Scientific Data Format Chasm. (1995) D. G. Jennings, W. D. Pence. M. Folk. ASP Conference Series, Vol. 77, 1995.

CFITSIO User's Reference Guide, Version 2.5 (2004). HEASARC. Goddard Space Flight Center, Greenbelt, MD

Last updated

1/24/2006