*Astronomical Data Analysis Software and Systems VI*
*ASP Conference Series, Vol. 125, 1997*
*Gareth Hunt and H. E. Payne, eds.*

# FITS++: An Object-Oriented Set of C++ Classes to Support FITS

A. Farris[1]

*Space Telescope Science Institute, Baltimore, MD 21218*

**Abstract.** FITS++ is a object-oriented set of C++ classes, supporting both a sequential and random processing model, designed to be the basis of applications that manipulate FITS files. All aspects of the FITS standard are supported, allowing users to access any portions of FITS headers or data. Extensive error handling is also available.

## 1. Introduction

The purpose of this set of C++ classes is to give programmers the tools necessary to manipulate FITS files for a broad range of applications. Its primary purpose is to import and export data while adhering strictly to the FITS standards. All FITS data structures are supported, including random groups, image extensions, and binary tables. On input, errors are detected and reported but corrections are attempted. On output, all errors that might result in writing an invalid FITS file are detected.

Unlike the previous version (Farris 1995), these classes support two processing models, sequential and random, integrated into one coherent framework. The particular processing model, open mode (ReadOnly, WriteOnly, ReadWrite) and other file attributes, are selected when the FITS file is opened. The sequential processing model is useful for tape, pipes, and standard I/O. FITS header-data-units are read or written in sequential order.

The random processing model is restricted to devices that support lseek. Any FITS header-data-unit can be accessed randomly. Any keyword in the header or any portion of its associated data can be read or written. The random model allows the user considerable flexibility in manipulating random access FITS files. An append mode also allows users to add data to the end of a FITS file and update size information in the FITS header when the writing of the data is complete, all while maintaining the internal consistency of the FITS file. The random mode maintains a directory of header-data-units, much like an operating system maintains a directory of files. This directory is created when the file is opened.

A subset of processing commands may be used in both the sequential and random modes. Programs that only use this subset can be written to work with either mode. The only difference is the statement that opens the file. Commands that are extended beyond this subset apply only to the random

[1]E-mail: farris@stsci.edu

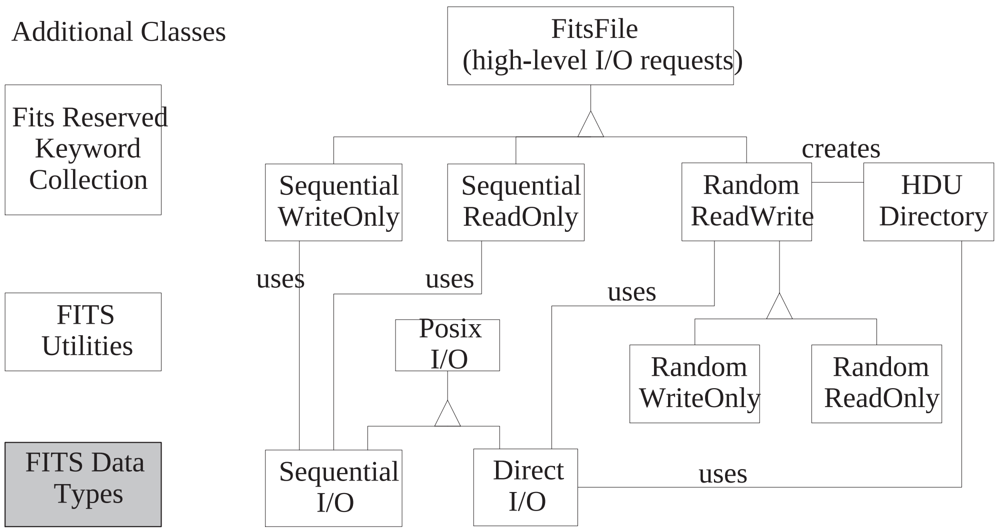


Figure 1. FitsFile Family of Classes.

processing model. Attempts to use these extended commands on a sequential file are flagged as errors.

All aspects of the FITS standard are supported, including image extensions, random groups structure, general extensions, ASCII and binary tables. The variable length array facility and the multidimensional array convention, used in conjunction with binary tables, are also supported.

FITS++ is implemented in C++ and will compile on most available C++ compilers. Language features recently added by the ANSI standardization committee and only available on newer compilers are avoided. GNU's C++ compiler, g++, running on SUN workstations is the default development environment. FITS++ runs on UNIX, VMS, and Windows-95/NT operating systems. Under VMS, DEC's C++ compiler is used. Both DEC VAX and Alpha platforms are supported and all VMS floating-point options (IEEE_float, G_float, and D_float) are supported.

## 2. Basic Design Considerations

Within FITS++, a FITS header-data-unit (HDU) can be formed in two ways: from an existing file, or from a FITS keyword list. In the first way, the HDU is initially associated with a file. In the second way, it is not. Regardless of how the HDU is created, once it is created it is an independent object in its own right. It may be modified or written to other FITS files. The header and data are read (or written) by separate operations.

On input (or output), all of the HDU's data does not have to be read (or written) at once. For output purposes, an HDU may be associated with any file. The HDU classes contain the knowledge of the internal structure of the data, as described by the FITS keywords. Their job is to provide access to the descriptive keywords and to the data themselves in the form of operations that

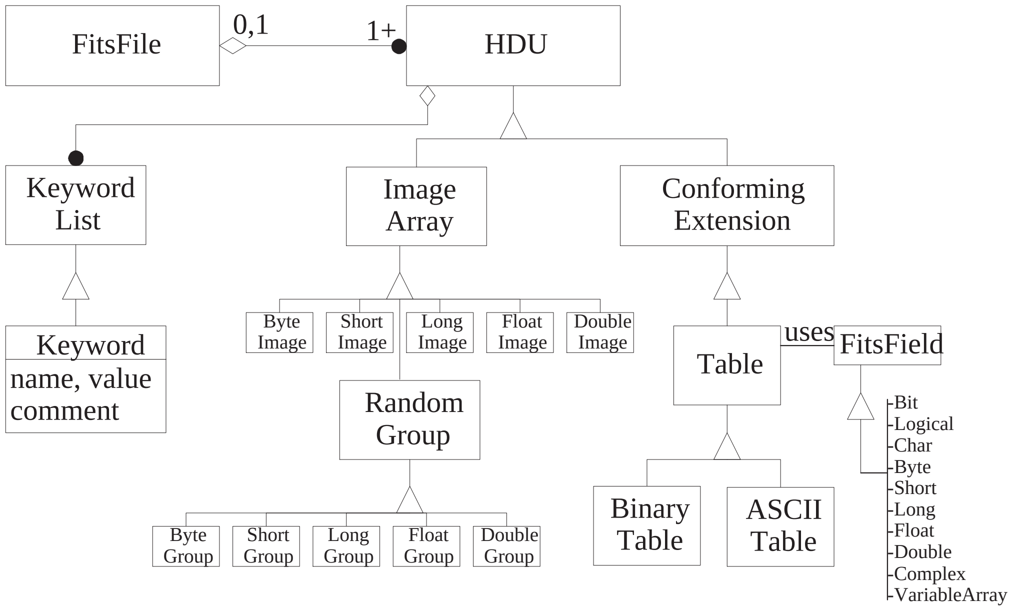


Figure 2. FITS HDU Family of Classes.

are meaningful to a data structure of that particular type. They know nothing about how this kind of HDU is formatted in a FITS file.

The FitsFile classes, on the other hand, know nothing about the internal structure of HDU data; they do know everything about how those HDUs are structured within FITS files. Furthermore, they know how to translate FITS headers into keyword lists and vice versa. System level I/O commands are encapsulated in a class that implements Posix I/O functions. This class is designed to be operating system independent, and runs under both UNIX and VMS.

Figure 1 shows the FitsFile family of classes. These classes perform basic I/O functions. The "open" function creates a FitsFile object. Destroying the object flushes buffers and closes the file. On input, high-level I/O commands allow one to select a particular HDU, determine its type, and create an HDU object, if desired. Generic or specific HDUs may be created. An ImageHDU may be created if one only wishes to access physical data and disregard the raw data type. Likewise, if one wishes to access a table, and ignore whether it is an ASCII table or binary table, one may do so. Low-level I/O commands provide mechanisms for accessing and modifying any portion of a FITS file with no restrictions. These can be used to cope with unusual situations.

Figure 2 shows the HDU family of classes. These classes implement the basic FITS data structures. All FITS data structures are supported, including full support for random groups and all standard extensions. An ImageHDU is automatically converted to either a primary HDU or an image extension, as appropriate. Image arrays of arbitrary dimensionality are supported, but are optimized for 1, 2, and 3 dimensional arrays. One can access either physical data (after applying BSCALE and BZERO) or raw data.

Tables are read by rows. Access to fields within a row is similar to programming interfaces to relational DBMS tables. The multi-dimensional and variable length array conventions, used in conjunction with binary tables, are fully supported. The append mode is particularly useful in creating tables. Rows of a table can be added in the append mode. When this mode is closed, the number of rows in the FITS header is automatically updated. Because the append mode maintains an externally consistent view of the FITS file, a separate process can read such a file as it is being updated. To such a process, the appended records appear to be FITS "special" records.

## 3. FITS Keyword Handling and Error Handling

Within these classes, FITS keywords are implemented using an encapsulated linked list. Uses of a keyword depend only on the keyword name, value, and comment. Detailed FITS formatting rules are invisible to the user. A FITS keyword-to-card translator is built into the FitsKeyword classes. They also use a table of properties of reserved FITS keywords. The list of keywords can be manipulated in a wide variety of ways. A keyword's name, index (if any), value, or comment may be changed. On input, an exact copy of the FITS keyword string is retained, and the order of keywords is maintained. On output, the user has the option of formatting the keyword string rather than relying on built-in formatting functions.

Errors are arranged in order of increasing severity and divided into: warnings (technically not errors, but are undesirable), minor errors (correctable, or not resulting in data loss), serious errors (conditions that will result in loss of data), severe errors (usually halting processing, such as I/O errors), and fatal errors (halting processing). These classes use a flexible method of error handling to give the user control over coping with error conditions. Global error conditions and error messages can always be accessed via global functions. In addition, the user can install an error handling function that is called if an error condition is raised. This function can be applied to global conditions or can be installed to catch errors pertaining to a particular FITS file.

## 4. Current Status

FITS++ is being used internally at STScI in the pipeline calibration and archive processing software for STIS and NICMOS, both of which are currently under development.

## References

Farris, A. 1995, BAAS, 27, 909

FITS 1993, "Definition of the Flexible Image Transport System (FITS)," Standard, NOST 100-1.0, NASA / Science Office of Standards and Technology, Code 633.2, NASA Goddard Space Flight Center, Greenbelt, Maryland

Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363