ADF

NASA/GSFC Astrophysics Data Facility

A User's Guide for the Flexible Image Transport System (FITS)

Version 4.0

April 14, 1997

NASA/GSFC Astrophysics Data Facility
Code 631
NASA Goddard Space Flight Center
Greenbelt MD 20771
USA

Preface

Since version 3.1 of *A User's Guide for the Flexible Image Transport System (FITS)* was published, there have been several significant developments in the FITS community:

- The International Astronomical Union FITS Working Group (IAUFWG) has endorsed the image (`IMAGE`) and binary table (`BINTABLE`) extensions and the agreement on physical blocking.
- The proposal for treatment of world coordinate systems has been expanded and refined.
- A number of conventions that are not part of the formal FITS rules have come into wide use.
- A significant body of FITS resources has become available on the World Wide Web.

This new version of the *User's Guide* has been written to reflect those changes. The discussion of the image extension and the rules for the binary table extension have been moved from section 5 (Advanced FITS) to section 3 (FITS Fundamentals). The papers describing the image and binary table extensions have now been published in the *Astronomy and Astrophysics Supplement Series* and are now considered to be among the fundamental FITS papers. Section 4 on World Coordinates has been updated to include the refinements incorporated in the current proposals, and it also discusses in greater detail how widely the different proposed conventions are used in the general community. The discussion of the three proposed binary table conventions, which are not part of the formal structure endorsed by the IAUFWG, remains in section 5. The discussion of applications of binary tables has been significantly expanded, and descriptions of a number of binary table and other conventions have been added. Section 6 has been rewritten to emphasize network sources of FITS information, and it discusses a number of sites on the World Wide Web that contain FITS documents, software, and sample files. Many of these sites did not exist when version 3.1 of the *User's Guide* was written, and others have been greatly expanded since. Three additional sample FITS headers, one using an ASCII table and two using binary tables, have been added to Appendix A. Appendix B lists the IEEE floating point number type corresponding to every bit pattern.

Thanks for comments on an earlier draft of version 4 go to M. Calabretta, D. Jennings, W. Pence, and R. Thompson. Also, thanks to D. Leisawitz for providing the DIRBE FITS header (example 6) and W. Pence for providing the ASCA FITS header (example 7). To be of most use to readers, a guide such as this one must go beyond the formal rules to discuss common practices that are not specified by those rules. In addition, users who are designing and developing FITS files need to know how FITS is likely to develop. Developing a formal standard for FITS has clarified a number of points that had been unclear or ambiguous in the original FITS papers. Some of the issues were regarded as not appropriate for a formal standard but deserving of further detailed discussion; at the recommendation of the Technical Panel developing the NASA/Science Office of Standards and Technology (NOST) standard, they are included in this *Guide*. Queries to the FITS Support Office and discussion on the FITS network newsgroup `sci.astro.fits` and the associated `fitsbits` electronic mail exploder have identified other points in need of explanation.

This *Guide* also describes a number of conventions that are widely used but have not been formally adopted by the FITS governing structure under the IAUFWG. Description in this *Guide* of such conventions is intended neither as a NASA endorsement nor as a requirement for use of these conventions by NASA projects. Where an issue is controversial, this *Guide* attempts to provide the arguments on all sides.

FITS is continually expanding: new conventions are proposed, existing proposals are modified, new issues are raised; and the FITS committees act. Some of this progress will occur during the period this Guide is being proofed and undergoing physical composition. Thus, some of these developments may, unfortunately, not make it into the current *User's Guide*. To keep up with current events, use the resources described in Section 6. Similarly, Web sites may be reorganized and the URLs corresponding to an individual page may change. In that case, the new location can generally be found by going to the main page for the site and following appropriate links.

As always, comments about the *Guide*, in particular about areas that need clarification or expansion, are encouraged. Send questions or comments to

FITS Support Office
Astrophysics Data Facility
Code 631
Goddard Space Flight Center
Greenbelt MD 20771
USA

Telephone: +1-301-286-2899

Electronic Mail: fits@nssdca.gsfc.nasa.gov

Contents

Appendixes

List of Tables

Section 1

The Origin and Purpose of FITS

1.1 The Need for FITS

In the late 1970s, the Westerbork Synthesis Radio Telescope (WSRT) in Westerbork, Holland and the Very Large Array (VLA) in New Mexico began producing high quality images of the radio sky. Since the two groups were observing at different frequencies, they wished to collaborate in constructing spectral index maps by combining data obtained from the two instruments. It was clear from the outset that this process would not be trivial. Two different institutions would normally structure their data in different ways. Machines at the two different installations might have different architecture, using a different internal representation for the same number. Lacking a standard format for transporting images, an astronomer taking data from an observatory to a home institution would have to create special software to convert the data to the format used at the home institution. Such software would restructure the data to the format of the home institution and perform whatever bit manipulations were necessary to convert numbers from the representation in the originating machine to that for the destination machine.

While radio astronomers were making great strides towards producing analog images of their normally digital data, optical astronomers were making great strides toward producing high quality digital data through the use of charge coupled devices (CCDs). Here too, transport of data in one wavelength domain for comparison with data in another wavelength domain was required. Because each installation had its own internal storage format, new software was needed whenever two installations exchanged data for the first time.

An obvious substitute for all these cumbersome processes was the creation of a single standard interchange format for transporting digital images among cooperating institutions. Then, each institution would need only two software packages: one to translate the transfer format to the internal format used by the institution, and one to transform the internal format to the transfer format. The Flexible Image Transport System (FITS) was created to provide such a transfer format.

From its initial applications—exchange of radio astronomy images between Westerbork and the VLA and exchange of optical image data among Kitt Peak, the VLA and Westerbork—the use of FITS has expanded to include the entire spectrum of astronomical data. It is being used for a variety of data structures from past, current and future NASA-supported projects, for example, X-ray data from the Einstein High Energy Astrophysics Observatory (HEAO-2), the Compton Gamma Ray Observatory, and the Röntgen Satellite (ROSAT) where NASA is cooperating with Germany and the United Kingdom; ultraviolet and visible from the International Ultraviolet Explorer (IUE) and the Hubble Space Telescope; and infrared data from the Infrared Astronomical Satellite (IRAS) and the Cosmic Background Explorer (COBE). It is also the standard for ground-based radio and optical observations, in use by such organizations as the National Radio Astronomy Observatory (NRAO), National Optical Astronomy Observatories (NOAO), and the European Southern Observatory (ESO).

1.2 What FITS Is

The fundamental unit of FITS is the FITS file, which is composed of a sequence of Header Data Units (HDUs), optionally followed by a set of special records. (The rather prosaic name *HDU* was the result of over a year's community discussion and failure to find anything better.) The first part of each HDU is the header, composed of ASCII card images containing keyword=value statements that describe the size, format, and structure of the data that follow. It may contain any information about the data set that its creator regards as important, such as information about the history of the data or the file, about the physical entity the data describe, or about the instrument used to gather the data. The data follow, structured as the header specifies. The size of logical records in both header and data is 23040 bits, equivalent to 2880 8-bit bytes or 36 header card images. The HDUs may be followed by special records; the only restrictions on these special records are that they have the standard 23040-bit logical record size and that they not begin with the string `XTENSION`. A FITS file is terminated by a logical end of file, whose precise physical nature will depend on the medium.

In its original form, a FITS file consisted solely of a single HDU, consisting of the header and a data array that was regarded as containing a digital image. This simple, one-HDU structure is known as Basic FITS. The header card images would describe the data array—the number and length of the axes and the data type of the values: unsigned one-byte, signed two-byte or four-byte integers. The original use of FITS to transport digital images is reflected in the "Image" in its name. However, even the data matrix of Basic FITS could be used to transmit any kind of multidimensional array, not simply an image. The first HDU of a FITS file, called the primary HDU, must still follow the Basic FITS format, although it need not contain any data.

FITS is no longer restricted to integer arrays.The array data may be Institute of Electrical and Electronics Engineers (IEEE) 32-bit or 64-bit floating point. The Basic FITS primary HDU may be followed other HDUs, called extensions, containing different data structures. Standard data formats include two kinds of tables: tables with ASCII entries and tables with binary entries, as well a multidimensional array extension format that allows extensions to contain the same type of data that is in the primary HDU. It is also possible to create non-standard formats, for use locally or as prototype designs for new standard formats.

Although its name implies "image" transport, FITS is not a graphics format designed simply for the transfer of pictures;it does not incorporate "FITS viewers", packages for decoding the data into an image. Users must develop or obtain separate software to read and display the data from the FITS file. Because of its wide use, FITS is supported by all the major astronomical imaging packages, and a number of other packages of FITS utilities and software are publicly available. The data structure is an essential part of the format and is available to the users. This property distinguishes FITS from many other data standards—those that are primarily labeling systems, and those for which the user accesses a hidden data structure through a set of standard tools.

1.3 The Philosophy of FITS

FITS incorporates a philosophy along with the data format. The underlying goal is to provide a standardized, simple, and extensible means to transport data between computers or image processing systems. Any FITS reader should be able to cope with any FITS formatted file, skipping over portions (extensions) and ignoring keywords that the reader does not and need not understand.

Simplicity requires that reading and writing FITS should be implemented in a fairly straightforward way on any computer used for astronomical reduction and analysis. Simplicity also implies that the structure of the file should be self-defining and, to a large degree, self-documenting.

The first word in FITS is "flexible". The format needs to be flexible to facilitate extensibility for different applications. Hence, the number of strict rules is not large. Because the files are self-defining, FITS can fulfill a large range of data transport needs. FITS can be used not only for unambiguous transportation of n-dimensional, regularly-spaced data arrays, but also for additional information associated with such a matrix. FITS can also transport arbitrary amounts of text within standard data files. The "history" of manipulations of the data can thus easily be recorded in self-documenting data files. FITS is sufficiently general for a wide variety of applications. The introduction of new keywords permits addition of new information as needed, and the use of extensions allows almost unlimited flexibility in the type of information to be stored. Thus, FITS can grow with the needs of the astronomical community.

The great flexibility of FITS is a potential weakness as well as a strength. While there is a great temptation to proliferate keywords and new extension types, caution should be exercised in this process. Because FITS is a worldwide medium of data exchange, extension formats need to be coordinated under the International Astronomical Union FITS Working Group (IAUFWG) to prevent duplication and inconsistencies in usage, and agreements should be reached governing keyword conventions in particular fields. The structure under the IAUFWG provides an overall authority for the FITS standard, but additions to FITS are not created by the IAUFWG but are designed by FITS users and then acted upon by the international structure. Although the number of strict rules is not large, there is an extensive set of recommended practices. Creators of FITS files should adhere to these recommendations if at all possible; in particular, the rules of FITS should not be exploited to create files that try to mimic the local format, and, although in technical compliance with the rules, depart from the recommendations to such an extent that they don't look like FITS files. General adherence to recommended practice will simplify the task of the FITS software developers; if a FITS file contains too many unconventional but permitted constructs, many FITS readers may not be able to handle it. Not everything that is permitted is wise.

Users who develop extensive libraries of FITS files need assurance that they will not have to periodically revise these files because of changes in the standard. This requirement gives rise to one of the fundamental principles of FITS: no change in the rules should render old FITS files unreadable or out of conformance—the principle of "once FITS, always FITS." This philosophy is reflected in data reduction and analysis packages in which all obsolete implementations are trapped and processed in the most accurate manner possible. While adherence to this principle has perpetuated some constructs that have proven with time to be awkward, it is better than the alternative of requiring revision of existing FITS files.

Changes in the FITS rules may add new structures that old software cannot handle. Revised software will be required for new standard extensions, but revising a software package is a far smaller effort than updating a full data library would be. As far as is possible, however, FITS should be expanded in such a way that the old software will still be able to process those parts of the file which it is capable of handling. In such a case, software should not fail or give incorrect results when confronted with the new extension or conventions; it should simply ignore them and continue to process those parts of the file that it can understand.

FITS is defined as a logical structure, not tied to the properties of any particular medium, thus allowing its continued use as the technology changes. Conventions for its adaptation to any medium are independent of the logical structure of FITS. Because its original development was for 1/2-inch magnetic tape, its structure is well adapted to that medium and the conventions are long established. More recently, more generalized conventions have been adopted for the expression of FITS on magnetic tape and on magnetic and optical disk. Conventions can be defined for new media as they develop.

Section 2

History

2.1 The First Agreement

In November 1976, R. Harten of the Netherlands Foundation for Radio Astronomy (NFRA) and D. Wells of the Kitt Peak National Observatory (KPNO), now part of NOAO, began work on a solution to the problem of exchanging data between observatories. In the spring of 1977, Harten and Wells each built prototype data interchange programs, and they exchanged tapes. During 1977 and 1978, J. Dickel (University of Illinois) carried radio and optical imagery encoded in the two formats between Westerbork and Kitt Peak.

In January 1979, during the National Science Foundation (NSF) Image Analysis meeting at KPNO, the exchange format problem was discussed. P. Boyce (NSF), the chair of the meeting, urged NOAO and NRAO to come up with a solution and assigned a task force of R. Burns (NRAO), E. Groth (Princeton), and Wells to begin the process. Burns organized a meeting at the VLA of this task force and the VLA programmers. On direction from the initial session of the meeting, E. Greisen (NRAO) and Wells retired to the VLA conference room in the cafeteria building and worked together to produce the Basic FITS Agreement in 36 hours on March 27/28, 1979. It incorporated the lessons learned from the Harten and Wells prototypes. One key issue during the development process was the choice of a logical record size that would be both convenient for all machines in use at the time and would at the same time be close to filling a physical block on magnetic tape. The adopted 23040-bit logical record was an integral multiple of the word sizes of all machines then in use and was close to the 30240-bit physical block size limit of the CDC 6000/7000 series tapes. One data structure was supported: an unsigned 8-bit, signed 16-bit, or signed 32-bit integer array with 0–999 axes. However, to permit future growth, additional records were permitted to follow the data array, as long as the logical record size was the 23040-bit standard, a provision that served as the basis for the later development of extensions.

In May 1979, NOAO and NRAO exchanged FITS tapes, thereby showing that the rules of the Basic Agreement would operate in practice. The first FITS tape was written by a PL/I program executing on an IBM-360 under OS/MVT (32-bits twos complement and EBCDIC) and was successfully read by a FORTRAN program executing on a CDC-6400 under SCOPE (60-bits ones complement and Display Code), very nearly the worst possible combination of environments for interchange. This insistence on testing the format before it was presented set a precedent for future development: a practical demonstration of transfer using a proposed FITS structure is still required before it can be approved.

On June 8, 1979, Basic FITS was presented at an International Image Processing Workshop in Trieste, Italy (Wells and Greisen 1979). Harten endorsed it, it won immediate acceptance; and within one year FITS became the worldwide *de facto* standard in astronomy. Wells, Greisen, and Harten (1981; hereafter FITS Paper I) published the description.

2.2 Random Groups

While FITS began as a means of transporting simple digitized images from machine to machine, it was soon realized that FITS could provide a framework for transporting other types of data. The first new FITS structure, designed by Greisen and Harten during late 1979 – early 1980 (Greisen and Harten 1981; hereafter FITS Paper II), was composed of a set of "random groups", each consisting of a sequence of parameters followed by a small array of data. The number and meaning of the parameters and the dimensions of the array would be the same for all groups. In some of the early literature, this structure is described as an "extension", but such terminology is now inappropriate, as the name "extension" refers to the structure described in sections 2.3 and 3.3. The principal application of this format was to radio astronomical aperture synthesis visibility data. These data consist of small groups of arrays that occur in a relatively random manner on one or more axes.

Random groups has failed to attain wide use in other areas and is now being replaced even for aperture synthesis data by binary tables. Future use is discouraged.

At the 1982 General Assembly, the IAU endorsed FITS, including the Random Groups format, as the recommended format for transport of binary data (IAU 1983).

2.3 Generalized Extensions

The special records provision of the original FITS design made it possible to extend the format. There were two principal reasons for wanting to extend FITS beyond the simple header/array structure:

- to transfer new types of data structures while adhering to the basic rules of FITS.
- to transfer collections of related data structures, thereby creating a simple hierarchical data base capability.

For example, the table extensions have allowed tables, lists, etc., associated with a data matrix to be written in the same FITS file as the data matrix, implicitly establishing the relationship among the different pieces of information.

The method chosen was to define "extensions", HDUs, which, like the primary HDU, are composed of a header consisting of card images in ASCII text with keyword=value syntax, followed by data. There could be many kinds of extensions, each with a different defined data format. Structuring extensions in this way made it easy to modify software that read the FITS header for the primary array to read extension headers as well. Information about the extension data would appear in the extension header in a way specified by the rules for that extension. All logical records would be 23040 bits (=2880 8-bit bytes), as the original paper describing FITS prescribed for information following the primary HDU. The HDU itself is called an extension. The design is called an extension type.

The requirement that no revision to FITS could cause an existing FITS file to go out of conformance dictated a number of the basic rules governing the construction of new extensions.

In the original FITS data sets, the Basic FITS structure of header and array appeared at the start of the file. Therefore, extensions would appear only after the primary Basic FITS header and array. Because the initial array ended only at the end of a 23040-bit record, an extension would always start a new record.

It was envisioned that most FITS extensions would become standard in the same way as Basic FITS, through acceptance by the astronomical community and endorsement by the IAU. An extension would go through a development period, initially being used only by a subset of the FITS community that would refine it. Other extensions might be in use only within a limited group and might never become standard at all. Now, a FITS file may include many extensions of different types. Given that the order of adoption of extensions as standard cannot be predicted with certainty, it would not have been wise to prescribe an order of extension types within a file. Suppose standard extensions were required to appear first, and the fourth extension on a data set were to become standard. Then, the data set would go out of conformance. It was thus agreed that extensions might appear in any order in a FITS file.

With extensions appearing in any order, those extensions a user might want to or be able to read could be separated by extensions with which the user would be unfamiliar; the user might wish to read, say, only the third and seventh and skip all the rest. The user should not have to know anything about the structure of the intervening extensions to be able to read the ones of interest. To make this process possible, two general rules were specified:

- Each type of extension must have a unique name, provided in the header.
- The header must provide information that will allow the software reading it to calculate its size.

The software reading the FITS file would have a list of types of extensions that it could handle. By reading the type name from a standard location in the header, the software would be able to determine whether or not it could handle this extension. If it couldn't, it could at least calculate how many records it would have to skip to reach the beginning of the next extension.

A complete set of rules, described as the Generalized Extensions agreement (Grosbøl *et al.* 1988; hereafter FITS Paper III), was endorsed by the IAU in 1988. These rules appear in section 3.3.

2.4 ASCII Tables

The concept of a standard flexible format for the transfer of astronomical data was so appealing that astronomical software designers sought to apply the format to data and information structures other than simple arrays. For example, astronomers make extensive use of catalogs. Such information would most naturally be stored as a table. The wide variety of tabular information led to the development of the ASCII table extension. The following three main classes of potential applications were envisioned:

1. Standard catalogs such as star or source catalogs.

2. Observing information such as observing logs, calibration tables, and intermediate tables related to the observing. The results of the observations might appear as the Basic FITS matrix, and the auxiliary information would follow in a table.

3. Tabular results extracted from observational data by data analysis software. As an example, many programs automatically detect sources in digitized images and write parameters such as position, flux, size, spectral index, and polarization into output files. Astronomers need to transmit these output tabular files; recipients can then use software designed to manipulate, merge, and intercompare these tables.

The ASCII table FITS extension (Harten *et al.* 1988; hereafter FITS Paper IV) conforms to the standard FITS rules and to the generalized rules for FITS extensions. The column headings are provided in an extension header that describes the contents of the table. The table data are stored as a large character array. Each row of the table consists of a sequence of fields. Each field is described by a series of keywords specifying the field format using FORTRAN-77 notation, the location in the row where it begins, and possibly a column heading or other information about the field.

2.5 Floating Point

In the original FITS format, the members of the data array were required to be of integer data type. Non-integral values, or values outside the range that could be expressed as integers, were stored through the use of scaling. The actual physical values would be derived from the numbers in the FITS file by a linear transformation. The coefficients for this transformation were stored in the header as values of keywords. This method of using integers to represent floating point values has a number of limitations. Only a limited dynamic range of values can be stored with precision using scaled integers. Many astronomical data systems use floating point in their internal data processing; conversion to integers consumes significant amounts of computer time.

The adoption by the IEEE of a standard format for floating point numbers and its widespread implementation by many computer systems provided the solution. On December 22, 1989, the IAU FITS Working Group adopted the Floating Point Agreement, establishing IEEE-754 (IEEE 1985) 32-bit and 64-bit numbers as the standard FITS floating point data types, effective January 1, 1990.

2.6 Physical Blocking

In 1979, when FITS was originally developed, the dominant medium for data storage and transport was 1/2-inch nine-track magnetic tape. In FITS Paper I, the physical block size was set equal to the logical record size. As time passed, it became clear that many of the major data producers regarded this block size as inefficient, in terms of both tape length used and the number of I/O operations required to write data. The new generation of computers, with Megabyte size memory, could easily read much larger blocks. As a consequence, FITS Paper III included a provision that there could be up to 10 logical records per physical block on 1/2-inch nine-track magnetic tape. New storage media, such as cartridge tapes and optical disks were replacing magnetic tape. Many of the new media could access data only in blocks of fixed length, typically 2^n bytes, and the FITS 23040-bit logical record length would not correspond to an integral number of these blocks. Whereas FITS had been discussed in FITS Paper I in the context of files on magnetic tape, the increasing use of electronic transport for files was leading to the concept of a FITS file as a pure bit stream, without special ties to any particular medium. However, a set of prescriptions for the physical expression of FITS files on different media was still needed. General rules for all media, and in particular for how to write FITS logical records to the 2^n-byte physical blocks, were proposed by Wells and P. Grosbøl (ESO) in 1991. With minor changes, they were approved by the IAUFWG in the spring of 1994. They appear in section 3.8 of this *Guide*.

2.7 Image Extension

In the late 1980s, there had been some discussion in the FITS community about providing a mechanism for including multidimensional arrays in extensions as well as in the primary data array. The IUE group was looking for a way to include several related arrays in the same file, in particular, both their data and a flag array in which the flag for each element would refer to the data for that element. Because the data types for the flag and the data were different, they could not just add another axis and include the flag data in the primary data array. J. D. Ponz and J. R. Muñoz of the ESA IUE group and R. Thompson (CSC) of the GSFC IUE group made a detailed draft available electronically early in 1992. The extension was given the name `IMAGE`. The only significant discussion was whether or not to provide for Random Groups records after an image extension; because Random Groups have largely fallen into disuse and are not recommended for future use, the decision was not to allow Random Groups records.

2.8 Binary Tables

One drawback of the ASCII table format was the space required by tables with a large number of entries, such as gain and calibration tables, because every number appeared in coded form. In addition, conversion of the entries in these tables to ASCII would be very time-consuming. Initially, the ASCII form had been needed to represent floating point numbers. The precise definition and general acceptance of the IEEE floating point format made it possible to include binary floating point numbers in table fields. Meanwhile, developers of software to write FITS files for results of the Very Long Baseline Array project found that it would be useful to be able to put vectors in table fields. A design for binary tables with vector fields was developed by W. Cotton (NRAO). Because the vector field might be thought of as a column rising out of the page, it was imagined to be a third dimension of the table, and the extension was given the type name `A3DTABLE`. This name was chosen rather than, for example, `3DTABLE`, to signify that the extension might still undergo further development; the final version would have a new name. This 3-D table extension was released as part of the NRAO Astronomical Image Processing System (AIPS) early in 1987.

At the beginning of 1990, NASA Headquarters specified that all NASA or NASA-supported projects must make their data available in FITS format. Many of the projects then in the process of developing their data plans were high energy astrophysics experiments. Their data would usually appear as event lists. These event tables could be very large. Map arrays would contain mostly zeroes and be inefficient. Binary tables would be the best format. Many of the FITS files designed by the high energy astrophysics community were based on the AIPS 3D table concept. This growing demand gave impetus to the development of a formal proposal for binary tables. Cotton distributed an initial proposal for a standard binary table extension, to be called `BINTABLE`, in April 1991. It incorporated all the features of the original `A3DTABLE` extension and included a number of additional keywords and field formats that had been suggested by readers of the original `A3DTABLE` description.

In late 1989, at a meeting at Green Bank to develop a standard format for single dish data, Wells had suggested the concept of including multidimensional matrices, not just vectors, in the fields of binary tables. The participants in the meeting had decided to adopt such a format and proceeded to discuss possible designs. Subsequently, a number of prospective binary table users expressed an interest in a mechanism for including arrays whose dimension might vary from row to row in the table. This issue was raised by D. Tody at the April, 1991 European FITS Committee meeting. Following discussions outside the formal sessions by a number of the participants, Cotton and Tody presented a design with a pointer data type, which was modified slightly in the discussion at the meeting. A formal text with appendixes describing possible formats for multidimensional fields and variable length arrays, and incorporating the keywords and field formats to make these structures possible, was released by Cotton and Tody in October, 1991.

As early as July, 1991, W. Pence (GSFC/HEASARC) had raised questions about how an array of strings was to be distinguished from a single long string. The ensuing electronic discussion led to the addition of a third appendix, describing a substring array convention. The revised `BINTABLE` proposal with this appendix added was released by Cotton, Tody, and Pence in May, 1993. In the spring of 1994, after a few points had been clarified, the IAUFWG endorsed the main proposal as part of standard FITS. The three appendixes – multidimensional arrays, variable length arrays, and arrays of strings – were not included as part of the endorsed standard; they are considered recommended but not required conventions. However, they have a special place in that the `BINTABLE` extension contains a number of provisions especially designed to make them possible.

Initial testing began in 1992, with an exchange of `IMAGE` and `BINTABLE` files among ESO, IUE, and the High Energy Science Archive Research Center (HEASARC) at Goddard Space Flight Center. In line with the evolution of FITS from a tape standard to a bit stream standard, the exchange was carried out by making files available through anonymous ftp rather than through exchange of tapes. Early in 1994, following the revision of `BINTABLE`, additional tests were carried out in preparation for the votes by the FITS committees. Files from the Space Telescope Science Institute (STScI) containing `IMAGE` and `BINTABLE` extensions were read at ESO, and another set of files from ESO was read at GSFC/HEASARC. The successful tests allowed the two extensions to proceed to a vote. On June 15, 1994, Chair P. Grosbøl announced that the IAUFWG had endorsed the blocking rules and the `IMAGE` and `BINTABLE` extensions. As both extensions are now standard FITS, they are now described in Section 3, rather than in Section 5 as was done in earlier versions of the Guide. The discussion of the `BINTABLE` appendixes remains in Section 5, as they were not part of the proposal adopted by the IAUFWG.

2.9 How FITS Evolves

The history of binary tables illustrates many of the aspects of the way in which additions to the FITS format have developed in the past and are likely to continue to evolve in the future. The initial concept for a structure typically arises from the designer of a data set who finds that none of the existing standard or proposed FITS formats will organize the data properly. A proposal is developed and a name chosen for the new extension type. This name must then be registered with the IAUFWG. It must be different from any name previously registered. The FITS Support Office maintains a list of the registered extension type names and will forward name registration requests to the IAUFWG. In some cases, a temporary local form of the extension with a different name will be be defined as well. This local name must also be registered with the IAUFWG. The local or developmental form may serve as the basis for designing data sets elsewhere while the full proposal is being developed, as occurred with `A3DTABLE`. During this period, the developer will discuss concepts of this new extension with others interested in the format and may modify it based on this discussion.

Eventually, a formal proposal will be made available for review by the astronomical community, normally by announcing World Wide Web (WWW) and ftp locations where it can be obtained. With the more general use of FITS and the increasing ease of electronic transmission of documents, proposals are becoming available to a wider public at earlier stages than was the case during the early days of FITS. Tests are run using the new format to confirm that it can be practically used for data transport. After the community has reviewed the proposal, any modifications have been made, and the format has been successfully used for data transport, the proposal is submitted for approval to the regional committees—the European FITS Committee, the Japanese FITS Committee, and the American Astronomical Society Working Group on Astronomical Software (WGAS) FITS Committee. Following approval by the regional committees, it is submitted to the IAUFWG. Approval by the Working Group establishes it as a standard extension.

A number of aspects of this process are worth noting. First, FITS formats have been developed through the efforts of individuals responding to particular practical problems. Although there is extensive community review, most of the initial detailed development is done by the people who actually need to use a format for a particular purpose, rather than by a formal committee. A new format must be demonstrated to work in practice through actual data transport before it can be approved as standard, ensuring that the standard is not purely theoretical but has actually been used. Finally, the approval process occurs by community consensus, as is customary in the world of standards.

Section 3

FITS Fundamentals

3.1 Basic FITS

The fundamental unit of a FITS data set is the file, which begins with the ASCII string `SIMPLE  =                    T`, where the first 6 bytes of the file contain `SIMPLE`, the "=" is in byte 9, the `T` is in byte 30, and the intervening bytes contain ASCII blanks. This 30-character string is the signature of FITS, the way in which generalized software can identify the file as FITS. The string is not to be used if the file deviates from the rules for FITS in any way. In such a case, a value of `F` may be appropriate. A FITS file ends with an end-of-file mark appropriate to the medium on which it is written. A FITS file is composed of 23040-bit (2880 8-bit byte) logical records, organized into a sequence of header data units (HDUs). This logical record length was chosen because it was an integral multiple of the byte and word lengths of all computers that had been sold in the commercial market in or before 1979, the time of the original FITS agreement. Each HDU consists of one or more logical records containing an ASCII header followed by zero or more records of binary data. The first HDU is called the primary HDU; those following are called extensions. The last extension may be followed by 23040-bit *special records*, which need not be organized as HDUs.

Each FITS header record consists of 36 80-byte "card images" written in 7-bit printable ASCII code (ANSI 1977) with the sign bit set to zero. The header may contain as many records as are needed for the card images. The `END` card image is the last. The remainder of the last record of the header is filled with ASCII blanks to its full 23040-bit length. These header records should contain all the information necessary to read and label the associated data. In addition, other information may be provided, such as the processing history or instrument status. The header records are followed by the data records.

The first or primary HDU is governed by special rules. Its data records, if present, contain a matrix of data values, in one of several binary formats, called the primary data array. The array may have no more than 999 axes. There need not be data in the primary data array. An empty primary data array is most likely to appear when all the data of the FITS file are found in extensions.

3.1.1 Primary Header

Each "card image" in the header is in the following form:

keyword = value /comment

Keywords can be no more than eight characters long. The only characters permitted for keywords are upper case (capital) Latin alphabetic, numbers, hyphen, and underscore, with underscore preferred over hyphen. Leading and embedded blanks are forbidden. There are two special classes of keywords: required keywords and reserved keywords. If a keyword is required, then a card image with that keyword must appear in the header. Some keywords are required in all FITS headers; others are required only in conjunction with certain FITS structures. Some structures may require certain keywords to appear in a specific order. Reserved keywords do not have to appear in any header, but they may be used only with the reserved meaning if they do. Users may define their own additional keywords for any FITS file.

The contents of the keyword field determine the structure of the value field. Keywords that have values associated must contain "= " in columns 9 and 10; otherwise, columns 9–80 are regarded as comment. Except for the special cases of the `HISTORY` and `COMMENT` keywords and a blank keyword field (section 3.1.1.2), if a keyword does not have a value, column 9 must not contain "=". The name of the keyword governs whether a value is present. Keyword values have one of four types: logical, character string, integer, or floating point. The following notation will be used to show what type of value must be associated with each keyword:

`KEYWORD` (value type)

The discussion that follows the keyword name will describe the meaning of the value.

The following content is required for the first ten columns of a header card image:

- Keyword name beginning in column 1, ending in or before column 8. The remainder of columns 1–8 is blank filled.
- If the keyword has a value associated with it, "=" in column 9, followed by a blank in column 10.
- If the keyword has no value and is not the reserved `HISTORY`, `COMMENT`, or blank field, any content other than "= " in columns 9 and 10.
- For the reserved `HISTORY`, `COMMENT`, and blank field keywords, the contents of columns 9 and 10 are not restricted.

To simplify the process of writing software to read FITS files, the following fixed format is mandatory for values of the required keywords. For the same reason, its use is strongly recommended for other keywords. Departures should be restricted to cases where use of fixed format is impossible for some reason. Use of the fixed format simplifies the task of the software in determining the type of the value. The structure of a fixed format value field depends upon its type:

- (Logical) `T` or `F` in column 30.
- (Character string) A beginning "`'`" in column 11 and an ending "`'`" in or after column 20 but no later than column 80, with the string in between. To include a single quote within a character string, use two successive single quotes; for example: *O'Hara* becomes `'O''HARA '`. Leading blanks in a string are significant; trailing blanks are not.
- Real part (integer or floating) right justified, ending in column 30, 20 columns maximum.
- Imaginary part (integer or floating) right justified, ending in column 50, 20 columns maximum, i.e., starting in column 31 or after.

The 20 columns specified in the fixed format may not always be sufficient to hold the full precision of a real floating point number or either the real or imaginary part of a complex floating point number. None of the required keywords used for the primary header, random groups or the ASCII table, binary table or image extensions requires a floating point value, and developers of new extensions might be well advised to avoid required keywords with floating point values. Many reserved keywords have floating point values, but there is no standard practice or recommended procedure for handling floating point values with too many digits to fit in 20 columns.

When the fixed format is not used, the value field must be written in a notation consistent with the list-directed read operations in FORTRAN-77. (With the exception of complex numbers, all the fixed formats follow the rules for FORTRAN-77 list-directed read.) Such a notation has the following requirements for the different formats:

- (Logical) The first non-blank character in columns 11-80 is `T` or `F`.
- (Character string) Begins with a "`'`" in column 11 or later, and ends with a "`'`" no later than column 80, with the string in between. Starting in column 11, only blanks are permitted in the value field before the opening quote. To include a single quote within a character string, use two successive single quotes; for example: *O'Hara* becomes `'O''HARA '`. Leading blanks in a string are significant; trailing blanks are not.
- (Integer) May occupy any of columns 11-80.
- (Floating) May occupy any of columns 11-80. The decimal point must always appear explicitly, whether or not exponential notation is used. When exponential notation is used, all letters, (e.g., `E`, indicating an exponential) must be in capitals.
- (Complex) Consists of real and imaginary components (integer or floating point), anywhere in columns 11-80, separated by at least one column.

Any information in a character string value that the reading software needs to retrieve the data from the FITS file should be in the first eight characters, for the benefit of primitive systems. Users should name their strings accordingly. If the string is used only to provide descriptive information or in the scientific interpretation of the file, it is not critical to have the meaning clear from the first eight characters, but it is still a good idea.

Comments may be incorporated in a header card image whether or not a value is present. If a value is present, place a slash (hexadecimal `2F`) between the value and comment field, with at least one blank between the end of the value and the slash. For the fixed format, the slash and space are not required, but using them when writing a FITS file will simplify the task of the reader. If the fixed format is not used, a slash serving as a delimiter, a requirement derived from FORTRAN list-directed read, is required before the comment. If the keyword has no associated value (which is immediately apparent when column 9 does not contain "="), then the entire content of columns 9–80 is a comment. In such a case, it is best to leave column 9 blank.

The first keyword in a header must be `SIMPLE` and have a value of `T` (true) if the file conforms to FITS standards. Subsequent keywords convey the form in which array values are stored, the number of dimensions in the array, and the length of each axis. The coordinate system, scaling, and other information may be given at user option.

Note that while the FITS required keywords will always be the same, the interpretation of the associated values may be expanded. For example, negative values of the keyword `BITPIX` are now used to indicate IEEE floating point data, and the `PCOUNT` keyword, which originated in connection with random groups, was then used in the generalized extensions definition and has now been adopted to describe the heap of variable length arrays logically included in a binary table. Neither later usage was part of the original definition.

Many keywords, called *indexed* keywords, consist of an alphabetic root and a number. If there is only one index, the number does not have leading zeroes: `NAXIS1`, not `NAXIS001`; `TTYPE11`, not `TTYPE011`. Such keywords are represented in this *Guide* by an upper case root followed by a lower case n, for example `NAXIS`n, `TTYPE`n. If there is more than one index, as in the `PC` matrix discussed in section 4.2.2.2, leading zeroes may be needed to delimit the indexes, for example `PC012001`, as `PC121` does not distinguish between $P_{12,1}$ and $P_{1,21}$.

3.1.1.1 Required Keywords

The following keywords are *required* for *all Basic FITS headers for all time*, and must appear in the order given below. All except `SIMPLE` must appear in other headers as well, in the same order. The value field must appear in the fixed format described in section 3.1.1.

1. `SIMPLE` (logical) - A value of "`T`" signifies that the file conforms to FITS standards. A value of "`F`" is used for files that resemble FITS files but depart from the standards in some significant way. One example would be files where the numbers are in the DEC VAX internal storage format rather than the standard FITS most significant byte first. The "FITS" files produced by some hardware that contain non-standard data formats such as two-byte unsigned integers or give the month number before the day number in date formats are another example. Such files might be convenient for internal use by a particular organization or for exchange between users with the same hardware for whom convenience is more important than standardization, when they wish the files to have an overall FITS-like structure. No installation should use them as the standard format for communication with outside users. Files with `SIMPLE` = F should not be described as FITS files.

2. `BITPIX` (integer) describes how an array value is represented:

8	ASCII characters or 8-bit unsigned integers
16	16-bit, twos complement signed integers
32	32-bit, twos complement signed integers
−32	IEEE 32-bit floating point values
−64	IEEE 64-bit floating point values

   No other values for `BITPIX` are valid.

   The name comes from the original FITS design, in which the values of the array were regarded as pixels in a digital image (BITs per PIXel). With the use of negative values of `BITPIX` to signify floating point array values, the number of bits per data array member is the absolute value of `BITPIX`.

3. `NAXIS` (integer) is, for the primary header, the number of axes in the data following the associated primary data array. A value of zero is acceptable and indicates that no data are associated with the current header. The most common reason for a primary HDU with no data is that all the data in the file are extensions. The maximum possible value is 999. Negative values are not allowed.

4. `NAXIS`$n, n = 1, \ldots,$`NAXIS` (`NAXIS`=0 -> `NAXIS1` not present) (integer) is the number of elements along axis n of the array; `NAXIS1` describes the most rapidly varying index of the array, `NAXIS2` the second most rapidly varying, etc. This convention is the same as the one used in FORTRAN. A value of zero for any of the `NAXIS`n signifies that no data array is associated with the header. None of the `NAXIS`n may be negative. The rules of FITS do not strictly forbid use of `NAXIS`n keywords for values of

n >NAXIS. While some groups use such keywords for special purposes, the practice is not recommended as a general rule.

...the other keywords follow until...

5. `END` (no value) - The last keyword must be `END`. This card image has no "=" in column 9 or value field but is filled with ASCII blanks.

Other keywords may appear only between the last `NAXIS`n and `END` keywords. The remainder of the last header record should be filled with ASCII blanks.

These keywords prescribe the size of the primary data array in bits, NBITS, through equation 3.1,

$$\begin{aligned} \text{NBITS} &= \text{ABS(BITPIX)} \times \\ &\quad (\text{NAXIS1} \times \text{NAXIS2} \times \cdots \times \text{NAXIS}m), \end{aligned} \tag{3.1}$$

where m is the value of `NAXIS` and the keyword names represent the values of those keywords.

Examples 1 and 2 in Appendix A illustrate primary headers that precede data arrays.

3.1.1.2 Reserved Keywords

Many of the following reserved keywords were originally suggested by Wells, Greisen, and Harten (1981). If a reserved keyword is used, the meaning and structure must be as described here. Keywords other than the reserved keywords should not be used in their place to express the same concepts. Reserved keywords may appear in any order between the required keywords and the `END` keyword.

Some of these keywords describe the data array.

- `BUNIT` (character) represents the physical units of the quantity stored in the array, e.g., Janskys, magnitudes/pixel. The name comes from "brightness units." Section 3.1.1.4 discusses recommendations for choice of units.
- `BSCALE` (floating) is a scale factor used in converting array elements stored on the FITS data set to physical values:

$$\text{physical\_value} = (\text{FITS\_value}) \times \text{BSCALE} + \text{BZERO}. \qquad (3.2)$$

If this keyword is not present, the scale factor is assumed to be 1.

- `BZERO` (floating) is the offset, the physical value corresponding to a stored array value of zero, in equation 3.2. If this keyword is not present, the offset is assumed to be zero.

  Be careful about possible overflows when using `BSCALE` and `BZERO` if the array elements are floating point (value of `BITPIX` < 0). Pay attention to the likely order of magnitude of the resulting values and be prepared to trap overflows if necessary.

  While the values of `BZERO` and `BSCALE` are floating point, the rules of FITS do not specify the data type of the result of scaling, as this result is not part of the FITS file. Whether the result is to be considered an integer or real number depends on the physical significance of the number.

- `BLANK` (integer) - If the text "`BLANK   `" appears in columns 1–8, then the value will be stored in those elements of an *integer* array that have an undefined physical value. The value of `BLANK` appears in the actual data array, without scaling. It should be regarded as a code, not a number; a scaling transformation with `BSCALE` and `BZERO` should not be applied to array members that have the value given by the `BLANK` keyword. Do not use this keyword if the FITS data array is IEEE floating point, becuse this function is performed by the IEEE floating point NaN. The `BLANK` keyword does not have the same meaning as filling columns 1-8 with ASCII blanks.

The reserved keywords permit complete specification of a linear coordinate system for any axis. Other coordinate systems can be identified through the name given by `CTYPE`n, comments, and user-specified keywords.

- `CTYPE`n (character) is the name of the physical coordinate for axis n (e.g., frequency, RA, Dec).

- `CRPIX`n (floating) is a location along axis n called the reference pixel, or reference point, used in defining the range of values for the physical coordinate of axis n. It is given in units of the counting index. The counting index for axis n runs from 1 to the value of `NAXIS`n, incrementing by one for each pixel or array position. The value of `CRPIX`n may be a fractional index number (e.g., 2.5) and/or be outside the limits of the array; if the array runs over index values 1–10, the reference point may still be -5. The term "reference pixel" originated in the days when the data array was assumed to represent a digital image. The location of the index number relative to an image pixel, i.e., center or corner, is not at present specified in FITS, but the World Coordinates proposal currently under consideration places it at the center, following the common usage in astronomy. A full discussion of this issue appears in section 4.1.

- `CRVAL`n (floating) is the value of the physical coordinate identified by `CTYPE`n at the reference point on axis n.

- `CDELT`n (floating) is the rate of change of the physical coordinate along axis n per unit change in the counting index, evaluated at the reference point.

- `CROTA`n (floating) is the rotation angle, in degrees, of actual axis n of the array from the coordinate type given by `CTYPE`n. As there is no prescribed rule for describing such rotations, the nature of the rotation should be explained in detail using comments.

Default values have not been defined for any of these keywords.

These reserved keywords, from FITS Paper I, allow the definition of simple rectangular coordinate systems, but they do not prescribe the relation between the plane rectangular coordinate system of the FITS array and the spherical coordinate region of the sky that it represents. This question, along with the current comprehensive proposal under consideration by the FITS community, is discussed in section 4. Part of the proposal replaces the simple (`CROTA`n, `CDELT`n) method with a more comprehensive method of defining coordinate transformations in three dimensions.

- `DATAMAX` (floating) is the maximum data value in the array, after any scaling transformation has been applied to the stored array value.

- `DATAMIN` (floating) is the minimum data value in the array, after any scaling transformation has been applied to the stored array value.

Note that `DATAMAX` and `DATAMIN` apply to the physical values represented, not to the numbers in the FITS file. In determining the values for `DATAMAX` and

DATAMIN, special values such as the IEEE special values and values derived from integer array members set to the value of the BLANK keyword are not considered.

Some keywords provide information on the observations represented or the production of the data set.

- DATE (character) is the date the file was written ('dd/mm/yy'—note order). UT is recommended. The value may refer to the creation date of the original file rather than that of the current copy. This rule allows files to be copied without changing the value of the DATE keyword. Currently, date, month, and year are always two digits, with a leading zero if necessary. With this format, the century is ambiguous.
- DATE-OBS (character) is the date of data acquisition (UT recommended); it tells when the observations were made ('dd/mm/yy'— note order). Whether this value (and that of DATE) refers to the start, midpoint, or end of the relevant time interval is not specified. Use comments to say.

  With the turn of the century approaching, there have been extensive discussions as to how to distinguish the century when specifying a date. A proposal based on the ISO 8601 format is now under consideration by the regional FITS committees.
- ORIGIN (character) is the installation where the FITS file is being written.
- TELESCOP (character) is the data acquisition telescope.
- INSTRUME (character) is the data acquisition instrument.
- OBSERVER (character) is the observer name or other identification.
- OBJECT (character) is the object observed.
- AUTHOR (character) identifies who compiled the data associated with the header. Use this keyword when the data come from a published paper or have been compiled from many sources. It refers to the author of the data being carried by the FITS file, not to the creator of the computer-readable form. It should not be used to identify the software that created the FITS file or the writer of that software. The CREATOR keyword (section 5.6.1) has been suggested to identify the software creating a FITS file.
- REFERENC (character) is the bibliographic reference for the data associated with the header.

- EQUINOX (floating) is the equinox of the coordinate system (in years). In early FITS data sets, the keyword EPOCH was used with this meaning. Since the term "epoch" has a different meaning, referring to the actual date of observation, the correct term EQUINOX should be used in the future. EPOCH will still be accepted with this meaning in old data sets and should not be otherwise used. Software to read FITS data sets should interpret EPOCH as equivalent to EQUINOX. The epoch or date of observation can be given as the value of the DATE-OBS keyword.
- BLOCKED (logical) - deprecated - when set to the value T advises that a tape may be blocked with more than one logical record per physical block. A value of F has no meaning. Its significance is now primarily historical and is discussed further in section 3.8. However, because the keyword name BLOCKED has been reserved, it cannot be used for any other purpose.

Other keywords signal a card image with comments or other text.

- Columns 1–8 blank, no keyword, means columns 9–80 are a comment.
- COMMENT (none) means columns 9–80 are a comment.
- HISTORY (none) means columns 9–80 are a comment. This keyword is intended for use when the associated text discusses the history of how the data contained in the array were processed.

These keywords are the only ones without values that can have "=" in column 9. Users may define other keywords to contain comments as well. For these keywords, column 9 may not contain "=". The reason this restriction does not apply to the COMMENT, HISTORY and ␣␣␣␣␣␣␣␣ keywords is principally historical; the original FITS paper defined columns 9-80 as being a comment and did not restrict the content. For the COMMENT, HISTORY, and blank field keywords, the reader will be able to tell by the keyword name that there is no value. However, for user-defined keywords, the reader has no other way of telling *a priori* whether the field has a value or not. To minimize confusion, it is best to avoid "=" in column 9 even after COMMENT, HISTORY, and blank keyword fields as well.

3.1.1.3 Some Hints on Keyword Usage

While only the keywords listed in 3.1.1.1 and 3.1.1.2 are formally reserved by the rules of FITS, standard meanings for others may be adopted by general agreement. Section 4 discusses a number of such keywords. Conventions for keyword meanings can also be developed within a given discipline, as has been done for high energy astrophysics, radio interferometry, and single dish radio observations. Some of these conventions are discussed in section 5. Keywords should not be used in a way that conflicts with a generally accepted convention. If the convention applies to all FITS files, avoid using the keyword with a meaning other than that of the convention; if the convention is discipline-specific, say, to high energy astrophysics or radio astronomy, a different meaning may apply to a different discipline. Keyword meanings in conflict with widely accepted usage may confuse readers.

For keywords that have no value or are used to transfer text, such as `HISTORY`, `COMMENT`, and `REFERENCE`, the same keyword may be used repeatedly to extend text over a sequence of card images. However, repetition of a keyword with conflicting values may cause confusion. There is no standard interpretation in such a case determining which value is to be retained as the true value of the keyword. Different FITS readers may interpret the header differently. Do not repeat a keyword on different card images if the values conflict.

The comment field following the keyword and value provides an opportunity to explain the meanings. Such comments should supply additional information. Some FITS writers automatically generate comments such as "Physical coordinate of axis" following a `CTYPE`*n* keyword or "Physical units of matrix" following a `BUNIT` keyword. Such comments provide no more information than has been given by the FITS rules and are not very helpful for the human reader. Comments should provide information not available from the FITS syntax, for example, what the axis is (right ascension? declination? frequency?) or the units of the values in the primary data array (Janskys/beam? magnitudes/arcsec^2?). The number of characters in a string value is often limited to accommodate reading software, resulting in abbreviations; the comment field can and should be used to explain these abbreviations in full. While the meaning of an abbreviation may be obvious at the writing installation, it may not be so clear to a reader.

3.1.1.4 Units

The units named in the IAU (1988) *Style Guide* are recommended for array and keyword values, with the exception of measurements of angles. This set of units includes the SI units and additional units that are standard in astronomy, for example, magnitudes. Using `BUNIT` and `TUNIT`n keywords will remove any confusion; similarly, the comment field following `CTYPE`n keywords can be used to specify the units. Fractional units should not be used. If, for example, for space reasons, temperatures must be listed in the file in units of tens of degrees, the units specified by `BUNIT` should be degrees, and scaling (section 3.1.2.1) should be used to transform from the file values to physical values. Always explicitly describe non-standard units. A copy of the IAU Style Guide recommendations is available on the IAU World Wide Web site, currently at `http://www.lsw.uni-heidelberg.de/iau/units.html`.

For celestial coordinate systems specified with the `CTYPE`n, `CRVAL`n, `CDELT`n, and `CROTA`n keywords, or with the keywords used in the world coordinates conventions described in section 4, the use of decimal degrees for the angular measurements is required, and degrees are recommended as the units for other angular measurements (with, for example, the value of `BUNIT` = 'deg ').

3.1.2 Primary Data Array

In the original design of FITS, the data array represented a digitized image. Each array element would therefore correspond to a pixel of this image. Consequently, while the elements of a FITS data array are often called "pixels", the array does not represent an image and the term "pixel" may simply refer to a member of an array rather than a section of an actual image.

The primary data array starts at the beginning of the record following the last primary header record. The data occur in the order defined by the header—pixel numbers increasing, with the index along axis 1 varying most rapidly and the index along the last axis varying least rapidly. Data are packed into the 2880-byte records with no gaps. Each entry in the array immediately follows the preceding entry; the first pixel of any given axis does not necessarily appear in the first word of a new record. If the last member of the data array is not at the end of a record, the remaining bits of the record are filled with zeroes, corresponding to 0 (integer) or +0.0 (floating point).

The bytes in each word are in order of decreasing significance, with the sign bit first. Thus, INTEL and VAX machines will have to reverse the order of the bytes in 16- and 32-bit integers and make the appropriate conversions between IEEE floating point and internal storage order before writing or after reading. Numbers are stored in twos-complement form; conversion will be required for ones-complement machines.

The data type of the array members may be any type corresponding to a valid value of `BITPIX`: unsigned eight-bit integers or characters, signed 16- or 32-bit integers, or single or double precision IEEE floating point. Proposals to add 16- and 32-bit unsigned integer types have been discussed from time to time, but a consensus in favor of these proposals has not developed. Data with the numerical range corresponding to unsigned integers can be included in a FITS file by using signed integers and then scaling, for example, setting `BZERO` equal to 32768 (2^{15}) for 16-bit integers.

Some detector systems produce files that conform to the FITS rules except in their use of multibyte unsigned integers; these files should not be described as FITS files and should not have `SIMPLE` set equal to `T` in the first card image. Such files are an example of a good use for setting `SIMPLE` equal to `F`.

3.1.2.1 Scaled Integers

When FITS was originally developed, there was no standard format for the internal representation of floating point data. Only integer formats were allowed in the data array. The scaling scheme using the keywords `BZERO` and `BSCALE` was devised in order to make it possible to represent inherently floating point physical values using a data array of integers. The physical value of the data represented by an array member would be derived from equation 3.2. While scaling is no longer required to represent floating point numbers, it remains a useful tool. Representation of unsigned integer quantities is one application.

Be careful when attempting to scale data sets with large dynamic range. If one converts such data to integers and then tries to convert back to floating point, so much precision may be lost that the original data values will be irretrievable.

3.1.2.2 Undefined Integers

The value of the keyword `BLANK` is used to identify undefined integer array values, which might result from data dropouts or from operations that are undefined (like a divide by zero). For 16-bit integers, the most commonly used value of `BLANK` is -32768, but users may choose the value most appropriate for their data. The BLANK keyword is not used for floating point data, where the IEEE NaN identifies undefined values.

3.1.2.3 IEEE Floating Point Data

FITS allows transmission of 32- and 64-bit floating point data within the FITS format using the IEEE (1985) standard. This Floating Point Agreement also applies to random groups records and to any extensions for which BITPIX is not explicitly restricted (e.g., BITPIX=8 for XTENSION= 'TABLE   '). Values for BITPIX of −32 and −64 indicate IEEE single- and double-precision floating point data, respectively.

The text of the Floating Point Agreement (Wells and Grosbøl 1990) is as follows:

> The Basic FITS, Random Groups and Generalized Extensions Agreements are revised to add IEEE-754 32- and 64-bit floating point numbers to the original set of FITS data types. BITPIX=−32 and BITPIX=−64 signify 32- and 64-bit IEEE floating point numbers; the absolute value of BITPIX is used for computing the sizes of data structures. The full IEEE set of number forms are allowed for FITS interchange, including all special values (e.g., the "not-a-number" cases). The order of the bytes is sign and exponent first, followed by the mantissa bytes in order of decreasing significance. The BLANK keyword is ignored by FITS readers when BITPIX=−32 or −64.

For a complete, precise description of the IEEE floating point format, refer to the IEEE standard. The following discussion is provided to help in the interpretation of floating point data.

An ordinary IEEE floating point number consists of three components: a sign, an exponent, and a fraction. For regular IEEE 32-bit floating point numbers, the sign is contained in bit 1, the exponent in bits 2–9, and the fraction in bits 10–32. The fraction has an implied binary point in front. The value is given by

$$\text{value} = (-1)^{\text{sign}} \times 2^{(\text{exponent-127})} \times (1+\text{fraction}). \tag{3.3}$$

For regular IEEE 64-bit floating point numbers, the sign is contained in bit 1, the exponent in bits 2–12, and the fraction in bits 13–64. The fraction has an implied binary point in front. The value is given by

$$\text{value} \;=\; (-1)^{\text{sign}} \times 2^{(\text{exponent-1023})} \times (1{+}\text{fraction}). \tag{3.4}$$

Fraction bytes are in order of decreasing significance (i.e., the standard non-byte-swapped order).

For example, suppose the single precision 8-bit byte pattern is `40400000`. The sign bit is `0`, the exponent bit pattern is `100 0000 0` (or 128), and the fraction pattern is `1` followed by 22 `0`s with a binary point in front, or 0.5 decimal. The entire number is interpreted as

$$(-1)^0 \times 2^{(128\text{-}127)} \times 1.5 \;=\; 3. \tag{3.5}$$

The IEEE standard specifies in addition a variety of special exponent and fraction values in order to support the concepts of plus and minus infinity, plus and minus zero, "denormalized" numbers and "not-a-number" (NaN). The `BLANK` keyword of the original FITS Agreement is thus unnecessary and should be omitted by FITS writers and ignored by FITS readers when `BITPIX` $= -32$ or -64 (the NaNs of the IEEE format will act as the blank). FITS writers should not write the `BLANK` keyword if `BITPIX` $= -32$ or -64. For denormalized numbers, 1 is not added to the fraction, and the offset subtracted from the exponent is one smaller than for regular numbers: 126 for single precision and 1023 for double precision. This convention allows IEEE floating point to represent numbers that are smaller than those represented by the regularly defined values, although the number of significant digits decreases for smaller values. All of these special cases are fully accepted for FITS interchange. Appendix B lists the kind of value, regular or special, represented by all possible bit patterns.

The `BSCALE` and `BZERO` values should be applied by FITS readers if they differ from 1.0 and 0.0. However, scaling parameters should be used carefully with floating point values, because of the risk of generating overflows and underflows after scaling has been applied.

3.2 Random Groups

The random groups structure (Greisen and Harten 1981) was originally designed for applications in radio astronomy but was intended for other applications as well. It never gained acceptance outside the radio interferometry community, and some FITS readers designed for other communities cannot read it. In addition, the random groups records are a structural anomaly in FITS, as they are the only records that do not conform to the primary HDU – extensions – special records sequence. All of the original objectives originally intended for random groups can now be achieved through the use of the now standard binary table format described in section 3.6. Even the interferometry community, for whom random groups was originally intended, is developing new conventions based upon binary tables. Do not create FITS files using random groups; use binary tables instead. The description of random groups presented here is intended as a reference for those who may have to read old random groups FITS files, not as an encouragement to write them. A number of keywords have been reserved for the random groups structure, and may not be used for any other purpose, unless specifically stated in the FITS rules. Also, the random groups structure is the original source of the `PCOUNT` and `GCOUNT` keywords that were incorporated into the Generalized Extensions rules discussed in section 3.3.

While the basic array structure of FITS can handle a data matrix when the data are distributed evenly along all axes, sometimes the data may not be distributed uniformly along one or more axes. The random groups structure was created to handle such situations. Instead of being followed by a data array, the primary header is followed by a special set of records, of standard FITS 23040-bit size, called random groups records. These records contain a series of groups, each consisting of a sequence of parameters followed by an array. The parameters carry the data whose spacing is not uniform and which are not ordered, i.e., random, hence the name. The number of random parameters and the dimensions of the array must be the same in all groups.

On application is for *uv* interferometric visibility data; in fact, random groups are sometimes referred to as UV FITS. Consider a set of weighted complex fringe visibilities for the four Stokes polarizations at a sequence of evenly spaced frequencies. The axes are u, v, w, hour angle, baseline, fringe visibility information (real part, complex part, and weight), Stokes parameter, and frequency. The points along the frequency axis are evenly spaced, and the fringe visibility and Stokes axes can both be handled by using “evenly spaced” integer index points to represent the separate components – real, complex, and weight – for the visibility and the four Stokes parameters. Thus, an individual observation can easily be structured as a matrix with axes of visibility components, Stokes parameter, and frequency. However, the individual matrices are at nonuniformly distributed values of baseline, hour angle, and (u, v, w).

Using the random groups structure, the values of baseline, hour angle, and (u, v, w) would be specified as parameters before each (visibility, polarization, frequency) matrix. The combination of parameters and array constitutes a group. The structure of the group would be given by the form

$$|r_1, r_2, r_3, r_4, \ldots, r_N|p_{111}, p_{112}, \ldots, p_{lmn}| \tag{3.6}$$

corresponding, in this example, to

> $|u, v, w$, time, baseline$|$(visibility component, Stokes parameter, frequency)$|$

where $r_1, \ldots, r_N$ are random parameters 1 through N and $p_{111}, \ldots, p_{lmn}$ are pixel values in the order defined for a data matrix. The value of p could, in principle, be as large as 998 for each axis, as opposed to the 999 maximum for Basic FITS. The data array p_{ijk} starts immediately after the last parameter, r_N. The storage order and internal representation of a random groups data array are the same as for a Basic FITS simple array. It is interesting to note that the Basic FITS data structure is actually a subset of this more general structure, with no parameters.

3.2.1 Header

One of the signatures of the random groups structure is that the `NAXIS1` keyword is set equal to zero. The `NAXIS` and `NAXIS`n keywords will therefore have somewhat different meanings than in Basic FITS.

3.2.1.1 Required Keywords

The keywords required for an ordinary array are also required for random groups, in the same order.

1. `SIMPLE` (logical) "`T`" indicates a file in conformance to standard FITS.
2. `BITPIX` (integer) describes the representation of the values of the individual arrays and of the parameters. The parameters must be of the same data type as the array members. The values have the same meaning as for Basic FITS.

3. `NAXIS` (integer) is one more than the number of axes in each array (because `NAXIS1` is used as a groups indicator instead of an actual axis). The largest possible value is 999, representing 998 axes.
4. `NAXIS1` (integer) is set to 0 to indicate that no primary data array follows the primary header.
5. `NAXIS`n, $n = 2, \ldots,$ `NAXIS` (integer) is the number of elements along axis $n-1$ of the array. The index varies most rapidly along the $n = 2$ axis, least rapidly along the $n =$NAXIS axis.

There must be no other keywords between the ones listed above. Additional keywords follow, which must include among them

- `GROUPS` (logical) must have the value "T", to indicate that the data records following the primary data array are random groups.
- `PCOUNT` (integer) is the number of parameters preceding each data array.
- `GCOUNT` (integer) is the number of groups (parameters + array) in the random groups records.
- `END` (no value) is the last keyword. The header is filled with ASCII blanks.

The `PCOUNT` and `GCOUNT` keywords tell the reader the number NBITS of bits in the random groups records exclusive of fill:

$$\begin{aligned} \mathrm{NBITS} &= \mathrm{ABS(BITPIX)} \times \mathrm{GCOUNT} \times \\ &\quad (\mathrm{PCOUNT} + \mathrm{NAXIS2} \times \mathrm{NAXIS3} \times \cdots \times \mathrm{NAXIS}m), \end{aligned} \tag{3.7}$$

where m is the value of `NAXIS`.

3.2.1.2 Random Parameter Reserved Keywords

The random parameters may be labeled and scaled in a fashion similar to the members and axes of a Basic FITS array, using the keywords `PSCAL`n and `PZERO`n:

$$\mathrm{physical\_value} = (\mathrm{FITS\_value}) \times \mathrm{PSCAL}n + \mathrm{PZERO}n. \tag{3.8}$$

The following keywords are reserved for describing random parameters.

- `PTYPE`n (character) is the name of the quantity represented by the nth random parameter.
- `PSCAL`n (floating) gives the scale factor for random parameter n. If this keyword is absent, the scale factor is assumed to be 1.
- `PZERO`n (floating) gives the offset for random parameter n. If this keyword is absent, the offset is assumed to be zero.

Even though the random parameters must have the same data type as the array elements, greater precision can be achieved in the random parameters than in the data array through the use of repeated `PTYPE`n keywords. If more than one `PTYPE`n keyword has the same value, then the parameters associated with all those `PTYPE`n keywords are being used to represent the same quantity. The value of that quantity is derived by summing the true values derived for the parameters n corresponding to those `PTYPE`n keywords – the values derived using the stored values and the `PSCAL`n and `PZERO`n values. For example, if

```
PTYPE1  = 'GLON    '
PTYPE2  = 'GLON    '
PSCAL1  =                  1.0
PSCAL2  =              1.0E-04
```

where `GLON` stands for Galactic longitude, then the value of Galactic longitude for each group is derived by adding the value of the first random parameter for that group to 10^{-4} times the value of the second random parameter.

The rules governing units in random groups are the same as those for a primary data array, as given in section 3.1.1.4.

3.2.2 Data Records

The data records begin in the first record following the last header record, similarly to the way a primary data array is stored. The first parameter of the first group is at the start of the first data record. The first member of the array in each group follows the last parameter, with no spaces or fill. It does not start a new record. Similarly, each parameter-array group immediately follows the previous one; a new group need not begin a new record. The same data types are allowed as for Basic FITS. The data type for the random parameters *must* be the same as the data type for the data arrays.

3.3 Extensions

The data to be transported do not always fit conveniently into an array format. Also, auxiliary information associated with an image or data set may need to be saved in the same file, but may not fit into the array structure. To handle such cases, the concept of FITS extensions has been introduced (Grosbøl *et al.* 1988). Extensions have the same overall organization as Basic FITS: a header consisting of keyword = value ASCII card images followed by data. As in Basic FITS, the header and data each consists of an integral number of 23040-bit records. Except for the `SIMPLE` keyword, which is replaced by an `XTENSION` keyword, the header must use the keywords required for the Basic FITS primary header (section 3.1.1.1). As in Basic FITS, the data start in the first record following the last header record. However, as the data structure need not be a simple array, there are additional required and reserved keywords for particular structures.

The developers of the extension concept realized that a set of rules was needed to prevent conflicts between a new extension and Basic FITS, random groups, or an existing extension. The rules that were developed are called the Generalized Extensions Agreement. An extension that conforms to these rules is called a conforming extension. An extension that has been endorsed by the IAU FITS Working Group or other appropriate authority is called a standard extension. All standard extensions are conforming extensions. The detailed structure of a conforming extension does not need approval by any authority, but the type name must be unique and registered with the IAUFWG. A file whose extensions all conform to the rules for generalized extensions is in FITS conformance, even if the extensions are not standard extensions, and the `SIMPLE` keyword of the primary header should have the value `T`.

Two basic rules govern all existing extensions and serve as the foundation for the more detailed requirements for generalized extensions:

- All FITS extensions must appear after the main FITS header and its associated primary data array.
- Each extension begins at the start of a new 2880-byte record.

The generalized requirements for FITS extensions are as follows:

- Extensions should have the same structure as the Basic FITS HDU: header plus data. The extension should be self-defining and the header readable both by people and machines. The rules that apply to creating FITS headers also apply to extension headers; they should contain a required set of standard keywords and consist of ASCII text card images. While the information in an extension may be of any length, the extension as a whole must be filled to an integral number of 23040-bit records.
- Only the binary and character coding conventions specified in the original FITS papers and the Floating Point Agreement should be used in FITS extensions.
- It should be possible to append any number of extensions to a primary header and its associated data array.
- If there is more than one type of extension in a FITS file, then the extensions may appear in any order.
- Existing FITS formatted data, including those with standard extensions, must be compatible with the new extension standard. FITS files that contain combinations of new and standard extensions must be allowed in order to facilitate the transition to a new design.
- The presence of a new extension in a FITS file should not affect the operation of a program that does not know about the new type of extension.
- Any program scanning a FITS file, on any medium, should be able to locate the beginning of any extension and find the start of the next one. This rule implies that the extension header must specify in a consistent and standardized manner the total number of data bits that are associated with it.
- It must be possible to devise new types of extensions without prior approval. *Keywords* in the primary FITS header may not be used to

announce the existence of a particular type of extension because such keywords would need to be approved by the IAUFWG.

- Anyone wishing to create a new extension format is free to do so. Creators of new extensions should check the list of extension type names registered with the IAUFWG, which is maintained on-line by the FITS Support Office, and register a new type name for their format.

- No information in either the primary header or the extension header should specify the physical block sizes of FITS blocks on tape or any other medium (but see the discussion of section 3.1.1.2 about the `BLOCKED` keyword).

The detailed rules for FITS extensions implement these requirements. Keyword syntax for extension headers is governed by the same rules as for primary headers. The standards concerning data types and bit order for the main FITS data records also apply to extensions.

The Generalized Extensions Agreement requires the use of a single additional keyword in the main header only:

`EXTEND` (logical) if true (`T`) indicates that there *may* be extensions that conform to the generalized extension standards following the data records. Its absence guarantees that there are no extensions, but its presence does not guarantee that extensions are present. Because extensions are not required following a primary header with `EXTEND` = T, the primary HDU alone can be copied unchanged. A value of `F` has no meaning and should not be used. The `EXTEND` keyword must immediately follow the last `NAXIS`n keyword or an `NAXIS` keyword with a value of zero. FITS data in a file may consist entirely of extensions, with no primary data. In such cases, the primary header must still be present, with `EXTEND` = T and `NAXIS` = 0 to indicate no primary data array. Examples 3 and 6 in Appendix A illustrate FITS primary headers with no data that occur at the start of files where all the data are in extensions.

3.3.1 Required Keywords for an Extension Header

The first keywords of a FITS extension header, from `XTENSION` to the last `NAXIS`n, *must* appear consecutively in the order listed below.

1. `XTENSION` (character) indicates the type of extension. This keyword must appear on the first card image in the header. With this keyword at the front, any software can immediately read the extension type and determine whether or not it is a type that the software can handle.

2. `BITPIX` (integer) gives the number of bits per "pixel" value. The values allowed are the same as those for the primary data array.

3. `NAXIS` (integer) gives the number of "axes" or analogous structures; a value of zero is allowed, which indicates there are no data records in the current extension. The maximum possible value is 999. Negative values are not allowed.

4. `NAXIS`$n, n = 1, \ldots,$ `NAXIS` (integer) (`NAXIS`n=0 for any n implies that the extension has no data) is the number of array elements along the nth axis or its analog.

In order to allow FITS readers that are unfamiliar with an extension to skip it, a header must specify the size of the extension in bits. To do so, two parameters, `PCOUNT` and `GCOUNT`, are defined in the header:

- `PCOUNT` (integer) - The value may be zero. Note that although this keyword originated in the Random Groups (section 3.2) rules, the interpretation for an extension may be different.

- `GCOUNT` (integer) - If simple image-like data (e.g., a table) are being written, its value will be 1.

  The `PCOUNT` and `GCOUNT` keywords may appear anywhere between the last `NAXIS`n (or `NAXIS`=0) keyword and the `END` keyword. The best place is immediately after the last `NAXIS`n keyword; the existing standard extensions require that location, but otherwise software to read FITS files should not assume they will be there. They may be defined in any way that, along with the `BITPIX`, `NAXIS`, and `NAXIS`n values, gives the correct data size using the following formula to derive NBITS, the number of bits excluding fill:

$$\begin{aligned} \mathrm{NBITS} &= \mathrm{ABS(BITPIX)} \times \mathrm{GCOUNT} \times \\ &\quad (\mathrm{PCOUNT} + \mathrm{NAXIS1} \times \mathrm{NAXIS2} \times \cdots \times \mathrm{NAXIS}m) \end{aligned} \tag{3.9}$$

  where m is the value of `NAXIS`. NBITS may not be negative.

The number NRECORDS of data records following the header record is then given by

$$\mathrm{NRECORDS} = \mathrm{INT}[(\mathrm{NBITS} + 23039)/23040)]. \qquad (3.10)$$

Rules for individual extensions may place further restrictions on the position of the `PCOUNT` and `GCOUNT` keywords.

`END` is always the last keyword in a header. The remainder of the record following the `END` keyword is filled with ASCII blanks.

The data begin at the start of the first record following the last record of the header.

3.3.2 Reserved Keywords for Extension Headers

These keywords are not required but, if used, must have the meaning and syntax defined here. They may appear anywhere between the required keywords and the `END` keyword. These keywords are reserved for all extension headers and need not be specifically reserved for an individual new extension.

- `EXTNAME` (character) can be used to give a name to the extension to distinguish it from other extensions of the same type. For example, if there are several tables on a tape, they could be distinguished by their values of `EXTNAME`. The name may have a hierarchical structure giving its relation to other files (e.g., "map1.cleancomp"). Remember, however, that the reader should be able to retrieve the data properly using only the first eight characters of the string. Conventions for hierarchical structures of extension names have not been established.

  The distinction between the name and the type is that the type, the value of `XTENSION`, describes the structure of the extension, while the value of `EXTNAME` provides a name for the extension but says nothing about its structure. For example, the header for an ASCII table containing information about Seyfert galaxies could have `XTENSION = 'TABLE   '` and `EXTNAME = 'SEYFERTS'`.

- `EXTVER` (integer) is a version number which can be used with `EXTNAME` to identify an extension. The default value is 1.

- EXTLEVEL (integer) specifies the level of the extension in a hierarchical structure. The default value for EXTLEVEL should be 1.

Because the term "type" refers to the structure of an extension as given by the value of XTENSION, user definition of a EXTTYPE keyword is not recommended, though not strictly forbidden by the FITS rules.

As a sample application of hierarchical structure, consider an observing session in which several objects are observed. In addition to the actual observations, other information, say, about the observing system configuration or physical state, might be present. Some information might apply to the entire observing session, while other information would apply only to the observations of an individual object. In this case, an extension at EXTLEVEL = 1 could contain the information applying to the entire observing session as part of the header, while some information about the observations of individual objects could appear in the associated data in a master table. Each row in the master table would contain information about the observations of one object. One entry in each row would be the name of a table containing observations of that object. The observations would appear in separate table extensions with EXTLEVEL = 2 and EXTNAME equal to the name given in the master table. This concept has yet to be implemented in data for archive or distribution, and any software to support it is still in the developmental stage.

The grouping proposal, discussed in section 5.3, describes an alternative method of creating hierarchies.

3.3.3 Creating New Extensions

Use an existing standard extension type if you can; to do so facilitates use of existing analysis packages. New extension types are necessary only when information is organized in a way that existing extension types can't handle. Before starting to develop a new format, check with the FITS Support Office or IAUFWG to see if an existing or developing extension can be used to organize your data. Inquiries on `sci.astro.fits` (section 6.2) may also be helpful. If a new proposed extension type under development can be used, coordinate what you do with the developer. If a totally new extension is needed, try to design an extension that has as general an application as possible, with the eventual goal of adoption as a new standard extension. The type name of any new extension must be registered with the IAUFWG and be different from any already registered. The FITS Support Office maintains the current list. Special purpose extensions restricted to particular projects, with the exception of prototypes for more general designs, should be avoided.

3.4 ASCII Table Extension

Astronomical data are frequently organized in tables, for example, a standard source catalog or the results of a series of observations. Another use for tables is to hold information related to observations, such as observing logs and calibration data, which are more conveniently transported in a table accompanying the observations than as keywords. Not surprisingly, the first standard extension developed and endorsed by the IAU was that for tables (Harten *et al.* 1988). This ASCII table extension uses ASCII records to carry the information.

The data appear as a character array, in which the rows represent the lines of a table and the columns represent the characters that make up the tabulated items. Each member of the array is one character or digit. Each row consists of a sequence of fields. The sequence of fields is the same for all rows, and all rows must have the same length. Each field in the table contains one item in the table, either a character string or the ASCII representation of a number in FORTRAN-77 format. A particular field may consist of many columns, as many as are needed to carry the full string or number.

The field organization is described by a series of keywords. These keywords describe the initial column and number of columns for each field. Other kinds of information that can be listed in the header include the units of quantities in the field, scale factors between the values in the table and the physical quantities they represent—analogous to the `BZERO` and `BSCALE` values of Basic FITS—and the string used to represent an undefined value. Fields in the table may be reserved for comments, which would be skipped by automatic decoding software but may be read by simply printing the table. This property allows the transfer of tables that contain a large quantity of comments. Examples 3 and 5 illustrate ASCII table headers.

3.4.1 Required Keywords for ASCII Table Extension

The following keywords are *required* for ASCII table extensions and *must* appear in the first eight card images in the following order:

1. `XTENSION` (character) has the value `'TABLE   '` for ASCII table extensions.
2. `BITPIX` (integer) must have the value of 8, indicating printable ASCII characters.
3. `NAXIS` (integer) must have the value of 2 for ASCII table extension headers. The axes are the rows and columns of the table.
4. `NAXIS1` (integer) gives the number of characters in a table row.
5. `NAXIS2` (integer) gives the number of rows in the table. The value can be 1. While a value of zero is permitted, that value would indicate that no table is present.
6. `PCOUNT` (integer) is required with the value of 0 for the ASCII table extension.
7. `GCOUNT` (integer) is required with the value of 1 for the ASCII table extension. Each extension contains one table; the total number of bytes is NAXIS1 $\times$ NAXIS2.
8. `TFIELDS` (integer) has a value that gives the number of fields in each table row. Values from 0 to 999 are allowed.

Note that while the rules for generalized extensions do not specify where the `PCOUNT` and `GCOUNT` keywords appear between the last `NAXIS`n keyword and the `END` keyword, the ASCII table extension requires that they appear immediately after the `NAXIS2` keyword.

The following required keywords may appear anywhere between the `TFIELDS` and the `END` keywords.

- `TBCOL`n (integer) is the column number of the first character in field n. The first column of a row is numbered 1.

- `TFORM`*n* (character) is the FORTRAN format of field *n* (I, A, E, D, F). To repeat a field, the format must be repeated using separate `TFORM`*n* and `TBCOL`*n* card images; the repeat count (e.g., 2Fx.x) of FORTRAN cannot be used. The `TFORM`*n* values specify the width of the field and are of the form Iw, Aw, Fw.d, Ew.dEe, or Dw.dEe (integers, characters, single precision floating point, single precision exponential, and double precision exponential, respectively). The *w* is the width of the field, *d* the number of digits after the decimal point, and *e* is the number of digits in the exponent, as in FORTRAN. For integer fields, if -0 needs to be distinguished from $+0$ (e.g., degrees of declination), the sign will have to be a separate character field. It is better to use floating point. `TBCOL`*n* and `TFORM`*n* keywords must be present for values of *n* from 1 to the value of `TFIELDS`.

`END` is always the last keyword; the remainder of the header record is filled with ASCII blanks.

3.4.2 Reserved Keywords for ASCII Table Extension

These keywords may be used, in addition to the required keywords described in section 3.4.1, to describe the structure of ASCII table data. They are not required, but if they are used in an ASCII table header, they must be used as described here. These keywords should not be used in other extensions unless they have a function analogous to the one they have in the ASCII table extension. They may appear anywhere between the `TFIELDS` and `END` keywords, in any order.

- `TTYPE`*n* (character) is the heading for field *n*. The recommended characters are letters, digits, and underscore. Upper case values are preferred, but lowercase letters may be used, and string comparisons involving the values of `TTYPE`*n* keywords should not be case sensitive. The hyphen is permitted but *not* recommended. While more than one field may have the same heading (value of `TTYPE`*n*), the practice is not recommended.

- `TUNIT`*n* (character) is the physical units of field *n*. The rules governing the units to be used in ASCII tables are the same as those for a primary data array, as given in section 3.1.1.4.

- TSCALn (floating) is the scale factor for field n, as applied in equation 3.11. The default value is 1.0.

- TZEROn (floating) is the offset for field n (see equation 3.11). The default value is 0.0.

$$\text{Physical\_value} = (\text{Stored\_value}) \times \text{TSCAL}n + \text{TZERO}n. \quad (3.11)$$

  Note: TSCALn and TZEROn may not be used for A-format fields.

- TNULLn (character) is the representation of an undefined value. The string chosen does not have to be readable in the format specified by TFORMn; for example, a null value of '***' might be used for an I3 field. If the null string given as the value of the keyword does not fill the field, it is assumed to begin at the start of the field, and the part of the field following the string is assumed to be filled with blanks. A table will be more efficiently read if TNULLn is specified only when null values actually exist in the field, thus avoiding unnecessary string comparisons for the other fields.

3.4.3 Data Records in an ASCII Table Extension

The table data records begin with the record immediately following the last header record. Each record contains 2880 ASCII characters in the order defined by the header. The first entry of each row immediately follows the last entry of the previous row, and table entries do not necessarily begin at the beginning of a new record. After the end of the valid data, the last record should be filled with ASCII blanks. The table consists of NAXIS2 rows of length NAXIS1. Entries are in the FORTRAN-77 formats specified in the associated header. A-format fields should be encoded as plain text, without being encoded in string quotes. Numbers decoded with the I format may exceed the 16-bit integer range. Blank characters in a field are not interpreted as zeroes; all zeroes, even trailing zeroes, must be explicit. This rule is equivalent to setting the FORTRAN-77 OPEN statement specifier BLANK to NULL.

The w in the value of TFORMn specifies the width of field n. Field n begins in the position given by the value of TBCOLn and includes w characters. The sum of the widths of the different columns need not equal the value of NAXIS1, the length of a row. However, no field may extend beyond the end of the row, to column numbers greater than the value of NAXIS1. Blank columns may be included between fields, a good practice that enhances readability.

3.5 The Image Extension

While multiple arrays of the same data type can be combined into a single array by adding another dimension, there are cases where arrays related to the main data array may have different data types, for example, arrays of flags or weights. Arrays of values with different data types can in principle be stored in the same file by making them different array fields in the same row of a binary table (section 5.2.4.2), but, in practice, many find it simpler to put the arrays containing data types different from the primary data array in separate extensions. For this purpose, the IUE project developed the image extension type (Ponz, Thompson, and Muñoz 1994).

3.5.1 Header

The image extension follows the rules for required keywords in extensions stated in section 3.3.2. The first keywords of a FITS image extension header, through `GCOUNT`, *must* appear consecutively in the order listed below.

1. `XTENSION` (character) has the value `'IMAGE␣␣␣'` for image extensions.
2. `BITPIX` (integer) provides the number of bits per "pixel" value. The values allowed are the same as those for the primary data array.
3. `NAXIS` (integer) is, for an image extension, the number of axes in the extension data array. A value of zero is acceptable and indicates that no data are associated with the current header. The maximum possible value is 999. Negative values are not allowed.
4. `NAXIS`n, $n = 1, \ldots,$`NAXIS` (`NAXIS`=0 –> `NAXIS1` not present) (integer) is the number of elements along axis n of the array. The `NAXIS`n keywords in an image extension are governed by the same rules as those in a primary header.
5. `PCOUNT` (integer) has the value 0 for image extensions.
6. `GCOUNT` (integer) has the value 1 for image extensions.

(. . . the other keywords follow until. . .)

`END` (no value) The last keyword must be `END`. This card image has no "=" in column 9 or value field but is filled with ASCII blanks.

The remainder of the last header record should be filled with ASCII blanks.

Note that while the rules for generalized extensions do not specify where the `PCOUNT` and `GCOUNT` keywords are located between the last `NAXIS`n and the `END` keywords, the image extension rules require that they appear immediately after the last `NAXIS`n card image.

3.5.2 Data Records

The rules for the data in image extensions are the same as for the primary data array (section 3.1.2). An image extension header cannot be followed by random groups records.

3.6 Binary Tables

With the development of the IEEE-754 standard for floating point numbers, it became possible to incorporate binary values in tables. Tables with binary values are more compact, and the time spent converting from ASCII is eliminated, although display is not as direct as for ASCII tables. In developing the binary table design, a provision was made that a particular field or entry in the table could be a vector, not just a single number. The resulting binary table extension (Cotton, Tody, and Pence 1995; hereafter CTP) was endorsed by the IAUFWG in 1994. Like an ASCII table, a binary table contains a two-dimensional data matrix mapped into the rows and columns of a table. The entries, the values associated with a given row and column, are stored in one of several prescribed binary formats, rather than coded into ASCII. For each field there is a repeat count, which is one for a single entry and larger if the entry is to be a vector. This main table may be followed by additional data.

Appendixes to the paper proposed three additional conventions: one to handle fields where the length of an array might vary, one to allow the vector in a field to be interpreted as a multidimensional array, and one for delimiting arrays of strings. While the Working Group endorsement of binary tables did not include these conventions, the rules that were adopted reserved a number of keywords and values specifically intended for use by these conventions. These conventions are discussed in section 5.2.

In a binary table, each row consists of a series of entries in formats specified by the header keywords. The header describes the size and data type of each entry, and, optionally, provides a label, units, and the representation for an undefined value where applicable. The length and field structure of all rows of the main table in a particular binary table extension must be the same.

Every row in a particular binary table contains the same number of entries, although the number can vary from one binary table extension to the next in a FITS file. The header is a standard FITS extension header. For each table entry it specifies

1. The size and data type of the entry.
2. A label (optional).
3. The units (optional).
4. The representation of an undefined value (optional).

An entry may be omitted from the table but still defined in the header by using a zero repeat count in the value of the `TFORM`*n* keyword. Examples 4, 6, and 7 in Appendix A illustrate binary tables.

3.6.1 Required Keywords for Binary Table Extension Headers

The extension follows the rules for required keywords stated in section 3.3.2. The first eight keywords must appear consecutively in the following order:

1. `XTENSION` (character) has the value `'BINTABLE'` for binary table extensions
2. `BITPIX` (integer) must have a value of 8. A binary table is an array of bytes.

3. `NAXIS` (integer) must have a value of 2 for binary table extensions. As with ASCII tables, the axes are the rows and columns of the table.

4. `NAXIS1` (integer) has a value equal to the number of 8-bit bytes in a table row.

5. `NAXIS2` (integer) has a value equal to the number of rows in the table. Values of 0 (no table) and 1 (one row), are allowed as well as larger values.

6. `PCOUNT` (integer) is equal to the number of bytes in the binary table extension data following the main table. Note that this meaning is significantly different from that in the rules for random groups (section 3.2). This usage was designed for the variable length convention discussed in section 5.2.1. While conflicting uses would not be against the formal FITS rules, they would create confusion and should be avoided. Non-conflicting uses may be developed for other conventions.

7. `GCOUNT` (integer) has the value of 1 for binary table extensions.

   The total number of bytes is PCOUNT + NAXIS1$\times$NAXIS2.

8. `TFIELDS` (integer) has a value equal to the number of entries (fields) in each row of the table.

Also required, between the `TFIELDS` and `END` keywords, are `TFORM`n keywords for n=1, ...,TFIELDS.

`TFORM`n (character) has a value specifying the width and data type of field n and is of the form rT, where T is the field type and r is the repeat count that tells how many values of type T the field contains. The value of r gives the number of members in the array that table item contains. If the repeat count is absent, it is assumed to be 1. A repeat count of zero is permitted. Additional characters may follow the data type code. While these characters are not defined in the formal `BINTABLE` rules, the variable length array and string array conventions make particular interpretations, and conflicting uses should be avoided, to prevent confusion. The following values of T are permitted:

L Logical value.

X Bit array.

A Character. The substring array convention (section 5.2.2) uses the characters `:SSTR` immediately following the rA to indicate that the convention is being used to describe the substrings. This usage implies that a colon and different standard character string following the rA can be used as an indicator of other conventions for describing character fields. While the formal FITS rules do not prohibit other uses, such conflicting usage could be confusing and is to be avoided.

B Unsigned 8-bit integer or byte.

I Signed 16-bit integer.

J Signed 32-bit integer.

E Thirty-two bit IEEE floating point.

D Sixty-four bit IEEE floating point.

C Complex pair of single precision floating point values.

M Complex pair of double precision floating point values.

P Sixty-four bit descriptor of variable length arrays (pointer). The only allowed repeat counts are 0 and 1. This data type was designed to contain information on the location of data in variable length arrays (section 5.2.1). When used to indicate a variable length array, the value takes the form `rPt(maxelem)`, where `r` is the repeat count, `t` is any of the data types discussed here except P, and `maxelem` is an integer. While the binary table rules do not specifically forbid the use of this data type for other purposes, any use that conflicts with the variable length convention would cause confusion and is to be avoided.

For example, `TFORM4  = '16E      '` means that the fourth field of the table will consist of 16 32-bit floating point values. The repeat count represents a major enhancement over ASCII tables, where a field can contain only one value. The number of bytes in a row is the sum of the number of bytes in the individual fields and must equal the value of `NAXIS1`. Note that this rule differs from ASCII tables, which allow the value of `NAXIS1` to be greater than the sum of the `TFORM`n.

As with ASCII tables, to distinguish -0 from $+0$ in integer format, for example in degrees of declination, the sign field should be declared as a separate character field. This separation is unecessary for floating point.

`END` must always be the last keyword. The remainder of the last header record must be filled with ASCII blanks.

Note that while the rules for generalized extensions do not specify where the `PCOUNT` and `GCOUNT` keywords are located after the last `NAXISn` keyword, the rules for binary table extensions require that they appear between the `NAXIS2` and `TFIELDS` keywords.

3.6.2 Reserved Keywords for Binary Table Extension Header

These keywords are optional in a binary table extension but may be used only with the meanings specified below. They may appear in any order between the `TFIELDS` and `END` keywords. Note that some of these keywords are the same as those for an ASCII table extension. The reason is that they are used for a binary table in a way analogous to their use for an ASCII table. However, the allowed values and their meaning may differ somewhat from those for an ASCII table, as the rows of an ASCII table are composed of characters and those of a binary table are composed of bytes.

- `TTYPE`n (character) has a value giving the label or heading for field n. While the rules do not further prescribe the value, following the recommendations for the `TTYPE`n keyword of ASCII tables is a good practice: using letters, digits, and underscore but not hyphen; also, string comparisons involving the values should be case insensitive. HEASARC has made this practice one of their internal standards (section 5.6.1.1).
- `TUNIT`n (character) has a value giving the physical units of field n. The rules should follow the prescriptions given in section 3.1.1.4.
- `TSCAL`n (floating) has a value providing a scale factor for use in converting stored table values for field n to physical values. The default value is 1.
- `TZERO`n (floating) has a value providing the offset for field n. The default value is 0.

$$\text{(Physical\_value)} = \text{(Stored\_value)} \times \text{TSCAL}n + \text{TZERO}n. \quad (3.12)$$

As is the case for arrays, care should be taken to avoid overflows when scaling floating point numbers.

For L, X, and A format fields, the `TSCAL`*n* and `TZERO`*n* keywords have no meaning and should not be used. The meaning has not been formally defined for P format fields, but the general understanding is that the scaling should apply to the heap data pointed to by the array descriptors.

- `TNULL`*n* (integer) has the value that signifies an undefined value for the integer data types B, I, and J. It should not be used if the value of the corresponding `TFORM`*n* specifies any other data type. Null values for other data types are discussed in section 3.6.3.

- `TDISP`*n* (character) has a value giving the Fortran 90 format recommended for display of the contents of field n. (If Fortran 90 formats are not available to the software printing a table, FORTRAN-77 formats may be used instead.) All entries in a single field are displayed with the same format. If the field data are scaled, the physical values, derived by applying the scaling transformation, are displayed. For bit and byte arrays, each byte is considered to be an unsigned integer for purposes of display. Characters and logical values may be null (zero byte) terminated. The following formats are allowed:

  Lw Logical

  Aw Character

  Iw.m Integer

  Bw.m Binary, integers only

  Ow.m Octal, integers only

  Zw.m Hexadecimal, integers only,

  Fw.d Single precision real, no exponent

  Ew.dEe Single precision real, exponential notation

  ENw.d Engineering format - single precision real, exponential notation with exponent a multiple of 3

  ESw.d Scientific format - single precision real, exponential notation with exponent a multiple of 3, nonzero leading digit (unless value is zero)

  Gw.dEe General - appears as F format if significance will not be lost; otherwise appears as E

  Dw.dEe Double precision real, exponential notation

  In these formats, w is the number of characters in the displayed values, m is the minimum number of digits (leading zeroes may be required), d is the number of digits following the decimal point, and e is the number of digits in the exponent of an exponential form. Usage of this keyword in some ways parallels that of the `TFORM`n keyword of ASCII tables, in that it provides a formatted value for the number. However, the format given by the `TFORM`n keyword in an ASCII table describes the format of the number in the FITS file, but the format given by the `TDISP`n keyword of a binary table is different from that of the number in the file.

The following keywords are reserved for proposed binary table conventions:

- `TDIM`n (character) is used by the multidimensional array convention (section 5.2.3). For that convention, it has a value giving the number of dimensions of field n in the table, when those dimensions are two or more. The value is of form `'(i,j,k,...)'`, where i,j,k,... are the dimensions of the array stored in field n. This size must be consistent with the repeat count specified by the value of `TFORM`n.
- `THEAP` (integer) is used by the variable length array convention (section 5.2.1). For that convention, its value gives the location of the start of the heap used to store variable length arrays. The value is equal to the number of bytes of extension data preceding the start of the heap, including the main table and any gap between the main table and the heap.

All keywords reserved under the generalized extensions agreement (section 3.3.2) apply to binary tables.

3.6.3 Binary Table Extension Data Records

The data records in a binary table consist of the main data table and possibly additional data after the main table. The `BINTABLE` main table logical data records begin with the record immediately following the last header record. Each record contains 2880 8-bit bytes in the order defined by the header. Table rows do not necessarily begin at the beginning of a new record. The size of the main table is the product of the values of the `NAXIS1` and `NAXIS2` keywords. The additional data begin immediately after the end of the main table; they do not start at the next record. The last record of the data should be zero filled past the end of the valid data, with all bits cleared. If there are no additional data after the main table, this record will be the last record of the main table; if there are additional data, this record will be the last record of the additional data.

The data types are defined as follows:

L A logical value consists of an ASCII "T" indicating true or "F" indicating false. A null character (zero byte) signifies an invalid value.

X A bit array starts in the most significant bit of the byte, and the subsequent bits are in the order of decreasing significance in the byte. A bit array field in a binary table consists of an integral number of bytes with those bits that follow the array set to zero. No specific null value is prescribed for bit arrays, but the following three conventions are suggested:

- Designate one bit of the array to be a validity bit.
- Add a type L field to the table to indicate the validity of the bit array.
- Add a second bit array which contains a validity bit for each of the bits of the original array.

Use of any of these conventions will be a decision of an individual project or particular group of FITS users. Do not expect that general software to read FITS will necessarily be able to interpret them.

B An unsigned 8-bit integer has the bits in decreasing order of significance. By applying scaling, this field may be used to store quantities whose physical values are signed.

I A 16-bit integer is a twos-complement integer with the bits in decreasing order of significance.

J A 32-bit integer is a twos-complement integer with the bits in decreasing order of significance.

A Character strings consist of 8-bit ASCII characters in their natural order. An ASCII NULL (hexadecimal 00) character may be used to terminate a character string before the length specified by the repeat count is reached. Strings occupying the full length of the field are not NULL terminated. An ASCII NULL as the first character signifies a NULL (undefined) string. The printable ASCII characters, that is, those in the range hexadecimal 20-7E, and the ASCII NULL after the last valid character are the only ones permitted.

E Single precision floating point values are in IEEE 32-bit precision format, as described in section 3.1.2.3.

D Double precision floating point values are in IEEE 64-bit precision format, as described in section 3.1.2.3.

C A complex value is composed of a pair of IEEE 32-bit floating point values: the first is the real part and the second the imaginary part.

M A double precision complex value is composed of a pair of IEEE 64-bit floating point values: the first is the real part and the second the imaginary part.

P An array descriptor consists of two 32-bit twos-complement integer values.

For the floating point types—E, D, C, and M—the IEEE NaN values represent undefined or null values; a value with all bits set is recognized as a NaN. All IEEE special values are recognized: infinity, NaN, and denormalized numbers. If either the real or imaginary part of a complex value contains a NaN, the entire complex value is regarded as invalid.

Alignment may be a problem when binary tables are read. Suppose the sequence (1I, 1E) is read directly from the I/O buffer. Some machines may require that the number of bytes between the start of this sequence and the start of a 4-byte floating point number in it be evenly divisible by 4. In that case, the floating point number will not begin at a proper location, and a data alignment error will result. If alignment is important, the data should be copied from the I/O buffer into an aligned buffer before they are read.

3.7 Reading FITS Format

It is more difficult to specify and give examples of the minimum requirements to read FITS formats because reading FITS means a translation of the FITS data set into the internal format of a data analysis package. Each package has its own unique requirements. When developing software to read FITS format, allow for the flexibility of FITS. Don't assume that files will always follow practices that are recommended but not required. For example, don't expect that values of optional keywords will use the recommended, but not mandatory, fixed format; be prepared to accept all formats consistent with FORTRAN-77 list-directed read. Also, assume that keywords will appear in a particular order only when this order is specified by the rules of FITS. Because *values* of keywords may contain lower case letters, comparisons should not be case sensitive.

A FITS file may contain records after the end of the FITS primary HDU and the extensions. Such special records can be recognized by the fact that they begin where FITS would begin an extension, at the end of the previous extension or the primary HDU, but the first 8 bytes do not contain the string "`XTENSION`." While the FITS rules do not forbid the string "`SIMPLE  `", its presence would be confusing and should be avoided. These records must have the standard FITS length of 23040 bits, to make it easy for a FITS reader to go through them in a search for the end of file. A generalized FITS reader should be prepared for them.

3.8 FITS Files and Physical Media

FITS is defined as a logical structure, independent of the medium on which the data appear. FITS files are regarded as bit/byte streams with a 2880-byte logical record length. Data written in FITS format can, for example, be on magnetic tape, magnetic disk, optical disk, or simply on-line. However, the reader of a FITS file must know its physical structure in order to unpack it.

Historically, magnetic tape was the original storage medium for FITS formats. The initial conventions for 1/2-inch 9-track magnetic tape were established in the original FITS paper (Wells, Greisen, and Harten 1981) and modifications to these conventions were presented at the same time as the rules for extensions (Grosbøl *et al.* 1988). The IAUFWG has since endorsed an additional set of formal rules to relate physical and logical structure for other media. Each medium has its own particular rules. The complete rules are as follows:

- For 1/2-inch 9-track magnetic tape, physical blocks may contain one to ten logical records. Unlabeled tapes are strongly recommended; however, tapes with ANSI labels are permitted. Data reduction and analysis packages reading labeled tapes must count the tape label as the first file.
- For fixed block sequential media with a physical block size that is not an integral multiple or fraction of 2880 (for example, 512), the 2880-byte logical records are not tied to the physical block size but are written

independently, over multiple blocks if necessary. The end of the last logical record of the FITS data set will probably not correspond to the end of a physical block. A short logical record extending to the end of the physical block will follow. Readers should regard a record of less than 2880 bytes as the equivalent of an end-of-file mark. Examples of media covered by these rules are optical disks accessed as a sequential set of records, QIC format 1/4-inch cartridge tapes, and local area networks.

- For bitstream devices, FITS files are written with a blocking factor of one, i.e., with a physical block size of 23040 bits, the same as the logical record length.
- For variable block length sequential media, such as DDS/DAT 4-mm and 8-mm Exabyte cartridge tape, physical blocks may contain an integral number of logical records from one to ten, the same rule as for 1/2-inch 9-track magnetic tape. In the case where the data system is able to generate only fixed blocks, the fixed block convention is used.

Except where a logical record of fewer than 23040 bits serves as an end-of-file indicator, a file should be terminated by the structure for the given medium that corresponds to the tape end-of-file mark. If more than one FITS file appears on a physical structure, the end-of-file indicator appropriate to the medium should separate the files, appearing immediately before the start of the primary header of the next file.

In the original FITS specification, the physical block size was set equal to the logical record length for simplicity. As time passed, however, it became clear that a significant number of the major institutes producing data regarded this physical block size as an inefficient use of tape. On a 6250-bpi tape, only 55% would contain data and 45% would be inter-record gaps; consequently, at the same time as the rules for extensions were developed, it was agreed that physical blocks on 1/2-inch 9-track magnetic tape could contain one to ten logical records. Blocks containing more than one record were called long blocks. At that time, because the multirecord blocks were new and might not be expected by some FITS readers, the `BLOCKED` keyword was introduced as an indicator that there *might* be more than one logical record per physical block on a tapes. However, usage was not universal; therefore, absence of the `BLOCKED` keyword therefore did not guarantee single record blocks, and by its definition, its presence did not guarantee multiple-record blocks. It could thus not be used effectively as intended. As the multiple record block became established as standard, and with the rise of the concept of FITS as a purely logical structure, the need for a BLOCKED keyword has disappeared. Information describing how a tape is blocked should be sent with it, and users should ask for blocking information when receiving tapes. The keyword itself should no longer be used as a blocking indicator, but it remains formally reserved for its original purpose and should also not be used for any other.

(The rules for the BLOCKED keyword also were in conflict with other rules in FITS. This keyword was required to be in the first logical record; however, if the value of NAXIS was greater than 33—or 32 when EXTEND was present—it could not be put in the first record without violating the rule that the NAXISn keywords must be consecutive.)

Conventions for writing a FITS file on more than one physical unit (tape, disk) are only now being developed. In the original FITS paper, Wells, Greisen, and Harten (1979) suggested a method for handling multivolume unlabeled tape, but this proposal has not been put into practice widely. Many current programs that read FITS files will be unable to handle a multiple-unit file. The grouping proposal, discused in section 5.3, uses tables to associate physically separated FITS HDUs into a logical unit.

Section 4

World Coordinate Systems

While every point in the FITS data array can be located in the cooordinate system determined by the array axes, scientific interpretation requires knowledge of the physical or world coordinates corresponding to the index points. For example, in an array that represents a spectrum, the relation between wavelength and index number on the wavelength axis must be known. The original FITS papers reserved several keywords — CTYPEn, CRPIXn, CRVALn, CDELTn, and CROTAn — for use in specifying this relation. While the original paper (Wells, Greisen, and Harten 1981) defined these keywords only for linear transformations, they can be also be adapted for non-linear transformations using the definitions in Section 3.1.1.2. However, for a non-linear transformation, some keywords in addition to those reserved will be required to provide a full description.

To create FITS arrays representing the images of objects in the sky requires projection of the celestial sphere onto the array plane. The original FITS papers did not specify how this process was to be carried out. The first approach to this problem, implemented in the Astronomical Image Processing System (AIPS) and described in AIPS Memo 27 (Greisen 1983), has been widely used, but no conventions have been formally endorsed by the IAU FITS Working Group. E. Greisen and M. Calabretta (1996; hereafter GC) have expanded the AIPS approach into a comprehensive proposal for conventions to be used in projecting spherical coordinates onto a two-dimensional plane. This proposal has evolved in response to considerable community discussion and is intended for formal submission to the IAUFWG. Their proposal discusses a number of other World Coordinates issues as well.

The remainder of this section (4) will describe the GC proposal, some other proposed conventions that are gaining acceptance, and additional FITS community practices. Following these conventions in creating a data set will make it easier for many others to read and understand the data. In particular, designers of FITS files that will be distributed outside the originating site should not adopt conventions that conflict with those described in this *Guide* without consulting with other members of the FITS community, particularly the anticipated readers of the file. In particular, to prevent confusion, avoid using the keywords and notation of the conventions described here with different meanings. Because these conventions have not been formally approved as part of standard FITS, data sets that use other conventions are not out of conformance. However, these conventions are widely used and have the best software support; creators of FITS data that use different conventions will probably need to supply supporting software to their users.

4.1 Indexes and Physical Coordinates

When the data matrix represents a digital image, transformation between the data matrix and the physical picture requires knowledge of where in the pixel — center or corner — the data point is. Historically, astronomers have generally assumed that the index point in a FITS file represents the center of a pixel. This interpretation is endorsed by GC. It differs from the common practice in computer graphics of treating the center of a pixel as a half-integral point. GC note that the pixel in a FITS file is commonly regarded as a volume element in physical space, which might be viewed from different perspectives via transposition and rotation. Under such operations, only the center of an element remains invariant. Pending adoption of a standard convention by the astronomical community, FITS writers should use appropriate comments in the comment field of the card image or the `COMMENT` keyword to inform readers of the file which convention is being used. Once the community has accepted a convention, a single comment noting that the convention is being used will be sufficient.

The relation between array order and position in the displayed image, for example whether the first pixel is at the top or bottom, is also a matter of convention. GC recommend that FITS writers order pixels starting in the lower left hand corner of the image, with the first axis increasing to the right, as in the rectangular coordinate x-axis, and the second increasing upward (the y-axis). Readers may choose to use a different display, as might be done when picturing how a field mapped visually would look through a telescope. Use of this convention does not replace the description of the coordinate axes using `CRVAL`n and the other keywords.

Although the original description of the `CRPIX`n, `CRVAL`n, and `CDELT`n keywords was in the context of linear transformations, these keywords can be used to define non-linear transformations. The key point is that in such a case `CDELT`n must represent the rate of change in the physical coordinate per unit change in counting index *at the reference point given by* `CRPIX`n. For a non-linear scale, the ratio between change in the physical coordinate and in the counting index could be different at other pixels. Conventions for such transformations have not been established. Development or use of any such convention, including the definition of any new keywords, should be in cooperation with the others in the FITS community who need such conventions.

4.2 Proposed Conventions

Although they have not been formally endorsed by the IAUFWG, the AIPS conventions (Greisen 1983) have been widely used, in particular for distribution of IRAS infrared and Einstein X-ray imagery. A detailed discussion and analysis appears in AIPS memo 27 (Greisen 1983). Hanisch and Wells (1988; hereafter HW) proposed adopting these conventions as standard within FITS. They also discussed a number of other conventions recommended at a meeting held in Charlottesville, Virginia, in January 1988, sponsored by NASA Code EZ. The GC proposal marks further movement in the directions begun at Charlottesville. The discussion in the remainder of section 4 of the *User's Guide* is based primarily upon those three documents, especially GC. However, it does not necessarily reflect the views of all the cited authors on all points.

4.2.1 Improved Axis Descriptions

The original FITS Paper (Wells, Greisen and Harten 1981), assumed that the units of the axes would be the basic SI units appropriate to the `CTYPE`n axes. But the choice of units may not be obvious. GC propose that a new keyword be reserved for such cases:

`CUNIT`n (floating) is the units in which both `CRVAL`n and `CDELT`n for axis n are expressed.

Units identified using this keyword should be chosen using the principles defined in section 3.1.1.4 and must conform to all the requirements in that section.

An axis may be described by more than one label. For example, a spectrum may be described by frequency, wavelength, and velocity, only one of which can be linear. GC propose reserving the following additional keywords:

- `C`m`YPE`n (floating) is label m for axis n.
- `C`m`NIT`n (floating) gives the units for axis n when label m is used.
- `C`m`PIX`n (floating) is the reference point of axis n when label m is used.
- `C`m`VAL`n (floating) is the coordinate value of the reference point on axis n when label m is used.

The values of m may run from 2–9; the original `CTYPE`n, `CRPIX`n, `CRVAL`n, and `CDELT`n keywords are used for $m = 1$. Note that the reference pixel need not be the same for all descriptions of an axis. The values of a given pixel in the different units must be consistent, e.g., 1420.4 MHz and 21.1 cm. There are no defaults assumed for $m \geq 2$; all of these keywords should be specified.

4.2.2 Sky Images

There are five steps in converting a digital array into an image on the sky:

1. (optional) Transform from actual image pixel numbers to those that would be observed by an ideal instrument.
2. Transform from the origin and axes of the instrumental coordinate matrix system to the origin and axes of a set of projected sky coordinates.
3. Convert the scale along each axis from pixel to physical values, still in the projection plane.
4. Determine the projection of the sky onto the coordinate plane of the digital array and deproject from the array plane onto the sky.
5. Transform from sky coordinates derived by deprojection to a standard celestial coordinate system.

4.2.2.1 Pixel Regularization

The first step above is for the case where the array positions of the coordinate points differ from those that would result from the projection from the sky. Examples are distortions of astrometric plates caused by imperfect optics or emulsion shrinkage and the effects of ionospheric refractions on radio maps. GC describe how to carry out this process through the use of a Pixel Regularization Image. This step involves only the correction for small distortions; all aspects of the coordinate transformation between the rectangular grid and physical space should be incorporated into the subsequent steps.

4.2.2.2 Transforming to Projected Sky Coordinate

The coordinate system of the FITS file data array, will, in general, be determined by the geometry of the instrumentation; consequently, the direction of the axes in the FITS file will need to be transformed to a coordinate system more natural to the physical system being measured. The initial FITS paper (Wells, Greisen, and Harten 1981) specifies use of the `CROTA`n keywords for describing how the axes of the FITS array are rotated to yield physical coordinates but does not define precisely how they are to be used. AIPS uses the `CROTA2` keyword to represent the angle, in decimal degrees, of the declination or latitude axis with respect to the 2 axis of the data array, measured in the counterclockwise direction. AIPS does not use the `CROTA1` keyword and assumes a default value of zero for the `CROTA2` keyword. The `CROTA`n keywords are necessary only when the physical coordinates are rotated relative to the array coordinates. These AIPS conventions were endorsed at the Charlottesville meeting, have been widely used, and will be generally understood. However, some individual FITS files may define the `CROTA`n differently. Check the comments.

The `CROTA`n keywords cannot be applied in a straightforward way to more than two dimensions or to cases where the axes are not orthogonal. GC define the more general pixel coordinate (PC) matrix P to transform between the FITS array axes f_j and axes in the direction of the physical coordinate system but on the array scale a_i,

$$\Delta a_i = \sum_j P_{ij} \Delta f_j \tag{4.1}$$

where the Δ signify that positions are with respect to the reference point. Keywords are used to specify the elements of the PC matrix:

`PC`$iiijjj$ (floating) is an multi-indexed keyword giving the value of the pixel coordinate matrix element P_{ij} in equation 4.1.

The PC matrix technique permits considerable flexibility in transforming between axes, including axes that are not orthogonal. Some of this flexibility may be better left unused. In particular, it is possible to transpose the axes when going from the pixel axes $(i, j, \ldots)$ to the physical axes $(x, y, \ldots)$. Suppose the first two axes are transposed, so that axis 2 in physical space corresponds to axis 1 in pixel space and vice versa. The most rapidly varying axis is axis 1 (i in pixel space), described by the values of NAXIS1 and CRPIX1. The corresponding physical axis is the one (y) described by CTYPE2, CRVAL2, and CDELT2. CTYPE1 describes the first axis in physical space, which is now not the most rapidly varying. If these concepts take some time to think through here, imagine trying to understand them in a FITS file. Such complications can be avoided if the tranformation is defined such that the diagonal terms of the PC matrix predominate. Another potential application of the PC matrix that is best unused is the ability to reverse the direction of the axis. GC strongly recommend that the elements of the PC matrix should be positive. Use the CDELTn keywords to reverse axis directions.

Because the AIPS convention has been widely used, new software that uses the PC matrix will still need to interpret files using the AIPS conventions. Also, until FITS reading software routinely includes support for the PC matrix, FITS files where the PC matrix represents a rotation equivalent to that of the old CROTAn method should also include the CROTAn keyword as well as the PC matrix. If θ is the latitude angle represented by CROTAn in the AIPS convention, and δ_n is the value of CDELTn, then

$$P_{11} = \cos\theta \tag{4.2}$$
$$P_{12} = -(\delta_2/\delta_1)\sin\theta \tag{4.3}$$
$$P_{21} = (\delta_1/\delta_2)\sin\theta \tag{4.4}$$
$$P_{22} = \cos\theta \tag{4.5}$$

It is always possible to express the CROTAn rotation in the PC matrix form. However, not all of the more general rotations described by the PC matrix can be described using CROTAn. When the coordinate transformation follows the AIPS convention, the $\mathsf{P}(\theta)$ relation can be inverted to yield two separate and equivalent expressions for θ:

$$\theta = \tan^{-1}\{-[(\delta_1/\delta_2)(P_{12}/P_{22})]\} \tag{4.6}$$
$$= \tan^{-1}[(\delta_2/\delta_1)(P_{21}/P_{11})] \tag{4.7}$$

For a coordinate transformation following the AIPS convention, both expressions give the same value. For more general tranformations, both should be computed. If the difference between the two values of θ obtained is within the expected precision, use the average (to reduce the round-off errors). If the difference is larger, then the coordinate axes are not orthogonal and the CROTAn description cannot be used (HW).

4.2.2.3 From Pixel to Physical Values

Application of the PC matrix yields a coordinate system in the direction of the physical axes but in pixel units. Conversion to physical units is performed by applying the scaling specfied by the `CDELT`n.

In some past applications of this process, steps (2) and (3) have been combined into a single step, using a matrix that is the product of the `CDELT`n, which may be thought of as a diagonal matrix, and the PC matrix. This matrix, called the coordinate description (CD) matrix, described the tranformation from the array coordinates in pixel units to the physical coordinate in physical units. It was represented using keywords of the form `CDi_j`, by the Space Telescope Science Data Analysis System/Image Reduction and Analysis Facility (STSDAS/IRAF), or `CDiiijjj`, proposed by HW. This method was described in the previous version (3.1) of this *Guide*. FITS readers should be prepared to encounter it in some existing data sets.

4.2.2.4 Deprojection

The plane coordinates in physical units represent a projection of the celestial sphere. This projection is not unique, and different projections may be useful for different purposes. Maps of the earth, for example may be Mercator, equal-area, or some other projection, depending on the intended use. In this step, the projection is reversed, and the plane coordinates are transformed back to the projected sphere.

AIPS memo 27 (Greisen 1983) identified four common choices, which are discussed here because of their wide use. For three of these four projections, the choice for the tangent point, the point common to the celestial sphere and the plane on which it is projected, is arbitrary. AIPS adopted the convention that the reference pixel, given by the values of `CRPIX`n, is at this tangent point. If the tangent point of the projection is outside the image, then the values of the `CRPIX`n will be outside the data array. Note that the values of the `CRVAL`n do not change between images in different regions of the same projection.

In the TAN (tangent) or gnomic geometry, the projection of a point of the celestial sphere onto the plane is found by extending the line from the center of the sphere to the point on the tangent plane. This projection is common in optical astronomy. In the SIN (sine) or orthographic geometry, common in radio aperture synthesis, the projection point is on the perpendicular between the point on the sphere and the surface of the tangent plane. In the ARC geometry, the projection point is on an arc passing through the point on the sphere and the tangent plane, with the center of the arc at the reference point. This projection preserves angular distances. It is the natural projection of Schmidt camera imagery, such as the Palomar Sky Survey, and is also used in single-dish radio telescope mapping. GC note that the NCP geometry discussed in the AIPS memo, a projection onto a plane perpendicular to the North Celestial Pole used by the Westerbork Synthesis Radio Telescope and a few other east-west radio interferometers, can be treated as a special case of the orthographic (SIN) projection. The SIN and TAN projections are special cases of the *perspective zenithal projections*, which are generated from a point and carried through from the unit sphere to the plane of projection.

The proposal adopts the notation that has been used in AIPS, using the last four characters of the value field of the `CTYPE`n keywords to identify the projection. Table 4.1 shows the codes for the projections described above.

Table 4.1: Common Projections

`-TAN`	tangent
`-SIN`	sine
`-ARC`	arc

GC define 25 projection codes in all, including the spherical cube projection used by the COBE project (`CSC`). Many standard projections are special cases of these 25. The `-NCP` designation of the original AIPS memo is no longer used. For each projection they provide the transformation equations and a diagram.

The following indexed keyword is reserved under this proposal for defining projections:

`PROJP`n (floating) is the nth projection parameter. Projection parameters are additional quantities that are must be specified in order to define the projection completely. The specfic meaning of the keyword or keywords is defined separately for each projection. Many projections do not need these additional parameters.

GC call the deprojected coordinate system that results from this step the *native coordinates*.

4.2.2.5 Conversion to Standard Celestial Coordinates

In the original AIPS convention, the native system was required to be locally parallel to the standard celestial coordinate system at the tangent point. GC relax this requirement. The axes of the native sky coordinate system derived by deprojection need not be in the same direction as those of the standard celestial coordinate system. GC provide equations for the transformation. The following keywords are reserved:

- `LONGPOLE`*n* (floating) for *long*itude of the *pole*, is the longitude of the North Pole of the standard coordinate system in the native coordinates.
- `LATPOLE`*n* (floating) is used in special cases when two values for declination of the north pole of the native system as measured in the standard system are consistent with the projection and the value of `LONGPOLE`; it determines which is used.

`LONGPOLE` is understood to have a default value of 0^o or 180^o where possible. The choice depends on the latitude of the reference point in the standard system and in the native system; if the latitude in the standard system is greater, `LONGPOLE` is 0^o; if the latitude in the native system is greater, it is 180^o. With this convention, celestial latitude and native latitude increase in the same direction at the tangent point. A detailed discussion of the transformations and the application of the reserved keywords can be found in GC and is beyond the scope of this *Guide*.

GC adopt the AIPS convention of using the first four characters of the value field of the `CTYPE`*n* keywords to identify the standard system, as shown in Table 4.2.

Table 4.2: Identification of Sky Coordinate Systems

`RA--`,	`DEC-`	equatorial coordinates (α, δ)
`GLON`,	`GLAT`	galactic coordinates (ℓ^{II}, b^{II})
`ELON`,	`ELAT`	ecliptic coordinates (λ, β)
`SLON`,	`SLAT`	supergalactic coordinates
`HLON`,	`HLAT`	helioecliptic coordinates

Galactic coordinates as defined above are in the "new" system adopted by the IAU in 1950, where the origin, 0^o latitude and 0^o longitude, is in the direction of the galactic center. Identifications for older systems or future systems should be in the form (x`LAT`, x`LON`) but not one of the designations used in Table 4.2. Ecliptic coordinates refer to the mean ecliptic and equinox of the date of observation, in the post-IAU 1976 system.

The values of the `CTYPE`n provide both the world coordinate system and the projection used. For example, if the data matrix contained a sky map in right ascension and declination projected on the plane using the tangent projection, the `CTYPE`n values would be

```
CTYPE1  = 'RA---TAN'
CTYPE2  = 'DEC--TAN'
```

The coordinates must be logically consistent; for example, `'RA---TAN'` should not be paired with `'GLON-ARC'`.

4.3 Coordinate Keywords

There are a number of standard equatorial coordinate systems. HW proposed the following keywords to supplement the reserved keywords discussed in section 3.1.1.1.

- `RADECSYS` (character) standing for *RA/Dec sys*tem specifies the reference frame for the equatorial coordinate system. It should be used when the `CTYPE`n values are an `RA--xxxx`/`DEC-xxxx` pair. Table 4.3 shows the four values defined.

Table 4.3: Reference Frames for Equatorial Coordinate Systems

`'FK4     '`	mean place old (before the 1976 IAU) system
`'FK4-NO-E'`	mean place old system but without e-terms
`'FK5     '`	mean place new (after the 1976 IAU) system
`'GAPPT   '`	geocentric, apparent place, after the 1976 IAU system

The default value is `'FK4     '`. Under the FK4, the date of equinox (value of the `EQUINOX` keyword) is interpreted as a Besselian epoch and under FK5 as a Julian epoch. If there is no `EQUINOX` keyword (or `EPOCH` for old FITS files), the equinox is assumed to be 1950.0 for FK4 and 2000.0 for FK5.

- `MJD-OBS` (floating) specifies the time of observation in the form of a Modified Julian Date (JD - 2400000.5), in the International Atomic Time time scale. Whether this point refers to the starting, midpoint, or ending time is not defined. Use comments to say. Use of this keyword permits greater precision than `DATE-OBS`. Because the FK4 reference frame is not inertial—there is a small but significant rotation relative to distant objects—the epoch or time of observation is required to specify when the mean place was correct.

4.4 Current Status

In support of this proposal, Calabretta has developed the WCSlib software package. It is available under a GNU Public License through the NRAO FITS archive described in Section 6.2.

The Greisen and Calabretta (1996) proposal is still under discussion in the FITS community and may undergo further changes. The following aspects of the proposal have wide acceptance already and should be generally understood by FITS readers:

- Index point at the *center* of a pixel
- Transformation matrix (future applications are expected to follow the PC notation and usage, although there are many existing files with the CD notation and combined rotation-scaling usage)
- `SIN`, `TAN`, and `ARC` projections, with the standard coordinates and the native coordinates locally parallel at the tangent point, as in the AIPS convention.
- The conventions for including the name of the coordinate system and projection in the value of `CTYPE`n.

- The coordinate keywords described in Section 4.3.

Trial implementations and discussions will continue before a standard for world coordinates becomes part of the overall FITS standard. Among the questions still under discussion are the following:

- Is there a way to use this formalism for spectra, time series, and unequally spaced data?
- Is the notation of section 4.2.1 the best choice for multiple WCS used for the same image?
- Can the pixel correction image method be implemented for complicated FITS files, for example, those with many images, each with its own correction?
- Should the parallel use of the `CROTA`n notation continue, and, if so, how long?
- Can agreement be reached on a more specific algorithm for choosing the `CTYPE`n values to be used to identify future coordinate systems and projections?
- Is the description of the common projections (e. g., `SIN`,`TAN`) now too complicated?
- Should all of the projections be included if this proposal is adopted as the standard, or should only the basic projections be included, with the rest as appendixes?

Keep up with the current discussion in the forums described in section 6. Make certain that the community has reached agreement in the areas described above before implementing any of those features in archival data sets.

Section 5

Advanced FITS

5.1 Registered Extension Type Names

Extension type names must be unique; a particular type name may refer to only one structure. In order to avoid duplication, all extension type names must be registered with the IAU FITS Working Group, even if the extension type is being used by only a single organization internally. To register a FITS extension name, contact either the FITS Support Office at `fits@nssdca.gsfc.nasa.gov` or the chair of the IAUFWG at `dwells@nrao.edu`. Provide a short description and justification with the request for registration. Reserve extension type names when their use is planned, even though details of the structure have not been worked out.

Table 5.1 lists the FITS extension type names and their registered status with the IAUFWG at the time this *Guide* is being written. Also included in the table are the organization with current responsibility for the extension and its development and, when available, a reference describing the extension. Table 5.2 explains the status codes in Table 5.1.

The up-to-date official listing of extensions registered with the IAUFWG is available on-line from the FITS Support Office at `http://ssdoo.gsfc.nasa.gov/astro/fits/xtension.html` or `ftp://nssdc.gsfc.nasa.gov/pub/fits/xtension.lis`. This listing is revised as information is received from the IAUFWG about the registration of new type names or progress of a particular extension type through the approval process.

Type Name	Status	Reference	Sponsor	Comments
`'A3DTABLE'`	L	NRAO(1990)	NRAO	Prototype binary table design supported in AIPS, superseded by `BINTABLE`, which supports all `A3DTABLE` features.
`'BINTABLE'`	S	Cotton, Tody, and Pence(1995)	IAU	Binary Tables. Available at FITS Archives in files `documents/standards/bintable*` for 1995-Feb-06 Note: only main document, excluding appendixes.
`'COMPRESS'`	R	Warnock *et. al.* (1990)	GSFC A/WWW	Suggested name by A. Warnock Preliminary proposal
`'DUMP␣␣␣␣'`	R	none	none	Suggested name for binary dumps. No full proposal submitted.
`'FILEMARK'`	R	none	NRAO	Intended for structure to represent equivalent of tape mark on other media. No full proposal submitted.
`'IMAGE␣␣␣'`	S	Ponz, Thompson, and Muñoz (1994)	IAU	Image extension, contains an array of one or more dimensions.
`'IUEIMAGE'`	L	Muñoz (1989)	IUE	Prototype matrix extension used for archiving IUE products, superseded by `IMAGE`.
`'TABLE␣␣␣'`	S	Harten *et al.* (1988)	IAU	ASCII tables.
`'VGROUP␣␣'`	R	Jennings *et al.* (1996)	GSFC	Reserved for possible use in supporting analog of HDF group structures under FITS. Current proposal does not use separate extension type.

Table 5.1: Reserved Extension Type Names

Table 5.2: Possible Status Levels for FITS Extensions

Status	Description
S	Standard FITS extension type accepted by the IAUFWG and endorsed by the IAU.
P	Proposal for FITS extension type accepted by regional FITS committees but not yet by the IAUFWG.
D	Draft proposal, to be discussed in regional FITS committees and by community.
R	Reserved FITS extension type name for which a full draft proposal has not yet been submitted.
L	Local FITS extension type in use only by a particular group.

5.2 Conventions for Binary Tables

In the paper presenting the binary table extension (Cotton, Tody and Pence 1995; hereafter CTP), three conventions are described in appendixes. While these conventions were not part of the formal extension rules endorsed by the IAUFWG, a number of keywords were reserved with the intention that those keywords would be used for these conventions. Those keywords should not be used for any purpose inconsistent with these conventions. Publication in the binary table paper means that the these conventions will be widely recognized even without formal IAUFWG endorsement. Unless there is agreement by a particular community or group of FITS users on an alternative convention, the goal of standardizing formats for convenient data transport will best be achieved if users adopt these conventions when formatting their data in FITS.

The HEASARC FITSIO and FTOOLS packages support both variable length arrays and multidimensional arrays. The IUEDAC software in IDL can read both variable length and multidimensional arrays and can write multidimensional arrays but not variable length arrays.

5.2.1 Variable Length Arrays

In some tables, the number of elements in the array contained by a particular field may vary from row to row. One approach would be to set the repeat count equal to the largest number possible and use fill values for the remaining elements in the field when the number of actual values is smaller. However, if the variations from row to row are large, the resulting binary table will contain a large number of fill values, an inefficient use of storage. The binary table rules contain provisions that support a convention for efficient storage of variable length arrays, although the convention itself is not part of the formal proposal.

The basic concept is that the data for variable length arrays are not stored physically in the main table but in a heap area that follows the table. This heap area takes up part of or all of the space following the main table reserved by the `PCOUNT` keyword in the header. At the entry in the table where the array would be located, there is instead an array descriptor or pointer, which tells the user or software

- where the data to be logically included as that entry of the table are to be found in the heap.
- the size of the data.

The data identified in the heap are treated as if actually located in that entry of the table instead of in the heap. Since an array descriptor has a single, standard size, all rows in the main table will have the same length. This structure allows software that is unable to handle variable-length arrays to read the rest of the main table and then use the value of `PCOUNT` in the header to move on to the next extension when it has finished reading the fixed-length entries in the table. Variable length arrays may be of any of the data types used in binary tables except array descriptor and may be multidimensional.

The value of the `TFORM`*n* keyword in the header, which has the form '`rPt`(maxelem)', identifies the data type (`t`) of the array to be logically included in field *n* and specifies the largest number of elements the array can have (maxelem) in any row of the table. The maxelem value, which the wording in CTP includes as an essential component of the keyword value, allows software to read the table directly into a data base that supports only fixed size data arrays without first having to read the table solely to determine the size of the arrays. Without it, the task of the reader is more complicated. The `P` tells the reader that the actual contents of the field in the main table will not be the data array but the descriptor. For example, the card image

```
TFORM8  = 'PB(1800)'          / Variable length byte array
```

signifies that field 8 of the table should be regarded as logically containing a byte array of up to 1800 elements (here bytes). The actual physical contents of field 8 are two 32-bit integers. The first 32-bit integer is the number of heap elements to be considered a part of the current row. This value may differ from row to row, but, in the illustration here, cannot exceed 1800. A row in the table data may have a value of zero in this field, thus providing for fields where data are not present in every row. The second 32-bit integer is the offset of the start of the byte array from the beginning of the heap, counted in number of bytes. An array starting at the beginning of the heap has an offset of zero. The remaining elements of the array follow in sequence in the heap; the contents of a single field of a single row cannot be broken up in the heap. However, different fields in the same row need not appear in the heap in the order in which they appear in the table. Gaps in the heap, with no data, are possible. If two or more table fields have identical contents, the descriptors for both may point to the same area of the heap.

The heap need not begin at the end of the main table data. The user may choose to provide a gap between the main table and the beginning of the heap. The size of this gap can be found using the value of the `THEAP` keyword. If there is no gap, the value of `THEAP` is NAXIS1$\times$NAXIS2. If there is, the size of the gap is the difference between the value of `THEAP` and NAXIS1$\times$NAXIS2.

As an example, suppose the main table consists of five 168-byte rows, 840 bytes in all, but the user wishes to have the heap begin on a new record. (This construct is provided for purposes of this example; it is not necessarily recommended usage.) The value of `NAXIS1` is 168, representing the length in bytes of a row in the table, and the value of `NAXIS2` is 5, signifying 5 rows. There is then a $(2880 - 840)$- or 2040-byte gap between the main table and the heap. The value of `THEAP`, representing the number of bytes between the start of the main table data and the start of the heap, is 2880. Suppose further that the heap itself has a length of 5760 bytes. The value of `PCOUNT` is equal to the size of the gap area (2040) plus the size of the heap (5760), or 7800.

The storage location to which the array descriptor points must be located entirely within the heap. In particular, it may not be in the main table.

Reading a variable-length array on a random access storage medium, such as disk, is straightforward. When an array descriptor is encountered, the information in it can be used immediately to find the appropriate data in the heap and read them; the program can then move to the next table entry. On sequential access storage media, such as tape, the process is more complicated. As the main table is read, a table of the array descriptors must be generated and stored. Space is reserved in the output data set to hold the actual array, based upon the specifications of the array descriptors. When the main data array has been read, the array descriptors are first sorted in order of appearance in the heap, and then the data are read from the heap in order and stored as specified by the array descriptors. If more than one field has the same stored descriptor, the same heap data may be stored in more than one place.

Because variable-length arrays are more complicated to deal with than ordinary fixed-length arrays and require an extra data access per array to obtain all the data for a field, they should be used only when absolutely necessary.

As this convention is not part of the formal rules for binary tables, not all binary tables software will necessarily be able to read variable-length arrays, except for using the value of `PCOUNT` to skip the heap. In addition, before generally distributing a file with variable length arrays, be sure, by means of practical tests, that the file can be read on a machine other than the one used to write it.

5.2.2 Arrays of Strings

A user may wish to include a series of strings rather than one single string in a field of type A. To distinguish the component strings from the complete string field, they will be described as substrings in the remainder of this section. The convention discussed in this section allows a reader to use the value of the `TFORM`*n* keyword to tell whether a character string field contains a single string or a series of substrings. The convention provides for a simple fixed length case, where all substrings are of the same length, and a variable length option, where the length of the substrings may vary. The convention makes use of the fact that additional characters are permitted after the data type code in the value of `TFORM`*n*. It is signaled by the presence of a string of the form '`rA:SSTRw`[`/nnn`]' in the value field of the `TFORM`*n* keyword, with the `/nnn` present only if the length of the substrings may vary. The `:SSTR` is the indicator that the substring array convention is in use. A FITS reader that is not able to process substrings under this convention can treat the entire contents of the field as a single string.

In a fixed length field, `w` is the length of the individual substrings, for example,

```
TFORM5  = '40A:SSTR8'         /Fixed length substrings
```

tells the reader that field 5 is comprised of an array of five eight-character substrings. If a substring does not fill its specified subfield, the remainder must be padded with ASCII blanks. In the case above, for example, if the string in the first subfield were only six characters long, then bytes 7 and 8 would be blanks.

If the total length of the field is not an integer multiple of the length of the substring, the remainder of the field is undefined. For

```
TFORM5  = '14A:SSTR3'         /Fixed length substrings
```

field 5 contains four substrings of three bytes each followed by two undefined characters.

For a table field composed of substrings of variable length, all substrings but the last are terminated by an ASCII text character deliminator, whose decimal value, which must be in the range 032-126, is provided by the nnn in the value of the `TFORM`*n* keyword. Leading zeroes are included in the nnn. The last substring is terminated with an ASCII NULL (decimal 000; hexadecimal 00) regardless of the value of nnn. The value of `w` is the maximum number of characters in any substring of the array, not including the delimiter. For example, for

```
TFORM6  = '100A:SSTR8/032'     /Variable length strings
```

field 6 is 100 characters wide and composed of substrings whose length may vary, so long as no substring is longer than 8 characters. Each substring except the last is terminated by an ASCII BLANK (decimal 032 or hexadecimal 20); the character immediately following this delimiter is the first character of the next substring. The last substring is terminated by an ASCII NULL.

Undefined or null variable length substrings are represented by a string of zero length; the first position of the substring contains the designated delimiter (or an ASCII NULL if it is the last substring of the field). An ASCII NULL in the first position of the field signifies that the field contains no substrings. There is no corresponding representation for an undefined or null substring in an array of fixed length substrings. Possibilities include a blank substring or a user-defined substring, although general FITS readers won't necessarily recognize the special meaning of either. If a distinction needs to be made between undefined and blank substrings, it is better to use the variable length convention.

A possible alternative procedure for representing arrays of fixed length substrings is through the use of the `TDIM`*n* multidimensional array convention described in section 5.2.3. The first dimension is the number of characters in a string, and the remaining dimensions are those of the array. This convention must be used if the array of strings is two or more dimensions. CTP recommend use of the substring array convention for a one dimensional array of strings.

5.2.3 Multidimensional Arrays in Binary Tables

The original binary table proposal, the `A3DTABLE` extension, introduced the concept of a table entry or field of more than one element, a vector. It provided this structure through the use of a repeat count. However, since an array of more than one dimension can be reduced to a one-dimensional vector, as occurs when arrays are stored in a computer, the repeat count can be used to define multidimensional arrays as well. To do so, conventions are needed to specify how such an array is organized in the table field.

5.2.3.1 `TDIM`*n* Keyword

CTP propose one convention, using the `TDIMn` keyword, which has been reserved for multidimensional array description. The value of the `TDIMn` keyword, `'(i,j,k,...)'`, describes the dimensions of the array, $i \times j \times k \times \ldots$, stored such that the index along the first dimension (i) varies most rapidly, the same convention used for the primary data array. The product of the dimensions must be equal to the repeat count specified in the value of the `TFORM`*n* keyword. This convention is not a formal part of the proposed binary table definition. Other conventions are possible and may be used.

5.2.3.2 Green Bank Convention

This convention was developed at a meeting to discuss standard FITS formats for interchange of single dish radio astronomy data, held at Green Bank WV, in

October, 1989. It is designed primarily for the case where only one field is a multidimensional array or where all array fields in a row have the same dimensions.

The extension header keyword

> `MAXIS` (integer) gives the number of dimensions of the array in the table field.

The properties of the array would be given in columns in the table. The column labels would be the names of the keywords used to describe a primary or image data array, except that `NAXIS`n would be replaced by `MAXIS`m. These labels are given by the values of the `TTYPE`n keywords in the binary table header. Thus,

- `TTYPE`n`= 'MAXIS`m`'` means that the number of elements along axis m in the array contained in the table is given in column n of the table. `MAXIS1` describes the most rapidly varying axis of the array.
- `TTYPE`n`= 'CTYPE`m`'` means that the name of the physical coordinate of axis m in the array contained in the table is given in column n of the table.
- `TTYPE`n`= 'CRPIX`m`'` means that the reference point for axis m in the array contained in the table is given in column n of the table.
- `TTYPE`n`= 'CRVAL`m`'` means that the value of the physical coordinate for axis m at the reference point in the array contained in the table is given in column n of the table.
- `TTYPE`n`= 'CDELT`m`'` means that the rate of change of the physical coordinate along axis m per unit change in the counting index at the reference point of the array contained in the table is given in column n of the table.

Because the array field must be the same size in all rows, the values of the `MAXIS`m will usually be the same for all rows of the table, and they can then be given as header keywords instead of table columns:

`MAXIS`$m, m = 1, \ldots,$ `MAXIS` (integer) gives the number of elements along axis m in the array contained in the table.

It is possible for array to have the same size in two rows but different numbers of elements along the axes, for example 36×6 in one row and 24×9 in another. In that case, the number of elements along each axis must be given in a table column.

One field – call it k – will contain the actual data array. For this row, two keyword-value pairs are required:

```
TFORMk  = 'DATA    '
TMATXk  =                    T
```

A value of `F` is understood for the `TMATX`m keyword for all other columns and need not be included explicitly.

It is possible to combine this convention with the use of the `TDIM`n keyword proposed in the binary table paper by setting

```
TDIMk     = '(MAXIS1,MAXIS2,...)'
```

where the `MAXIS`m represent the number of elements along axis m that would otherwise be the values of the `MAXIS`m keywords. The CLASS package for analysis of radio astronomical data supports the Green Bank convention.

5.2.4 Some Applications of Binary Tables

5.2.4.1 Replacing Random Groups

As noted in section 3.2, the random groups structure of Greisen and Harten (1981) has been used only for transporting radio interferometry observations. The binary table structure with multidimensional arrays can be used to hold the exact same data as the random groups structure. Each parameters/array group corresponds to a row in the table. The parameters map into single-item fields and the array into a field with a multidimensional array. The names and other information describing the parameters would be in the table field headings. This format is more flexible than random groups. The parameters in the random groups format must be of the same data type (e.g., two-byte integer, double precision floating point) as the members of the array they precede. In a binary table, on the other hand, the data type of the fields containing the parameters does not have such a restriction.

5.2.4.2 Multiple Arrays in One HDU

While a series of matrices can often be combined into a single primary data array by adding another dimension, to do so is not always possible. For example, consider an array that consists of a digital image accompanied by an array of quality flags, each of which corresponds to a pixel on the original image. If the data type of the flags differs from that of the image pixels, the flags cannot be combined into the primary data array. One way is to have the data array and flag arrays as separate image extensions. An alternative is to use binary tables. Each row would contain data for one image. One column would be a multidimensional array in floating point format, containing the image, and another would be a multidimensional array in integer format, containing the flags.

5.3 Hierarchical Grouping Proposal

(Jennings *et al.* 1996) have proposed this convention, which provides a mechanism for logically associating different HDUs that are physically separated into groups that may have a hierarchical structure. It arose in part from efforts to convert between FITS and the Hierarchical Data Format (HDF). HDF is under continuous development at the National Center for Supercomputing Applications (NCSA), and current information is available from the NCSA World Wide Web site at `http://hdf.ncsa.uiuc.edu/`. The discussion in this *Guide* is taken from that paper. Whether this convention needs to be submitted for IAUFWG endorsement or may be considered a recognized convention without being part of the formal FITS rules is still under discussion.

The grouping proposal provides several capabilities:

- It provides an alternative to `EXTLEVEL` for creating a hierarchical structure, one that has an explicitly defined tree structure identifying the higher level groups to which each lower level group belongs.
- It creates a FITS structure analogous to the Vgroup structure of HDF.

- It provides a means for logically including a particular HDU in more than one group without repeating the HDU.
- It provides a means for associating HDUs that are in different files, even those in different physical locations, into logical groups.

The last feature is becoming particularly useful as the fraction of data available on-line has increased and access from one computer to data on another has become routine. Related data may now be stored in separate locations and accessed electronically, instead of putting separate copies of all the related data on several machines.

Jennings *et al.* provide an illustration of where such a convention would be useful. Consider a set of observations, each of which has associated with it an observation log, event list, derived image, and calibration data, and for which many observations share the same calibration data. It would be possible to logically associate each (log, event list, image, calibration) set into a separate group; in addition, the same calibration data HDU could belong to many groups but would only need to be stored once. Another possible group would be one of observations of the same object at different times, and these observations could be combined into a group even though they might reside on multiple FITS files.

An extension HDU called a *group table* defines a group and its members. This table may be either an ASCII table extension (`XTENSION= 'TABLE   '`) or a binary table (`XTENSION= 'BINTABLE'`). Group tables are required to contain the following keywords in the header:

- `EXTNAME` (character) must have the value `GROUPING`
- `EXTVER` (integer) has a value that serves as a group identification number. Every HDU with `EXTNAME = 'GROUPING'` in a given FITS file must have a different value of `EXTVER`. This number distinguishes the group from any other defined in the file.

The following keyword is not required but is strongly recommended:

`GRPNAME` (character) provides a name for the group table. This name should contain only letters, digits, and underscore (hexadecimal 5F, decimal 95); in particular, it should not contain any embedded ASCII blanks (hexadecimal 20, decimal 32). The value is case-insensitive.

A group can contain any number of members. The members may be a mix of HDU types, in any order. They may be primary HDUs, extensions of types endorsed by the IAUFWG (currently `TABLE`, `IMAGE`, and `BINTABLE`), and other extensions that conform to the general rules for extensions provided in section 3.3 but have not been endorsed specifically by the IAUFWG. Member tables may be group tables, thus identifying subordinate groups. Group members can be identified by the group table in two ways: by reference or by position. Either or both can be used. Both have potential drawbacks. For identification by reference, the HDU is identified using the values of the `XTENSION`, `EXTNAME`, and `EXTVER` keywords. A possible difficulty is that there may be more than one HDU with these keywords in a previously created data set, making it impossible to identify the HDU uniquely. For identification by position, the numerical order of the HDU in the data set is specified. Caution is required in the use of this method because the order of HDUs in a FITS file may be changed when the file is copied. The location of group members not in the same file as the group table is specified by the World Wide Web Uniform Resource Identifier (URI). The table must specify the kind of URI being used: Uniform Resource Locator (URL), Uniform Resource Name (URN), or any future form of URI. The identification information is provided in columns of the group table with prescribed names, given as values of the `TTYPE`*n* keyword.

The following keywords are used to define columns of the group table if all or some group members are identified by reference:

- `TTYPEn = 'MEMBER_XTENSION'` is the heading for a column whose entries are character strings with the values of the `XTENSION` keyword, for extensions that belong to the group. For primary HDUs that are members, the string `'PRIMARY␣'` is be used. If no value is given, as when the group table contains a mix of identifications by reference and by position and the HDU is identified by position, an ASCII NULL (hexadecimal `00`, decimal 000) may be used.

- `TTYPEn = 'MEMBER_NAME'` is the heading for a column whose entries are character strings with the values of the `EXTNAME` keyword for group member extensions. The treatment of extensions with no value of `EXTNAME` is the same as that for the case where no value of `XTENSION` is available.

- `TTYPEn = 'MEMBER_VERSION` is the heading for a column whose entries are integers with the values of the `EXTVER` keyword for group member extensions. For primary headers or where the `EXTVER` keyword is absent, the entry should be 1.

The following keyword is used to define a column needed if all or some members are identified by position:

- `TTYPEn = 'MEMBER_POSITION'` is the heading for a column of group member positions given as integers, with 0 (zero) representing the primary HDU, 1 (one) the first extension, etc. If the value is invalid or undefined, the entry should contain the null value specified by the `TNULLn` keyword for that column.

The following keywords are required if some or all of the group members are not in the same file as the group table:

- `TTYPEn = 'MEMBER_URI_TYPE'` is the heading for a column containing the acronym for the URI type used to identify the member location. Entries permitted as this is written are URN and URL; other entries will become acceptable as new URI types are defined.
- `TTYPEn = 'MEMBER_LOCATION'` is the heading for a column containing the URIs for the group members. For group members in the same file as the group table, the field may be filled with blanks; for members on the same machine but in a different file, the relative URI may be used instead of the absolute URI.

If either of the above fields is undefined, the field should contain an ASCII NULL.

A group table may contain any number of additional, user-defined columns. Group table columns may appear in any order. The values of `TTYPE`n are not case-sensitive.

To permit inclusion of HDUs from existing FITS files in groups, the grouping proposal does not require any new keywords for group members. Use of the `EXTNAME` and `EXTVAL` keywords defined in section 3.3.2, the rules for generalized extensions, permitting identification by reference, is strongly recommended. Every member of a group located in a particular file should have a unique combination of `XTENSION`, `EXTNAME`, and `EXTVER` keywords. Group tables that are members should have a unique combination of `EXTNAME` and `EXTVER` keywords, and there should not be an ASCII group table and a binary group table with the same (`EXTNAME`, `EXTVER`) in the same file. In addition, the use of `EXTNAME` allows HDUs of a given extension type and structure to be identified by a common name.

Use of the following two keywords also is strongly recommended:

- `GRPID`*n* (integer) gives the `EXTVER` value of the *n*th group table of which the HDU is a member if the member is in the same file as the group table and is the negative of the `EXTVER` value if the member and group table are in different files. The presence of this keyword also warns software that is changing the position of the HDU that the HDU is a member of a group, naming the group table where entries used for identification by position will need to be changed.
- `GRPLC`*n* (character) gives the URI for group table *n*. It is not necessary if the value of `GRPID`*n* is positive.

An HDU may be a member of many groups. If the HDU is a member of *m* groups, then there will be `GRPID1` ... `GRPID`*m* keywords, each identifying one of these groups. If the value of a particular `GRPID`*n* is negative, it is the negative of the group identification number (the value of `EXTVER` in the group table), the group table is not in the same file as the member, and there will be a `GRPLC`*n* keyword with the URI where the group table is located.

The Jennings *et al.* proposal presents a number of illustrations. As of this writing, it is available at
`http://wwwadf.gsfc.nasa.gov/other/convert/group.html`.

5.4 STScI Inheritance Convention

The rules of FITS do not specify whether or not keywords in the primary header apply to extension data in the same file. STScI has adopted the following conventions governing such inheritance of keywords (Zarate and Greenfield 1996).

1. Primary header keywords can be inherited only if there is no primary data array.

2. Primary header keywords apply to extensions only if the header contains the keyword `INHERIT` with the value set to `T`.

3. Required FITS keywords are not inherited.

4. If a keyword appears both in the primary header and an extension header, the value in the extension header will apply to the extension.

5. The reserved commentary keywords, `COMMENT`, `HISTORY`, and ␣␣␣␣␣␣␣␣, are never inherited.

6. Changes to an inherited keyword appear only in the extension header, not the primary header, i.e., the inherited keyword will appear in the extension header with its new value and the primary header value remains unchanged.

As with any FITS convention, software may allow for overriding the inheritance convention for specific purposes. For example, the IRAF FITS kernel has a mechanism to override the rules given in items 1 and 2.

5.5 Checksum Proposal

Seaman and Pence (1995) have proposed a means for embedding a checksum within a FITS files. The proposal has been presented for discussion to the community as the first step in the process of seeking endorsement by the IAUFWG. This section summarizes their proposal; more details can be found in the paper itself. The purpose of such a checksum is to verify that there has been no error in a data transmission. A checksum is calculated for the data that are to be sent and then calculated in the same way where the data are received; the two are then compared. All checksums are calculated using ones-complement arithmetic; that is, the result of overflow in addition in the most significant bit is carried over to the least significant bit.

Three keywords are reserved for this proposal:

- `CHECKSUM` (character) is a 16-character string containing an ASCII encoding of the ones-complement of the checksum of the HDU calculated with the value of the `CHECKSUM` keyword set to `'0000000000000000'` (ASCII 0s or hexadecimal 30s) before the checksum is computed.

- `DATASUM` (character) is a character string, containing the unsigned integer value of the checksum of the data of the HDU, formatted as a string. The string formulation is used because FITS does not support unsigned integer keyword values. If the HDU has no data, the value field may be set to `'0'` (preferred) or the keyword is omitted. The single digit is used in the value field to distinguish it from an actual zero data checksum. Absence of a `DATASUM` keyword implies no knowledge of the checksum of the data records.

- `CHECKVER` (character) is a character string that identifies the checksum algorithm used. The algorithm defined by Seaman and Pence has the identification `'COMPLEMENT'` and is the default value. Development of future algorithms should be coordinated within the community and governed by a registry under the IAUFWG like the one for extension type names described in section 3.3. It is recommended that this keyword appear only if an algorithm other than the default `COMPLEMENT` algorithm is used.

For purposes of calculating the checksum, the header and data of each HDU are treated as a series of 32-bit words. Checksums are calculated by adding these words, using ones-complement arithmetic, to yield a 32-bit integer. First the ones-complement checksum of the data records is calculated. The unsigned integer result is then formatted as a string and put in the value of the `DATASUM` keyword. For example, if the checksum of the data records is the unsigned integer 2503531142, then the `DATASUM` keyword in the header will appear as

```
DATASUM␣=␣'2503531142'
```

The checksum keyword in the header is then set to an encoded ASCII zero:

```
CHECKSUM=␣'0000000000000000'
```

The ones complement checksum of the header is then calculated, including the `CHECKSUM` and `DATASUM` keywords with the above values, and the already calculated data checksum is added to it, yielding a checksum for the entire HDU. The bit complement of the total, which is its ones-complement additive inverse, is then calculated such that

$$\text{original\_checksum} + \text{bit\_complement} = 0$$

This bit complement is then ASCII encoded. As part of the encoding process an encoded ASCII zero is added:

$$\text{encoded\_complement} = \text{bit\_complement} + \text{ASCII\_zero}$$

This encoded complement replaces the original value of the checksum keyword, the ASCII zero, in effect subtracting it out:

$$\begin{aligned}
\text{new_checksum} &= \text{original_checksum} + \text{encoded_complement} - \text{ASCII_zero} \\
&= \text{original_checksum} + \text{bit_complement} \\
&= 0
\end{aligned}$$

Thus, with the encoded complement in the header, the checksum of the HDU is now zero.

Seaman and Pence recommend an algorithm for encoding the checksum complement for inclusion as the value of the `CHECKSUM` keyword. This scheme encodes not only the meaning of the binary data but the checksum's actual 32-bit unsigned integer value. Each of the four bytes in this value is numerically divided by four and the result placed in the corresponding byte of four quotients in four successive integer aligned fields; the remainder for each byte is added to the first quotient. For example, the first byte of the second quotient field contains a value equal to one-fourth of the first byte of the original checksum value. The sum of the four quotients is the original checksum; in addition, the byte alignment is maintained: the sum of the first bytes of the four quotients is the first byte of the original checksum complement, similarly with the other three. An ASCII zero is then added to all the bytes, thus creating a 16-character printable ASCII string. To improve readability, if byte values are not alphanumeric, then the values will be shifted by simultaneously incrementing one byte and decrementing the corresponding byte in another of the four quotients in a systematic way, until both are alphanumeric. In this process, the sum is not changed. In accordance with the FITS fixed format rules for character strings, the initial quote is in column 11 and the first byte will be in column 12. In order to maintain proper byte alignment within the four byte word, the final byte of the encoded checksum is permuted to column 12; thus, it remains the fourth byte of a four-byte word, the next byte is the first, and all the rest maintain proper position. This value is then set in the value field of the checksum. Seaman and Pence present a detailed example. The string can easily be decoded by permuting the encoded string back to its original alignment, subtracting the hexadecimal `30` offset, and summing the four parts.

This algorithm is not mandated by the proposal; other encodings are possible but must satisfy the following constraints:

- The 32-bit unsigned integer `CHECKSUM` value is encoded into a 16-character ASCII string.
- Only the ASCII alphanumeric characters 0-9, A-Z, and a-z are permitted.
- The quotes enclosing the ASCII string value of the `CHECKSUM` keyword must be in columns 11 and 28.
- The encoded ASCII value will cause the 32-bit ones-complement checksum of the HDU to be zero.

The checksum is zero for the entire file because it is zero for each individual HDU of . Requiring a zero checksum for individual HDUs rather than only for the file as a whole allows HDUs to be removed from or added to the file without altering the checksum.

The following *strong recommendations* are made regarding the usage of these keywords:

- FITS files should include the checksum keywords.
- The checksum keywords should be included in all HDUs of FITS files that contain more than one HDU.
- All HDUs of a single FITS file should use the same checksum algorithm, i. e., the values of `CHECKVER` should be the same.
- When an HDU is modified, FITS software should update the checksum keywords if present and add them if not present.

- At the very least, FITS software should delete the keywords if an HDU is modified and the checksums are not recomputed.

The last two recommendations must be followed to avoid creating HDUs containg checksum keywords with values inconsistent with the HDU contents.

Seaman and Pence present a detailed example of the use of the checksum convention and the encoding described above, as well as software to in both C and FORTRAN 77 to implement the encoding. The C routines are adapted from the on-line software for the NOAO/IRAF *Save the Bits* archive; the FORTRAN 77 routines are adapted from the FITSIO package.

5.6 Other Proposed Conventions

5.6.1 HEASARC

The HEASARC at Goddard Space Flight Center is responsible for archiving a considerable volume of data from many different high energy astrophysics missions, including missions under the auspices of NASA and missions in which NASA has cooperated with and supported other agencies. Under the Office of Guest Investigator Programs (OGIP), the HEASARC FITS Working Group has been organized to formulate recommendations for FITS practice, required for files produced by HEASARC/OGIP but intended for application throughout the high energy astrophysics community.

5.6.1.1 Keywords and column names

In keywords and table column names, HEASARC recommends use of the underscore (hexadecimal `5F`) rather than the hyphen (hexadecimal `2D`). The only cases where a hyphen is to be used in keywords are

- where use of a hyphen is established usage, as for example in the `DATE-OBS` keyword.
- to represent a minus (negation) sign

This set of recommendations is similar to those in the original paper for ASCII table fields, as discussed in section 3.4.2, but extends them to keywords and binary table fields. Extending the rules for keywords to column names makes it possible to replace a table column heading with a keyword with the same name if the value for that column in every row of the table is the same, as in the Green Bank convention (section 5.2.3.2).

HEASARC/OGIP requires the following additional conventions for column names in all tables it produces and recommends them for others:

- Different columns in the same table cannot have the same name and all columns must be unique in the first 16 characters.
- Embedded blanks are specifically forbidden, as they are for keywords.
- Values of the `TTYPE`*n* keyword are case-insensitive, as specified for ASCII tables in the NOST standard described in section 6.1.2.
- Column names must begin with an alphabetic character (A-Z, a-z), not a digit or underscore.
- Column names may be up to 68 characters in length.

5.6.1.2 Proposed `CREATOR` Keyword

> `CREATOR` (character) identifies the software that originally created a FITS file.

The value refers to the software that originally defined the FITS file structure and wrote the contents; it should not be changed if the file is copied largely intact into another file. Use `HISTORY` keywords to document such travels of the file after it has been created. When appropriate, the value of the `CREATOR` keyword should identify the specific version of the software that created the FITS file; e.g. `'WRITEFITS V3.2'`. The `AUTHOR` keyword (section 3.1.1.2) has sometimes been misused for this purpose, but `AUTHOR` is reserved for the human author or compiler of a document.

5.6.1.3 Proposed `TSORTKEY` Convention

The purpose of this convention is to provide information on the ordering of rows in a FITS ASCII or binary table. How the table has been ordered is useful information for users; indeed, it may be necessary for some data processing operations. Usually, the rows are in order of increasing values in a particular column in the table, for example, time or right ascension. To specify the order used in a particular FITS table, a new keyword is used:

> `TSORTKEY` (character) lists the name of the primary sort column, given by the value of the `TTYPE`*n* keyword, followed, optionally, by the names of secondary sort columns. The names must be separated by a comma; there may be zero or more spaces between a comma and the next column name.

The default order for sorting in this convention is ascending order; if the sort is in descending order, the name of the column is preceded by a minus sign.

This convention prescribes the following ascending sort order for the FITS data types:

Integer and floating point In numerical order for both ASCII and binary tables; ASCII tables are not sorted in order of the internal ASCII representation of their contents.

Complex In numerical order of the real component first; rows with the same value of the real component are sorted in order of the imaginary component.

Bit Zero or "unset" bits precede "set" bits.

Logical False (`F`) values precede true (`T`) values.

Character and string In order of the ASCII collating sequence; for a string, first in order of the first character, then in order of the second character, than in order of following characters in sequence.

In an ascending order search, null or undefined characters follow all defined values.

The convention also provides a sort order for vector columns. If the order is not otherwise specified, rows are sorted first on the order of the value of the first element, then in order of the value of the second element, and on in sequence. Other columns may be used for sorting. If the column `ARRAY` is being sorted first on element i, then on element j, and then on element k, then

`TSORTKEY~='ARRAY(`i`),ARRAY(`j`),ARRAY(`k`)'`

If the sorting is on consecutive columns $m \ldots n$, then

`TSORTKEY~='ARRAY(`m`:`n`)', ARRAY(`k`)'`

A string is treated as a vector of characters. When sorting strings, a blank element precedes any characters. For an ascending order default sort, shorter strings precede longer ones: "`FORM`␣␣" precedes "`FORMAT`".

This convention can be used only if the values of the `TTYPE`n keywords labelling the columns of the table obey the following restrictions:

- There must be a `TTYPE`n for every column, and the name of every column (value of `TTYPE`n) must be unique.
- The first character of a column name must not be a minus sign.
- No column name may contain a comma (hexadecimal `2C`), or an open or close parenthesis (hexadecimal 28 and 29).
- Embedded blank characters are discouraged, with the underscore recommended to link separate words in a column name.

5.6.1.4 Maximum and Mininum Values in Table Columns

HEASARC uses the following keywords to specify the limits on data in a table column. They are analogous to the `DATAMAX` and `DATAMIN` keywords used for arrays.

- `TDMAX`n is the maximum value in column n of a table, exclusive of null or fill values.
- `TDMIN`n is the minimum value in column n of a table, exclusive of null or fill values.
- `TLMAX`n is the maximum permissible value in column n of a table.
- `TLMIN`n is the minimum permissible value in column n of a table.

For example, if the third column in a table of positions was declination, then the following keywords might be present.

```
TTYPE3  = 'DEC     '
TFORM3  = '1E      '
TUNIT3  = 'DEGREES '
TLMAX3  =               90.00
TLMIN3  =              -90.00
TDMAX3  =               46.00
TDMIN3  =              -63.10
```

HEASARC also recommends the following restrictions when these keywords are used:

- The data type of the keyword value should be the same as that of the data in the column it describes.
- The keywords are to refer to *all* elements of a vector column
- As with `DATAMAX` and `DATAMIN`, the keywords apply to the values after any scaling prescribed by the values of the `TZERO`n and `TSCAL`n keywords has been applied.
- Values of `TDMIN`n less than those of `TLMIN`n and values of `TDMAX`n greater than those of `TLMAX`n are permissible; the meaning is not defined in this convention.
- If the value of the keyword for the maximum is less than that of the value of that for the minimum, then the values are taken to be undefined.

5.6.2 World Coordinates in Tables

Primary data arrays and image extension arrays are not the only way in which maps may be defined in FITS files. The following applications are also possible:

1. A vector column in a binary table may be a multidimensional array that represents a map in the same way as a primary data array or image extension array.
2. The entries in a binary table may represent objects and their positions on a map. An event list would produce such a table. The map axes could be, for example, sky positions (R. A., Dec) or locations on a detector grid (x,y). One column in the table would be assigned to each coordinate of the positions of the objects. The map coordinates and possibly the transformation between detector and sky position must be described in some way.
3. The user may wish to tabulate a set of coordinate transformation keywords applying to different images, for example, defining how the celestial pointing of an imaging instrument changed with time.

Standard keywords should have a consistent usage throughout FITS. The keywords reserved to describe primary arrays and images should be used to describe coordinates in these other applications only if they have they same meaning. In particular, they should never be used in a way that can or might lead to contradictory meanings in the same HDU.

In the case of a multidimensional array in a table, the coordinate keywords are being used in the same way; as for a data array, they tell how to interpret a one-dimensional array of data values in the FITS file as a multidimensional matrix. If there is only one array in the table, or if all the arrays in the table have the same properties, the standard keywords that describe the meaning and scaling of the axes—`CTYPE`n, `CRPIX`n, `CRVAL`n, and `CDELT`n—can be used in the header. On the other hand, because they are already used in the header to define the table as a two-dimensional array, the `NAXIS` and `NAXIS`n keywords cannot also be used to describe the member arrays. The Green Bank convention described in section 5.2.3.2 avoids the problem of keyword conflict by using different keywords, `MAXIS` and `MAXIS`m, to describe the array in the table.

In the other two cases the usage is different.

In array usage, the location of a data value relative to the axes is associated with its *position* in an array. In the table of positions, on the other hand, the location of a data value relative to one or more axes is associated with its actual *value*. One can even imagine a case where both kinds of coordinate might appear in a table, two fields giving x and y positions and another field containing an array. If the same keywords could be used in both contexts, it would not be clear which coordinate system was being described by the header keywords. To avoid possible confusion, the standard array keywords should not be used to describe a coordinate system where the values of individual table entries are located, such as the coordinate system of (x, y) positions.

For a table of coordinate tranformations, the temptation may be to use the standard keyword names as table headers. Doing so, however, would lead to a situation in which the same name had different meaning in the header and body, for example, when `NAXIS` = 2 in the header, it refers to the dimensions of the data table in the same HDU, but in `TTYPE`n `= 'NAXIS   '`, it refers to the dimensions of arrays in a different HDU. Again, the standard keyword names should not be used.

The most recent version of the World Coordinates proposal includes a description of conventions proposed by HEASARC. These conventions are available on the HEASARC and NRAO World Wide Web sites discussed in sections 6.2 and 6.3.

5.6.3 Compression

Proposals for ways to include compressed data in FITS files are still under development. A preliminary proposal by Warnock *et al.* (1990) is available from the NRAO ftp directories described in section 6.2. However, there are some compression-related questions this proposal does not address. Other installations are experimenting with local schemes.

5.6.4 Other Reserved Type Names

In the original FITS design, intended for half-inch magnetic tape, the end of a file could easily be identified by a tape end-of-file mark. Not all other media support such a clear hardware delimiter. D. Wells has proposed the creation of a

FILEMARK extension, which would represent an end-of-file on media where the end of the file cannot easily be marked on the hardware. The name has been reserved for use in developing this concept; as yet there is no detailed proposal.

A FITS file might contain a pure bit stream, such as telemetry. An extension type name DUMP has been proposed and reserved for such data. Details of how this extension would be structured remain to be developed.

5.6.5 Developing New Conventions

The conventions that have been described in this section are not formally a part of the FITS rules and do not exclude others. However, since the purpose of such conventions is to develop a widely used and understood set of practices, new conventions should be developed only if the existing conventions in general use will not work for new data. Community comment and review should be part of development of any new convention. If design of a FITS data set is constrained by project deadlines, the intended users of the data and other groups that are making similar data publicly available should be consulted early in the development process. Proposed data designs with new conventions should be made available for review and comment to the user community as soon as a draft is available; opportunity for review should also be given to the general astronomical community to the extent possible consistent with project procedures. Requests for review should emphasize any deadlines. The proposed conventions should then be revised on the basis of the response before being incorporated in data sets.

5.7 Keyword Domains

Keywords that are not required or reserved for the standard FITS structures might easily have different meanings for different groups. Thus, the significance of a keyword would depend upon the context. The FITS community has looked for possible ways to define domains that would identify which keyword definitions apply. The initial approach to defining keyword domains was the concept of keyword hierarchies. In this method, syntax in the FITS file would indicate that certain keywords were to be considered as part of a named domain. Such domains might exist in a hierarchy of levels. For example, the highest level might be a particular institution, and lower levels could be individual projects within the institution. However, consensus about what the structure should be appears unlikely.

The reasons for the lack of agreement become apparent from an examination of the principal approaches. One approach would use the string "`HIERARCH`" in columns 1–8 of a header card image as domain indicator. Domain names, at progressively lower levels, would follow in sequence in the card image, and, finally, the keyword=value statement would appear after the lowest level domain name. Critics of this approach argue that the `HIERARCH` keyword and the domain names would leave little room on the card image for the actual keyword and value information, or for comment fields. Also, many FITS readers would be looking in columns 1–8 to find the keyword, and the keywords behind `HIERARCH` would be inaccessible. Special parsers would need to be built.

The other approach would use a "`KHBEGIN `" keyword to signal the beginning of a domain; the value would be the domain name. Subsequent keywords, beginning in column 1, would have the value for that domain. A "`KHEND   `" keyword would signal the end of a domain. Domains could be nested. Within the domains, keywords would begin in column 1, as do the other FITS keywords. Critics of this approach argue that the placement of the low-level hierarchical keywords in columns 1–8 would mix them with other simple nonstandard FITS keywords, existing FITS readers might generate incorrect results, and special code would be needed to interpret the keywords properly.

The difference between these two approaches is not just a matter of detail; it is the principle of whether keywords restricted to a particular domain should appear in the first eight columns of the card image or should start in a later column. Because this issue has not been resolved by the FITS committees and resolution does not appear imminent, *do not assume that FITS readers will recognize either scheme.* For that reason, neither approach is discussed further in this document.

Some installations have implemented a convention similar to the `HIERARCH` proposal in which a keyword without values, such as `HISTORY` (or even `HIERARCH`) appears in the first eight columns, followed by one or more domain names and then the keyword. The ESO Archive, for example, uses `HIERARCH`. The first sample FITS header in the first FITS paper (Wells, Greisen, and Harten 1981) illustrated this approach, but it was not formally proposed or tested for interoperability at the time. If a group chooses to use such a convention, the header should always allow a reader unfamiliar with it to interpret those card images as pure comments. An example of such usage appears in Example 1 of Appendix A.

Existing constructs in FITS provide another approach to domain definition. Some reserved FITS keywords—for example, `ORIGIN`, `TELESCOP`, and `INSTRUME`—describe natural domain limiters, the installation where the FITS file was written and the telescope and instrument used to obtain the data. Keywords following an `ORIGIN` keyword in an HDU could be understood as having the meaning defined for the particular installation given by the value of the `ORIGIN` keyword; similarly `TELESCOP` and `INSTRUME`, when present, could signify that keywords following have the meaning defined for data sets from the telescope and/or instrument named. It is then possible to define keyword meanings specific to a particular organization, telescope, and/or instrument and use the values of the relevant keywords to signify that such a particular set of keyword definitions applies. In this way, even though a particular keyword might be used differently in different contexts, the applicable context could be identified.

The same problem can arise in naming table column headers as when choosing keyword names: different groups may use the same column headings with different meaning. Again, some method of defining project-specific meanings for the table headings is needed. The proposal to use the `ORIGIN`, `TELESCOP`, and `INSTRUME` keywords as domain limiters works here, too. These keywords could be either in the primary header of the file or the individual extension header. Including these keywords in the extension headers has the disadvantage that they will be repeated from header to header, using extra space, but it has advantages: the domain indicators appear as keywords in the same header as keywords giving the column labels they describe, and they will remain associated with the table, should for some reason the table but not the primary header be copied from the FITS data set to another, say, to a data set with a different primary header. The additional information gained by including domain keywords in each extension thus appears worth the small extra overhead.

One possible problem that has been suggested for this approach is that some local software systems might rearrange keyword order when reading the FITS file into an internal format structure and so lose the domain information. If this approach is used, keyword order should not be critical.

Because they are used in domain conventions, use of the keywords `HIERARCH`, `KHBEGIN`, and `KHEND` for other purposes is discouraged. All keywords with important information must appear in columns 1–8.

5.8 Polarization

No convention has been formally and universally adopted within FITS to describe polarization. The convention most widely used is the one developed by NRAO that is supported in AIPS. This convention has also been adopted in a proposed standard FITS form for interferometry intended to replace Random Groups.

When values of the Stokes parameters are mapped, an axis with the `CTYPE`n value `'STOKES  '` is used to describe which Stokes or correlator parameters are displayed in the images. The values and the parameters they denote are shown in Table 5.3.

Table 5.3: NRAO Stokes Parameters Convention

-8	-7	-6	-5	-4	-3	-2	-1	1	2	3	4
YX	XY	YY	XX	LR	RL	LL	RR	I	Q	U	V

X and Y are linear polarizations with fixed orientation relative to the antenna. In the interferometry proposal, X is horizontal and Y vertical. R and L refer to right and left circular polarization. The convention has been suggested that -2 be used to refer to left circularly polarized images and -1 to right circularly polarized images. The definitions of circular polarization follow the IEEE convention; right circular polarization corresponds to the electric vector of an approaching wave rotating clockwise. This convention should be noted in the comments; use of the opposite convention is not unknown.

5.9 Spectra

There is, as yet, no formal agreement or convention on encoding spectra from random positions on the sky in FITS format. Conventions for describing the mapping between pixel number and frequency need to be developed. In fact, there has not even been agreement on whether frequency should be the only independent variable for spectroscopic data exchange or whether wavelength also should be used. Section 4.2.1 discusses one proposal for attaching multiple labels to one axis. While the resolution of these issues is outside the scope of this *User's Guide*, it is important to recognize that they exist.

Spectra can be transmitted as one-dimensional arrays, using the Basic FITS format with `NAXIS` = 1. However, such usage is inefficient. Each spectrum would have its own header. Spectroscopic observations generally need more ancillary parameters than do images. The header may easily use more bytes than the data. Approaches that would reduce header overhead are needed.

It may be possible to encode many spectra in a single array by adding a second dimension—scan number. Information about the entire set of observations would appear in the primary header; information about the individual scans would appear in a table extension or extensions following the data. This choice has the advantage that the data format is the one most widely and easily read. A principal disadvantage is that the ancillary information for individual scans is separated from the results of the observations.

If there are not too many parameters and the individual spectrum vectors are not long, ASCII tables may be a convenient format. One row would correspond to an observation, containing the ancillary information and the results. Each wavelength would be a separate field of the table, and the other fields would contain the ancillary parameters. However, as the spectra become longer and the number of points in a data set increases, the inefficiencies in use of space by the ASCII table format become a serious problem.

Space can be reduced by using binary tables. The binary form is more compact than the ASCII representation, especially for exponential formats and numbers with many significant digits. Also, because many machines are now storing floating point numbers internally in IEEE format, using binary tables eliminates time and cost associated with internal conversion in the machine. Again, one row corresponds to one observation, but now the spectrum vector appears formally as a single field. Parameters whose values are constant for the entire set of observations would remain in the header, while those applying to a single observation matrix would appear in the same row of the table. Example 4 in Appendix A presents the header that might be written for a simplified binary table structure for spectra proposed by D. Wells.

5.10 High Energy Astrophysics Applications

The natural form for storing data for high energy astrophysics is as a table of events. Consequently, binary tables are used extensively for high energy astrophysics data. One example of a design for such tables is that for data from the High Resolution Imager on ROSAT.

- The primary header, in addition to the required keywords, contains data set identification and history. `NAXIS` is set to zero; there is no primary data array.
- The first extension is a "Good Time Intervals" table, identifying those time intervals during which the data received from the satellite can be regarded as valid measurements. Each row of this table consists of two double-precision floating point fields, listing the start time and end time of a time interval where the data are valid.
- The next extension is the "Events" table, including most of the data from the native time-ordered event file.
- The actual columns in the ROSAT tables contain an information structure governed by the way in which the ROSAT readings are organized.

The kind of information obtained from any high-energy experiment could be organized in a similar table. The header could, in addition to the card images describing the structure of the table, use keyword=value statements to convey such information as the conversion between instrument and sky coordinates, zero points for temporal and spatial scales, conversion between table values and physical values, and the instrument configuration. The table would contain such information as the coordinates (in the detector reference frame) of the individual events, the time of an event, the energy of an event, and the strength or intensity of an event. Example 7 in Appendix A shows a different FITS header for high energy astrophysics data, from the Advanced Satellite for Cosmology and Astrophysics (ASCA).

Section 6

Resources

6.1 The FITS Support Office

NASA has established the Flexible Image Transport System Support Office to assist FITS users and work with the astronomical community on the development and documentation of FITS. The FITS Support Office is a service of the Astrophysics Data Facility at NASA Goddard Space Flight Center.

The principal purposes of the FITS Support Office are to assist NASA projects in developing FITS data sets, to provide FITS information to the astronomical community and the public, and to take an active role in the evolution of FITS. Toward these ends, it has developed an on-line library of information, documents, and software, available either through the World Wide Web or through anonymous ftp. Documents developed by the FITS Support Office are available in printed form as well. The FITS Support Office also will answer questions about specific issues and problems not covered in the on-line material. For NASA projects, the FITS office also reviews whether proposed designs for FITS data sets and software conform to the FITS rules and recommended practices; designs from other organizations may be reviewed if they appear to be of wide community interest. From the kinds of inquiries received, the FITS office can develop a sense of the common questions about the existing standard formats, the problems with them, and what data structures, if any, these formats do not handle well. This experience enables the FITS office to provide recommendations on directions for FITS development to the IAU FITS Working Group and the regional FITS committees, and to work with developers of potential new extensions and conventions.

The FITS Support Office is coordinated by Dr. Barry M. Schlesinger, a member of the IAUFWG, whose research background is in the areas of stellar structure and evolution and variations in solar radiation. The FITS office can be reached by telephone at +1-301-286-2899.

6.1.1 On-line Information

The FITS Support Office maintains World Wide Web and ftp sites at Goddard Space Flight Center. The FITS Support Office home page is at `http://ssdoo.gsfc.nasa.gov/astro/fits/fits_home.html`. It describes the on-line material available from the FITS Support office and provides links both to the available hypertext material and to the material available at the ftp site `nssdc.gsfc.nasa.gov/pub/fits/`. It also links to other FITS resources on the Web. The ftp site has an `aareadme.doc` file that can be accessed directly, for those without Web connections.

The FITS Basics and Information page can be reached from the home page. It contains the following material:

- An overview of FITS, with a one-paragraph description and a discussion of how FITS evolves
- Discussion of FITS documents, both those available from the FITS Support Office and those available elsewhere
- Discussion of software packages, including software for various platforms that can display FITS images
- Discussion of other FITS resources available on-line
- Discussion of the FITS Support Office with contact information

Where possible, links are provided to the documents, software, and network locations described. There is also a plain text version, available at the ftp site. It provides URLs or other access information for the material described.

The FITS Support Office maintains the official current IAUFWG list of registered extensions, as its agent. This list is available in both hypertext and plain text forms.

The FITS Support Office documentation discussed in section 6.1.2 and the software and test files described in section 6.1.3 are available from the Web page or at the ftp site.

6.1.2 Documents

No current document precisely and formally describes the structure and syntax of FITS. The IAUFWG has endorsed the six FITS papers (Wells, Greisen, and Harten 1981; Greisen and Harten 1981; Grosbøl *et al.* 1988; Harten et al. 1988; Ponz, Thompson, and Muñoz 1994; Cotton, Tody, and Pence 1995) and the floating point and blocking agreements. By permission of the copyright holder, the European Southern Observatory, the FITS Support Office can furnish copies of the the first four FITS papers to non-profit organizations. The text of the Floating Point Agreement appears in section 3.1.2.3 of this *Guide*, and the Blocking Agreement is available electronically from the FITS Support Office at `ftp://nssdc.gsfc.nasa.gov/pub/fits/blocking94.txt`.

Largely because the format has evolved incrementally, there are a number of inconsistencies and ambiguities among the FITS papers, especially the first four. To provide a document with a clear and unambiguous specification of the rules of FITS, the NASA/Science Office of Standards and Technology (NOST) has developed a formal FITS codification, the *Definition of FITS* (NOST 1995). This NOST standard was developed by technical panels of astronomers chaired by R. Hanisch (STScI), chair of the Working Group on Astronomical Software of the American Astronomical Society. The FITS Support Office coordinator served as secretary of the panel, and the FITS office was responsible for writing the original text and for the bulk of the editing during the revision process.

Development of the standard follows a procedure developed at the NOST that follows the common practices of other standardization bodies. The technical panel develops an initial *Draft Standard*. When the panel has reached consensus on this *Draft Standard*, the document is made available to the astronomical community for comment, typically for a period of two months. Availability of the document for review is announced to the `sci.astro.fits` newsgroup and to the electronic mailing lists described in section 6.5. The American Astronomical

Society Executive Office also is notified. Special efforts are made to ensure review by members of the IAUFWG and the regional FITS committees. After the period for review is over, the technical panel reviews the comments and revises the standard in reponse. Replies are sent to all reviewers, describing the changes made by the technical panel in response to the review, and explaining the reasons when the recommended revision was not made. This cycle is repeated until the number of revisions recommended by the reviewers is so small that it is evident that the astronomical community has reached consensus.

The document that results from this process is called a *Proposed Standard*. This document is then submitted to a NOST accreditation panel, composed of the NOST Executive Board (the NOST Secretary, two other members from the Space Science Data Operations Office, and one from NASA Headquarters) and an outside astronomer. The role of this panel is to review the *process* of developing the standard: whether the steps taken to publicize the availability for review were adequate to reach the entire interested community, whether the technical panel replies addressed all the reviewers' points, and whether the resulting *Proposed Standard* represents a consensus of the community.

Version 1.0 of the standard covered the rules in the first four FITS papers and the floating point agreement. There were three technical panel review cycles. On June 18, 1993, the accreditation panel voted to approve the *Definition of FITS* as the NOST standard for the use of FITS. Questions about the specifications of units led to the development of a revision, version 1.1. There was only one review cycle; the number of comments was small and the community members present at the 1995 business meeting of the Working Group on Astronomical Software agreed that this lack of comments reflected general satisfaction with these changes. Version 1.1 was approved by the accreditation panel on June 14, 1995. The NOST *Definition* has been submitted to the chair of the IAUFWG as the start of the process of seeking endorsement of the NOST document as the international standard for FITS. In the ongoing electronic discussions of FITS, FITS rules are frequently quoted in the form in which they are expressed in the NOST document.

A technical panel has been formed to add the `IMAGE` and `BINTABLE` extensions and the blocking agreement to the NOST standard. Clarifications arising from community discussion of the FITS rules also will be made. The result of this process will be Version 2 of the *Definition of FITS*.

The standard is available in both printed and electronic form. Electronic copies are available in several forms: flat ASCII, uncompressed PostScript, compressed

PostScript,and LaTeX. Style (`nost.sty`) and index files are provided for use with the LaTeX form. The three forms have identical content, except that the ASCII version may not preserve the font usage in the other versions. The standard may be obtained from the anonymous ftp site in files of the form `fits_standard.*` through links starting at the FITS Support Office home page. When a *Draft Standard* is announced for community review, it is available in the LaTeXand PostScript forms; a text form is sometimes available.

The FITS Support Office maintains and updates this *User's Guide*, which is designed to provide a better introduction to FITS for the novice than the precise but dry standard. In addition to providing the rules, it explains them in more detail than the standard and discusses their motivation. It discusses recommended and discouraged practices and provides hints on how to use particular structures. Sample FITS headers illustrate the different formats. Some widely used conventions that are not specified in the formal rules of FITS are described. The *Guide* also discusses current FITS developments, including proposals for new rules or conventions. Copies are available in printed form and electronically in compressed or uncompressed PostScript and LaTeX, in files of the form `users_guide.*`. The style file `guide.sty` is provided for use with the LaTeX form.

6.1.3 Software and Test Files

The FITS Support Office has developed a simple C program called `headlist` to read and list all the FITS headers in a file, including extension headers as well as the primary header. It reads the file from standard input and writes to standard output. This program is useful when presented with a new and unfamiliar file. It does not evaluate the file and is not guaranteed to work if there are serious errors in the file, for example, if the size of the data in an extension is not correctly described by the values of `BITPIX`, `NAXIS`, the `NAXIS`n, `PCOUNT`, and `GCOUNT`.

The FITS Support Office has a software package written in C called the FITS Product Conformance Tester (FPCT), which validates conformance of primary HDUs to the FITS Standard. While limited to Basic FITS structures, its diagnostics are more descriptive than those of other packages; it is also more forgiving, as it attempts to continue past errors by making its best guess at what was intended rather than terminating on the first error. It is thus most useful for those new to FITS. The FPCT checks the primary header of a FITS file for the presence of all required keywords and their order. If it can retrieve information (values of `BITPIX`, `NAXIS`, and the `NAXIS`n) needed to read the primary data array, and the data in the array are integer, it will, at the user's option, produce an output file containing array members selected by the user. If the header is in conformance, it will print a message to that effect. If the header is not in conformance, but the software has inferred a data description, it will print a warning that retrieval of the data array may not be correct, with an indication of the severity of the problem. If the FPCT has been unable to retrieve the information needed to read the primary data array, it will print a diagnostic with information about the header errors and stop.

Both packages are available in the `software` subdirectory of the FITS anonymous ftp directories described in section 6.1.1 or can be obtained electronically by request from the FITS Support Office. The FPCT is accompanied by an instructions file. Read this file before using the software.

The `errtest` subdirectory contains a number of primary HDUs designed for checking the ability of FITS readers to cope with the unexpected. It includes one that is in compliance with the standard and a number of others that are identical except for deliberately inserted errors in syntax. One is simply a text extract, to test the behavior of the FITS reader when confronted with such a file, as might occur if such a file were mistakenly retrieved instead of a FITS file. An `aaareadme.doc` file describes the contents of this subdirectory. These files must be retrieved in *binary* format if they are to work properly. Many Web browsers interpret FITS files as text by default.

6.1.4 Contact Information

Printed copies of documentation are obtained from the Coordinated Request and User Support Office (CRUSO)

(Electronic mail) `request@nssdca.gsfc.nasa.gov`

(Telephone) +1-301-286-6695 8:00 A. M. - 4:30 P. M.

(FAX) +1-301-286-1635

(Postal)

Coordinated Request and User Support Office

Code 633
National Space Science Data Center
NASA Goddard Space Flight Center
Greenbelt MD 20771 USA

Times are U. S. Eastern time: GMT -0500 from the last Sunday in October through the first Saturday in April, -0400 the rest of the year. If no one is available to answer the telephone, messages can be left on voice mail.

Contact information for the FITS Support Office is as follows:

(Electronic mail) `fits@nssdca.gsfc.nasa.gov`

(Telephone) +1-301-286-2899

Approximate hours are 8:30 A. M. - 5:30 P. M. Eastern time; voice mail is available.

6.2 NRAO FITS Resources

An archive of FITS material is available at NRAO, at `http://fits.cv.nrao.edu/` or `ftp://fits.cv.nrao.edu/fits/` This site also supports a WAIS server named `nrao-fits` which has an index of all of the FITS-related text files in the archive; the file `nrao-fits.src` is available at `think.com` and at `ftp://fits.cv.nrao.edu/fits/wais-sources/nrao-fits.src`, as well as through the NRAO FITS Web site.

The `documents` directory contains a number of subdirectories with various FITS documents. The formats of the documents depend upon the size. Some small documents are available in ASCII text. Larger documents are in PostScript and often in LaTeX. The largest documents will also have gzip-compressed PostScript versions as well. The `standards` subdirectory contains copies of the NOST standard described in section 6.1.2, text of the agreement on physical blocking, and copies of preliminary drafts of the binary table and image extension papers. The `overviews` subdirectory contains electronic copies of this *User's Guide* and some FITS history. The `proposals` subdirectory is for proposals that are ready for consideration by the FITS committees. The `drafts` subdirectory contains drafts for extensions or conventions that may or may not be submitted to the FITS committees at a future time. The `wcs` subdirectory contains the current draft proposal for world coordinates conventions (Greisen and Calabretta 1996), draft conventions for world coordinates keywords that would be used in conjunction with this proposal, some earlier documents, and documents discussing map projections.

A directory called `os-support` is devoted to code for various environments. It contains some software, as well as copies of Usenet postings and mailing list announcements discussing other software. The information extends back over four years.

The `data` directory contains two subdirectories of FITS files. The `samples` subdirectory contains a set of FITS data files from different installations covering a number of disciplines, e.g., Hubble Space Telescope, IUE ultraviolet, radio VLBA and single dish measurements, and illustrating FITS conventions, e.g., world coordinates and grouping. The `tests` subdirectory has under it two subdirectories of test files. One, `ftt4b`, is a set of FITS files from the early 1980s, which contains a number of images from that era designed for testing FITS files and illustrating possible applications of the format, including the Mandrill to illustrate one way to store RGB. One file also provides a random groups example, which can be used for testing by anyone with a need to read older data in that format. The other subdirectory, `pg93`, contains test files specially designed for testing the ability of software to handle all standard FITS files, including the primary data array and all standard FITS extensions. In particular, one file contains an array with all the IEEE special values: infinity, NaN, and denormalized numbers. The data are artificial, with values specially selected for the purpose of validating the software's ability to read the data.

The `traffic` directory contains archives of FITS-related traffic from Usenet newsgroups and mailing list. It includes all postings to `sci.astro.fits` (the fitsbits mailing list), and FITS-related postings to other newsgroups. It also includes Frequently Asked Questions (FAQ) from selected Usenet newsgroups. FITS-related mailing lists and FITS-related messages from other mailing lists also are archived here.

W. Cotton has developed a family of FITS viewers for Microsoft Windows, Apple Macintosh, and Unix/X-Windows computer systems. These viewers have easy to use graphical controls and are suitable for use as external FITS image viewers for World Wide Web browsers. The programs contain extensive on-line documentation. NRAO distributes these viewers free of charge. Further information on the capabilities of the software and access to it are at `http://www.cv.nrao.edu/~bcotton/fitsview.html`.

6.3 HEASARC

The High Energy Astrophysics Science Archive Research Center (HEASARC) at Goddard Space Flight Center has been active in the development of FITS software and data set design. It has developed and continues to enhance packages of FITS subroutines and tools, which are publicly available and widely used. It has formulated a number of internal conventions, which are primarily related to high energy astrophysics but may have applications in other areas. A number of these conventions are discused in Section 5.6.1. HEASARC material is available through their World Wide Web page at `http://heasarc.gsfc.nasa.gov/`. It is also available at `ftp://legacy.gsfc.nasa.gov/`.

HEASARC has developed two major software packages. FITSIO is a machine-independent subroutine interface for reading or writing FITS files. By using these subroutines, the programmer can avoid having to deal directly with the internal details of the FITS file. FITSIO supports all the standard FITS file formats discussed in section 3. Some of the functions performed are opening and closing files, reading, writing or modifying header keywords, and reading or writing any element of a FITS data array or table. It also includes the VERIFITS program to verify that a FITS file conforms to the FITS standard. In addition to supporting the standard FITS formats, VERIFITS supports the variable length array convention discussed in section 5.2.1.

There are FITSIO versions in portable FORTRAN-77 and in ANSI C. Both versions run on Sun, DECstations, VAX/VMS, DEC Alpha (VMS and OSF/1), SGI, HP, NeXT, IBM PCs (DOS and Linux), Amiga PCs and Macintosh PCs; the FORTRAN version runs on Cray supercomputers and IBM mainframes as well. FITSIO requires random access I/O to the file and therefore does not support direct read or write access to sequential devices such as magnetic tape drives.

FITSIO source code and documentation are available through the Web from the `Software` item on the HEASARC menu bar or directly from `ftp://legacy.gsfc.nasa.gov/software/fitsio`.

The other package, FTOOLS, is a collection of utility programs, in ANSI Fortran or C, that allows the user to interactively create, examine, or modify the contents of FITS files, utilizing the FITSIO package. Perl5 scripts are available for combining several FTOOLS to perform complex tasks. FTOOLS supports all the standard FITS file formats discussed in Section 3.

A general FITS file viewer and editor called fv is included as a standard part of the FTOOLS distribution and is available as a standalone package as well. It can be used for viewing and plotting any FITS format data file. It has a graphical user interface (GUI) written in Tcl/Tk that provides spreadsheet-like widgets to display and edit data in any FITS table or image, a scrolling text window to display and edit FITS header keywords, and an image display that uses either the Smithsonian Astrophysical Observatory - The Next Generation (SAOtng) display tool or a custom designed Tk image widget. Users can also produce line plots of the values in two or more columns of a FITS table and export the plots to a PostScript file.

As of this writing, fv is supported on most Unix platforms; support for IBM and Macintosh PCs is planned.

The FTOOLS package, documentation, and installation instructions are available from the `Software` item on the HEASARC menu block or directly from `ftp://legacy.gsfc.nasa.gov/software/ftools`. The fv tool is available at the same location.

The HEASARC FITS Working Group (HFWG), a body within HEASARC established to support present and future multimission needs of the OGIP at NASA, and, in particular, to ensure that the FITS formats and keywords used do not violate the FITS standards, has developed a number of conventions and designs. While use of these conventions and designs is required only for OGIP, the high energy astrophysics community has been consulted during the development process, and many of these conventions are used elsewhere for high energy astrophysics FITS files and other FITS files. A number of these conventions have been described in section 5.6.1. Other conventions specify keywords and strings to be used for such information as quality flags, channel boundaries, exposure times, and units. Among the standard file formats are those for light curves and spectra. Complete documentation on the recommendations and formats that have been adopted by HFWG and on those that are under consideration can be found at heasarc.gsfc.nasa.gov/docs/heasarc/ofwg/ofwg_intro.html on the HEASARC main Web page or directly by ftp to ftp://legacy.gsfc.nasa.gov/.

The archive of traffic on the heafits mail exploder also is accessible from the HFWG page.

6.4 Some Additional Software Resources

FITS I/O software written in IDL is available as part of the IDL Astronomy User's Library, a central repository for general purpose astronomy procedures written in IDL, including procedures to convert between spherical coordinates and plane map coordinates. The library is not meant to be an integrated package but rather is a collection of procedures from which users can pick and choose for their own use. Submitted procedures are given a cursory testing but are basically stored in the library as submitted. The Astronomy User's Library is supported under the NASA Astrophysics Software Aids program. The material is available through idlastro.gsfc.nasa.gov/home_page.html or at ftp://idlastro.gsfc.nasa.gov (IP 128.183.57.82).

The Astronomical Data Center (ADC) at NASA/GSFC has developed a FITS Table Browser, which has been tailored specifically for use with the ADC CD-ROMs but may be used with other FITS ASCII tables. It reads standard FITS ASCII tables and allows the user to browse through them interactively and selectively display any field or record in a table. File extraction facilities permit writing all or part of the input table to disk in FITS or text file format. It is available at ftp://adc.gsfc.nasa.gov/pub/adc/software/browsers/ftb. See the file ftb_user_guide.txt for instructions on downloading, installation, and use.

The IUEDAC software (which is written in IDL) includes a FITS reader that can read primary HDUs and image extensions, ASCII tables, and binary tables, including binary tables with variable-length and multi-dimensional arrays. The accompanying FITS writer can write primary HDUs, image extensions, and binary tables including multi-dimensional but not variable length arrays, but it cannot write ASCII tables. The software is available from ftp://iuewww.gsfc.nasa.gov and runs on Unix, Ultrix, VMS, MacOS, and Windows 3.1 systems.

The National Center for Supercomputing Applications (NCSA) Hierarchical Data Format (HDF) group is developing a FITS server-side browser. Among the things it can do now are the following:

- Retrieve the images from a FITS file
- Allow viewing of a group of planes from a 3-D image
- Get summary information from the primary or extension headers
- Create an HTML table from an ASCII or binary table.

NCSA is also developing FITS–HDF conversion utilities. Because this site is evolving rapidly, the individual URLs have been changing, but it should be possible to find the FITS material by starting at `http://hdf.ncsa.uiuc.edu/`.

The three principal astronomical image processing software packages— AIPS, the ESO-Munich Interactive Data Analysis System (ESO-MIDAS), and IRAF—incorporate substantial support for FITS.

New software to display FITS images continues to be developed. Some nonproprietary packages are mentioned in one section of the FITS Basics and Information. A general discussion of software to convert between formats, including FITS, is in part 2 of the Graphics File Format FAQ, which is periodically posted to the newsgroups `comp.graphics.misc`, `comp.answers` and `news.answers`. It is also archived at `ftp://rtfm.mit.edu/pub/usenet/` and `http://www.cis.ohio-state.edu/hypertext/faq/usenet-faqs/`. The MIT site is often busy. The Ohio State site is occasionally reorganized with changes in the lower level of the URL.

6.5 Other Network Resources

A Usenet newsgroup, `sci.astro.fits`, is devoted to FITS. For those who prefer, there is an associated `fitsbits` mailing list. To subscribe, send electronic mail to `fitsbits-request@fits.cv.nrao.edu`. This newsgroup/mailing list is the principal forum for FITS discussions and announcements. Developments in the evolution of new extensions will appear there: reservation of type names, information on where to find particular proposals, discussions of the proposals, and notification of actions by the regional and international FITS bodies. The FITS Support Office uses this forum to announce developments such as progress on updates to the standard or a new edition of this *User's Guide*. Developers of publicly available FITS software packages also announce their packages here. In addition to the announcements, there is general discussion of FITS practices. The FITS Support Office maintains a monthly posting describing where FITS information can be obtained, which is posted as well to the newsgroups `sci.answers` and `news.answers` and is consequently automatically available at Usenet FAQ archives.

The exploder `dishfits@nrao.edu` was created to facilitate discussions on the subject of FITS formats for single dish radio astronomy. This exploder is now gatewayed to the newsgroup `adass.fits.dishfits`. To subscribe send electronic mail to `dishfits-request@nrao.edu`. Traffic on this exploder is archived at the NRAO site.

HEASARC has set up an electronic mail listserver called `heafits` for FITS issues that are specific to high energy astrophysics. To *subscribe* to the `heafits` list, send the following one-line electronic mail message to `listserv@legacy.gsfc.nasa.gov`:

`subscribe heafits` Your Name

where "Your Name" is your actual first and last names. Messages to the actual mailing list should be sent to `heafits@legacy.gsfc.nasa.gov`. An archive is at `http://heasarc.gsfc.nasa.gov/listserv/heafits/maillist.html`. There is a link to it from the HFWG page.

An anonymous ftp directory of FITS material is maintained at the European Southern Observatory. Its address is `ftphost.hq.eso.org` (IP 134.171.8.4).

Appendix A

Examples of FITS Headers

Example 1: VLA Image Header

(Note: The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
00000000011111111112222222222333333333344444444445555555555666666666
12345678901234567890123456789012345678901234567890123456789012345678
SIMPLE  =                    T /
BITPIX  =                   16 /
NAXIS   =                    4 /
NAXIS1  =                 2048 /
NAXIS2  =                 1024 /
NAXIS3  =                    1 /
NAXIS4  =                    1 /
EXTEND  =                    T / TABLES FOLLOWING MAIN IMAGE
BLOCKED =                    T / FILE MAY BE BLOCKED
OBJECT  = '3C405   '           / SOURCE NAME
TELESCOP= '        '           /
INSTRUME= '        '           /
OBSERVER= 'PERL    '           /
DATE-OBS= '27/10/82'           /OBSERVATION START DATE DD/MM/YY
DATE-MAP= '14/07/83'           /DATE OF LAST PROCESSING DD/MM/YY
BSCALE  =    7.04625720812E-05 /REAL = FITS_VALUE * BSCALE + BZERO
```

```
BZERO   =    2.18688869476E+00 /
BUNIT   = 'JY/BEAM '           /UNITS OF FLUX
EPOCH   =      1.950000000E+03 /EPOCH OF RA DEC
DATAMAX =      4.495524406E+00 /MAX PIXEL VALUE
DATAMIN =     -1.217470840E-01 /MIN PIXEL VALUE
CTYPE1  = 'RA---SIN'           /
CRVAL1  =    2.99435165226E+02 /
CDELT1  =     -4.166666986E-05 /
CRPIX1  =      1.024000000E+03 /
CROTA1  =      0.000000000E+00 /
CTYPE2  = 'DEC--SIN'           /
CRVAL2  =    4.05961940065E+01 /
CDELT2  =      4.166666986E-05 /
CRPIX2  =      5.130000000E+02 /
CROTA2  =      0.000000000E+00 /
CTYPE3  = 'FREQ    '           /
CRVAL3  =    4.86635000000E+09 /
CDELT3  =      1.250000000E+07 /
CRPIX3  =      1.000000000E+00 /
CROTA3  =      0.000000000E+00 /
CTYPE4  = 'STOKES  '           /
CRVAL4  =    1.00000000000E+00 /
CDELT4  =      1.000000000E+00 /
CRPIX4  =      1.000000000E+00 /
CROTA4  =      0.000000000E+00 /
HISTORY UVLOD /DATA BASE CREATED BY USER 76 AT 14-JUL-1983`10:17:08
HISTORY UVLOD OUTNAME='CYGA        ' OUTCLASS='XY    '
HISTORY UVLOD OUTSEQ=    1    OUTDISK= 3
ORIGIN  = 'AIPSNRAO VLA VAX3  '        /
DATE    = '19/08/83'                   / FILE WRITTEN ON DD/MM/YY
HISTORY AIPS   IMNAME='CYGA    '      IMCLASS='IMAP  ' IMSEQ=   1 /
HISTORY AIPS   USERNO=   76            /
END
```

(23 card images of ASCII blanks)

Discussion of Example 1 (VLA Image Header)

This header describes an image of 3C405 (OBJECT keyword) on 27 October 1982 (DATE-OBS keyword) by an observer with the designation "Perl" (OBSERVER keyword). The data matrix has four axes: right ascension, declination, frequency, and polarization (NAXISn and CTYPEn keywords). The matrix here contains an image for only one frequency and polarization. The sine projection from the sky sphere to the tangent plane is used. The coordinates are equinox 1950 (note the use of the EPOCH keyword, now discouraged; use EQUINOX instead). Right ascension is in *degrees*, extending from approximately 299.47779 degrees at point (pixel) 1 to approximately 299.3925 at point 2048, with the value 299.435165226 at point 1024 (CRVAL1, CDELT1, CRPIX1). Declination, in degrees, runs from approximately 40.57486 to approximately 40.61848, with the value 40.5961940065 at pixel 513 (CRVAL2, CDELT2, CRPIX2). The measurement is at 4866.35 MHz (CRVAL3). CDELT3 is not significant here; there is only one point. The values are stored in 16-bit integer format (BITPIX keyword). Values in the matrix are converted to flux in Janskys per beam (BUNIT keyword) by multiplying by 7.04625720812 (BSCALE keyword) and adding 2.18688869476 (BZERO keyword), corresponding to a range of data matrix values of -0.121747084 (physical value) or -32764 (in file) to 4.495524406 (physical value) or 32764 (in file) Janskys per beam. The EXTEND keyword has the value T, indicating that this Basic FITS unit may be followed by extensions; the comment notes that it is followed by tables. The use of the HISTORY keywords may appear to be analogous to the HIERARCH proposal; however, the embedded "keywords" have meaning only within AIPS; to the rest of the world they are comments and can be treated as such.

Example 2: M87 and Jet (KPNO)

(Note: The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
00000000011111111112222222222333333333344444444445555555555666666666677 7
12345678901234567890123456789012345678901234567890123456789012345678901 2
SIMPLE  =                    T  /  FITS TAPE WRITTEN AT KPNO, 04/18/80.
BITPIX  =                   32  /  4-BYTE, 2-S COMPLEMENT INTEGERS.
NAXIS   =                    2  /  NUMBER OF AXES.
NAXIS1  =                  256  /  NUMBER OF PIXELS PER ROW.
NAXIS2  =                  256  /  NUMBER OF ROWS.
BSCALE  =          1.00000E-06  /  PHYSICAL=INTEGER*BSCALE+BZERO.
BZERO   =                   0.  /
BLANK   =          -2147483648  /  INTEGER VALUE FOR BLANK PIXEL.

IPPS-RF = 'D      /013'         /  RASTER LFN/RASTER ORDINAL.
IPPS-ID = 'N4486 NUCLEUS AND JET 4350 [9 BAD PIXELS BLANKED]        ' /
IPPS-B/P=                   30  /  BITS/PIXEL OF IPPS RASTER.
IPPS-MIN=         3.700018E-02  /  MINIMUM VALUE IN RASTER.
IPPS-MAX=              1.19825  /  MAXIMUM VALUE IN RASTER.
ORIGIN  = 'KPNO    -- WFITS OF 04/17/80.' /
DATE    = '18/04/80'                            /
TIME    = '10.11.54'
JOBNAME = 'DCWGF27 '
COMMENT   THIS FILE TESTS THE 32-BIT PIXEL FORMAT.
COMMENT   IT ALSO TESTS THE BLANK PIXEL CONVENTION. THERE ARE 9 BLANKS IN THIS
COMMENT   IMAGE. THEY ARE A DETECTOR BLEMISH AREA. NOT ALL OF THE BLEMISH HAS
COMMENT   BEEN BLANKED.

OBJECT  = 'NGC4486  =M87 AND JET' /
TELESCOP= 'MAYALL4M'           / MAYALL 4-METER TELESCOPE AT KITT PEAK.
INSTRUME= 'VIDEOCAM - MAYALL 4M CASSEGRAIN' /

COMMENT   THE KPNO "VIDEO CAMERA" IS AN INTEGRATING DIGITAL TELEVISION SYSTEM
COMMENT   WHICH USES AN  RCA ISIT VIDICON TUBE AND ACCUMULATES THE IMAGE IN
COMMENT   A 256-SQUARE 20-BIT RAM DURING THE INTEGRATION.
COMMENT   THIS IMAGE HAS BEEN CORRECTED FOR THE DARK CURRENT AND GAIN VARIATIONS
COMMENT   ACROSS THE FIELD. I.E., THE DARK IMAGE WAS SUBTRACTED, AND THE FLAT
COMMENT   FIELD IMAGE WAS DIVIDED INTO THE RAW IMAGE. BECAUSE THE FLAT WAS NOT
COMMENT   NORMALIZED IN THIS CASE, THE RESULTING RATIO IMAGE IS OF ORDER UNITY.
COMMENT   THIS IS A QUANTUM NOISE LIMITED IMAGE IN RELATIVE INTENSITY UNITS.
COMMENT   THE GEOMETRY OF THIS IMAGE HAS NOT BEEN CORRECTED FOR THE PINCUSHION
COMMENT   DISTORTION OF THE VIDEO CAMERA.

END
```

(33 card images of ASCII blanks)

Discussion of Example 2 (M87 and Jet)

This header is based on the NRAO anonymous ftp collection of FITS files. There have been some minor revisions, none of which affect required or reserved keywords, to bring the style into conformance with recommended practice. Internal comments explain most of the keywords in detail. The data following this header will consist of a 256 x 256 (`NAXIS1`, `NAXIS2` keywords) matrix of 32-bit integers (`BITPIX` keyword). To derive the physical value represented, the integer on the original FITS tape must be multiplied by 10^{-6} (`BSCALE` keyword). Array members (pixels) that contain a -2147483648 are to be treated as not containing real data values (`BLANK` keyword). The matrix represents KPNO video camera (`INSTRUME` keyword) pictures of M87 (`OBJECT` keyword) taken on the Mayall 4 meter telescope (`TELESCOP` keyword). The tape was created on 18 April 1980 (`DATE` keyword). The date of the observations is not given in this header; if it were, there would be a `DATE-OBS` keyword. Note that even though some of the character string values are more than eight characters long, it is possible to obtain the essential information from reading only the first eight characters. There are also five keywords that are neither required nor reserved FITS keywords, those beginning with `IPPS-`. They illustrate how new keywords can be created for a FITS data set. Note the blank lines. The blanks in columns 1–8 indicate that columns 9–80 are to be treated as comment; in this case, they are blank.

Example 3: ASCII Table

(Note: The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
00000000011111111112222222222333333333344444444445555555555666666666
12345678901234567890123456789012345678901234567890123456789012345678
SIMPLE  =                    T / STANDARD FITS FORMAT (REV OCT 1981)
BITPIX  =                    8 / CHARACTER INFORMATION
NAXIS   =                    0 / NO IMAGE DATA ARRAY PRESENT
EXTEND  =                    T / THERE IS AN EXTENSION
ORIGIN  = 'ESO     '           / EUROPEAN SOUTHERN OBSERVATORY
OBJECT  = 'SNG - CAT.     '    / THE IDENTIFIER
DATE    = '27/ 5/84'           / DATE THIS TAPE WRITTEN DD/MM/YY
END
```

(28 card images of ASCII blanks)

```
XTENSION= 'TABLE   '           / TABLE EXTENSION
BITPIX  =                    8 / CHARACTER INFORMATION
NAXIS   =                    2 / SIMPLE 2-D MATRIX
NAXIS1  =                   98 / NO. OF CHARACTERS PER ROW
NAXIS2  =                   10 / NO. OF ROWS
PCOUNT  =                    0 / RANDOM PARAMETER COUNT
GCOUNT  =                    1 / GROUP COUNT
TFIELDS =                    7 / NO. OF FIELDS PER ROW
TTYPE1  = 'IDEN.   '           / NAME OF ROW
TBCOL1  =                    1 / BEGINNING COLUMN OF THE FIELD
TFORM1  = 'E14.7   '           / FORMAT
TNULL1  = '        '           / NULL VALUE
TTYPE2  = 'RA      '           / NAME OF ROW
TBCOL2  =                   15 / BEGINNING COLUMN OF THE FIELD
TFORM2  = 'E14.7   '           / FORMAT
TNULL2  = '        '           / NULL VALUE
TTYPE3  = 'DEC     '           / NAME OF ROW
TBCOL3  =                   29 / BEGINNING COLUMN OF THE FIELD
TFORM3  = 'E14.7   '           / FORMAT
TNULL3  = '        '           / NULL NAME
```

```
TTYPE4  = 'TYPE    '           / NAME OF ROW
TBCOL4  =                   43 / BEGINNING COLUMN OF THE FIELD
TFORM4  = 'E14.7   '           / FORMAT
TNULL4  = '        '           / NULL VALUE
TTYPE5  = 'D25     '           / NAME OF ROW
TBCOL5  =                   57 / BEGINNING OF COLUMN OF THE FIELD
TFORM5  = 'E14.7   '           / FORMAT
TNULL5  = '        '           / NULL VALUE
TTYPE6  = 'INCL.   '           / NAME OF ROW
TBCOL6  =                   71 / BEGINNING COLUMN OF THE FIELD
TFORM6  = 'E14.7   '           / FORMAT
TNULL6  = '        '           / NULL VALUE
TTYPE7  = 'RV      '           / NAME OF ROW
TBCOL7  =                   85 / BEGINNING COLUMN OF THE FIELD
TFORM7  = 'E14.7   '           / FORMAT
TNULL7  = '        '           / NULL VALUE
END
```

(35 card images of ASCII blanks)

Discussion of Example 3 (A Galaxy Catalog in an ASCII Table)

This pair of headers illustrates the case where all the data are in an extension. Like Example 2, it is from the NRAO ftp collection. It is a catalog of galaxies, probably of parent galaxies of supernovae (`OBJECT` keyword) originating at the European Southern Observatory (`ORIGIN` keyword and comments). It was created on 27 May 1984 (`DATE` keyword) and thus may not show how ESO-MIDAS would now write an ASCII table, but it can still illustrate the format. The primary header carries two additional pieces of information: that there is no primary data array (value of zero for `NAXIS` keyword) and that an extension *may* be present (`T` value for `EXTEND` keyword). The comment following the `EXTEND` keyword notes that an extension *is* present. The `END` card image is followed by 28 card images consisting of ASCII blanks, to fill the record to 2880 bytes. The string `XTENSION= 'TABLE    '` at the start of the next record introduces the header of an ASCII table extension. The `BITPIX`, `NAXIS`, `PCOUNT`, and `GCOUNT` keywords have the values required for an ASCII table. The `NAXIS`*n* keywords show that it is ten rows long with 98 characters per row. Each row consists of seven fields (`TFIELDS` keyword), all in FORTRAN format E14.7 (`TFORM`*n* keywords), with no blanks between (`TBCOL`*n* keywords). A null value in any field is represented by a blank (`TNULL`*n* keywords). The seven fields in each row represent a galaxy identifier, right ascension, declination, type, diameter of the galaxy at the 25 mag/arcsec level, inclination, and radial velocity, respectively (`TTYPE`*n* keywords).

Example 4: Binary Table Containing Spectra

(Note: The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
0000000001111111111222222222233333333334444444444555555555566666666666
1234567890123456789012345678901234567890123456789012345678901234567890
XTENSION= 'BINTABLE'         T / BINARY TABLE EXTENSION
BITPIX  =                    8 / BYTES
NAXIS   =                    2 / A TABLE IS A 2D MATRIX
NAXIS1  =                 4134 / BYTES PER ROW
NAXIS2  =                   20 / NUMBER OF ROWS
PCOUNT  =                    0 / NO 'PARAMETERS'
GCOUNT  =                    1 / ONE TABLE
TFIELDS =                    6 / 6 FIELDS PER ROW
EXTNAME = 'SPECTRUM'           / SAMPLE OF BINARY TABLE WITH SPECTRUM
EXTVER  =                    2 / SECOND VERSION OF TABLE
TFORM1  = '1J      '           / FIELD 1 - 32 BIT INTEGER
TTYPE1  = 'SCAN    '           / SCAN NUMBER
TFORM2  = '20A     '           / FIELD 2 - 20 CHARACTER STRING
TTYPE2  = 'SOURCE  '           / SOURCE NAME
TFORM3  = '1D      '           / FIELD 3 - DOUBLE PREC FLOATING POINT
TTYPE3  = 'BASEFREQ'           / REFERENCE CHANNEL
TUNIT3  = 'HZ      '           / FREQUENCIES IN HERTZ
TFORM4  = '1E      '           / FIELD 4 - SINGLE PREC FLOATING POINT
TTYPE4  = 'DELTFREQ'           / FREQUENCY INCREMENT
TUNIT4  = 'HZ      '           / FREQUENCIES IN HERTZ
TFORM5  = '1I      '           / FIELD 5 - 16 BIT INT
TTYPE5  = 'NCHAN   '           / NUMBER OF CHANNELS
TFORM6  = '1024E   '           / FIELD 6 1024 SNGL PREC FLT PT VALUES
TTYPE6  = 'SPECTRUM'           / MEASURED SPECTRUM
TUNIT6  = 'JY      '           / JANSKYS
COMMENT   FIELD 6 IS A VECTOR
END
```

(9 card images of ASCII blanks)

Discussion of Example 4 (A Binary Table Containing Spectra)

This extension header is the one that would be used for a structure designed by D. Wells to illustrate an efficient use of a binary table to hold spectra. The spectrometer for this illustration has 1024 channels, and the data are in floating point form. This design is, intentionally, grossly simplified for the sole purpose of illustration. Wells notes that real spectral observations would probably require 50 table columns in each row in addition to the spectra. The precise column labels are not part of the proposal; if such a structure were to be adopted, an *ad hoc* group would work out the details. Also, the data types used might vary among installations.

In this example, the primary header is not included, but, as with the ASCII table extension example, there must be one to start the data set. Other extensions might be present as well, so this extension isn't necessarily going to be the second header-data unit on the data set. The `XTENSION= 'BINTABLE'` string at the start of the record indicates that the header is for a binary table extension. The `BITPIX`, `NAXIS`, and `GCOUNT` keywords have the values required for a binary table extension. The value of 0 for `PCOUNT` shows that there are no heap or other data following the table. To describe one spectrum, 4134 bytes are needed (`NAXIS1`); this particular table has 20 spectra (`NAXIS2`). The `NAXIS1` value is equal to the sum of the number of bytes required by the different fields. Each row of the table can hold a 1024-channel spectrum and five additional items of information to describe it: scan number, source ID, frequency of reference channel, channel-to-channel difference, and the total number of channels. Frequencies are in Hertz (`TUNIT3` and `TUNIT4` keywords). There is room in field 6 for 1024 channels; there could be fewer. The title or name of the table is SPECTRUM (`EXTNAME` keyword); this table could be the second version or perhaps the second spectrum in the FITS file (`EXTVER` keyword).

Example 5: ADC FITS Table Header for AGK3 Catalog

(Note: The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
000000000111111111122222222223333333333444444444455555555556666666666777
123456789012345678901234567890123456789012345678901234567890123456789012
XTENSION= 'TABLE   '           / Table extension
BITPIX  =                    8 / Character data
NAXIS   =                    2 / Simple 2-D matrix
NAXIS1  =                   80 / Number of characters per record
NAXIS2  =               183145 / Number of records in the table
PCOUNT  =                    0 / No "random" parameters
GCOUNT  =                    1 / Only one group
TFIELDS =                   22 / Number of data fields per record
EXTNAME = 'AGK3    '           / AGK3 Star Catalogue of Positions
AUTHOR  = 'Dieckvoss, W.'
REFERENC= '1975, Hamburg-Bergedorf'
DATE    = '15/09/88'           / Date FITS file verified (dd/mm/yy)

HISTORY Source Catalog Reference:  AGK3 Star Catalogue of Positions and Proper
HISTORY Motions North of -2.5 Degrees Declination, Dieckvoss, W. (in
HISTORY collaboration with H. Kox, A. Guenther, and E. Brosterhaus) 1975,
HISTORY Hamburg-Bergedorf.
HISTORY
HISTORY A copy of "Documentation for the Machine-Readable Version of the
HISTORY AGK3 Star Catalogue of Positions and Proper Motions North of -2.5
HISTORY Degrees Declination (Dieckvoss and Collaborators 1975)", Warren Jr.,
HISTORY W.H. 1984, NSSDC/WDC-A-R&S 84-06, should accompany any computer-
HISTORY readable version of this catalog.

TTYPE1  = 'AGK3    '           / AGK3 zone and sequential number
TBCOL1  =                    1 / Start column
TFORM1  = 'A7      '           / Fortran format

TTYPE2  = 'AGK3_COMP'          / Component identification
TBCOL2  =                    8 / Start column
TFORM2  = 'A1      '           / Fortran format
TNULL2  = '0       '           / Null value

TTYPE3  = 'PMAG    '           / Photographic magnitude
TBCOL3  =                    9 / Start column
TFORM3  = 'I3      '           / Fortran format
TUNIT3  = 'mag     '           / Units are magnitudes
```

```
TSCAL3  =                  0.1 / Scale factor

TTYPE4  = 'SPTYPE  '           / Spectral type from HD, Yale, or McCormick
TBCOL4  =                   12 / Start column
TFORM4  = 'A2      '           / Fortran format

TTYPE5  = 'RAH     '           / Hours RA, epoch, equinox B1950
TBCOL5  =                   14 / Start column
TFORM5  = 'I2      '           / Fortran format
TUNIT5  = 'h       '           / Units are hours

TTYPE6  = 'RAM     '           / Minutes RA, epoch, equinox B1950
TBCOL6  =                   16 / Start column
TFORM6  = 'I2      '           / Fortran format
TUNIT6  = 'min     '           / Units are minutes of time

TTYPE7  = 'RAS     '           / Seconds RA, epoch, equinox B1950
TBCOL7  =                   18 / Start column
TFORM7  = 'I5      '           / Fortran format
TUNIT7  = 's       '           / Units are seconds of time
TSCAL7  =                0.001 / Scale factor

TTYPE8  = 'DECSIGN '           / Sign DEC, epoch, equinox B1950
TBCOL8  =                   23 / Start column
TFORM8  = 'A1      '           / Fortran format

TTYPE9  = 'DECD    '           / Degrees DEC, epoch, equinox B1950
TBCOL9  =                   24 / Start column
TFORM9  = 'I2      '           / Fortran format
TUNIT9  = 'deg     '           / Units are degrees

TTYPE10 = 'DECM    '           / Minutes DEC, epoch, equinox B1950
TBCOL10 =                   26 / Start column
TFORM10 = 'I2      '           / Fortran format
TUNIT10 = 'arcmin  '           / Units are minutes of arc

TTYPE11 = 'DECS    '           / Seconds DEC, epoch, equinox B1950
TBCOL11 =                   28 / Start column
TFORM11 = 'I4      '           / Fortran format
TUNIT11 = 'arcsec  '           / Units are seconds of arc
TSCAL11 =                 0.01 / Scale factor

TTYPE12 = 'NUMOBS  '           / Number of photog. obs. used to determine pos.
TBCOL12 =                   32 / Start column
TFORM12 = 'I2      '           / Fortran format

TTYPE13 = 'EPOCH   '           / (Epoch of AGK3 position) - 1900
TBCOL13 =                   34 / Start column
TFORM13 = 'I4      '           / Fortran  format
```

```
TUNIT13 = 'yr      '           / Units are years
TSCAL13 =                 0.01 / Scale factor
TZERO13 =               1900.0 / Zero offset

TTYPE14 = 'PM_RA_COSD'         / Annual proper motion in RA times Cos(delta)
TBCOL14 =                   38 / Start column
TFORM14 = 'I5      '           / Fortran format
TUNIT14 = 'arcsec/yr'          / Units are arcseconds per year
TSCAL14 =                0.001 / Scale factor

TTYPE15 = 'PM_DEC  '           / Annual proper motion in DEC
TBCOL15 =                   43 / Start column
TFORM15 = 'I5      '           / Fortran format
TUNIT15 = 'arcsec/yr'          / Units are arcseconds per year
TSCAL15 =                0.001 / Scale factor
COMMENT PM_DEC: If the calculated value for PM_RA_COSD or PM_DEC exceeded the
COMMENT PM_DEC: capacity of the field, a value of "+9999" is given.

TTYPE16 = 'EPOCH_DIFF'         / Difference in epoch (AGK3 - AGK2)
TBCOL16 =                   48 / Start column
TFORM16 = 'I4      '           / Fortran format
TUNIT16 = 'yr      '           / Units are years
TSCAL16 =                 0.01 / Scale factor

TTYPE17 = 'BD      '           / Bonner Durchmusterung number
TBCOL17 =                   52 / Start column
TFORM17 = 'A7      '           / Fortran format

TTYPE18 = 'BD_COMP '           / Numerical code for component of multiple system
TBCOL18 =                   59 / Start column
TFORM18 = 'I1      '           / Fortran format
TNULL18 = '0       '           / Null value

TTYPE19 = 'DISC_SUM'           / Sum of discrepancy codes, see comment
TBCOL19 =                   60 / Start column
TFORM19 = 'I2      '           / Fortran format
COMMENT DISC_SUM: The sum of discrepancy codes, or the sum of 2**V, where
COMMENT DISC_SUM: V = 0 (BD number), 1 (PMAG and/or spectral type), 2 (RA of
COMMENT DISC_SUM: AGK2), 3 (DEC of AGK2), 4 (RA of AGK3), 5 (DEC of AGK3).

TTYPE20 = 'PM_RA   '           / Annual proper motion in RA
TBCOL20 =                   62 / Start column
TFORM20 = 'I6      '           / Fortran format
TUNIT20 = 's/yr    '           / Units are seconds of time per year
TSCAL20 =               0.0001 / Scale factor

TTYPE21 = 'RA_RESID'           / Residual value in RA
TBCOL21 =                   68 / Start column
TFORM21 = 'I6      '           / Fortran format
```

```
TUNIT21 = 'arcsec  '           / Units are seconds of arc
TSCAL21 =               0.0001 / Scale factor

TTYPE22 = 'DEC_RESID'          / Residual value in DEC
TBCOL22 =                   74 / Start column
TFORM22 = 'I6      '           / 6 Fortran format
TUNIT22 = 'arcsec  '           / Units are seconds of arc
TSCAL22 =               0.0001 / Scale factor

END
```

(1 card image of ASCII blanks)

Discussion of Example ADC FITS Table Header for AGK3 Catalog

This ASCII table (XTENSION keyword) header was created by L. Brotzman on September 15, 1988 (DATE keyword) as an illustration of the FITS table headers that would appear on an ADC CD-ROM and were distributed with a request for comments. It is for the AGK3 catalog (HISTORY keywords; also EXTNAME) as compiled by W. Dieckvoss (AUTHOR) and collaborators (HISTORY), published in 1975 by the Hamburg-Bergedorf Observatory (REFERENC). The same catalog had been used for the example header in the FITS ASCII table paper (Harten *et al.* 1988).

The values of BITPIX, NAXIS, PCOUNT, and GCOUNT are those required by the rules for ASCII tables. Each row consists of 22 fields (TFIELDS) and is 80 characters long (NAXIS1). The sum of the field sizes specified in the TFORM*n* keywords, where *n* refers to the field number, is 80, the same as the value of NAXIS1; thus, for this table, the fields fill the entire row. There are 183145 rows (NAXIS2). The values of the TBCOL*n* keywords show where each column starts, the TTYPE*n* keywords give a column title, usually explained in the comment field of the same keyword, and the TFORM*n* keywords the FORTRAN format of the data in the field. Some keywords appear in only some fields: TUNIT*n* keywords provide units, all in lower case in this table although no special significance is to be ascribed to the case, TSCAL*n* and TZERO*n* keywords show where the value in the FITS table must be transformed to yield the physical value; and TNULL*n* gives the null string where needed. Blank lines are treated as comments, because the keyword field is blank.

The following list describes the contents of the fields, as described by the header keywords.

1. AGK3 starts in column 1 and contains a seven-character string giving the declination zone and sequence number within the zone of the star.

2. AGK3_COMP consists of a single character in column 8 giving a component identification for a member of a multiple star system. The string "0" (zero) represents a single star.

3. PMAG starts in column 9 and contains a three-digit integer giving the photographic magnitude of the star in units of tenths of a magnitude. The

value in the field must be multiplied by 0.1, the value of `TSCAL3`, to obtain the physical quantity in units given by `TUNIT3`.

4. `SPTYPE` starts in column 12 and contains a two-character string giving the spectral type.

5. `RAH` starts in column 14 and contains a two-digit integer with the hours of right ascension of the position of the star, equinox 1950, Besselian.

6. `RAM` starts in column 16 and contains a two-digit integer, with the minutes of time of right ascension of the position of the star, to be added to the hours of the previous field, equinox 1950, Besselian.

7. `RAS` starts in column 18 and contains a five-digit integer, which is 1000 times the number of seconds of time of the right ascension of the position of the star, to be added to the hours and minutes of the previous fields, equinox 1950, Besselian. This design allows storage in integer fields of right ascension accurate to 0.001 second of time.

8. `DECSIGN` is a single character in column 23 giving the sign of the declination. Note that, as discussed in connection with the `TFORM`n keywords in section 3.4.1, the sign is in a separate field, rather than incorporated into the value of the declination, in order to express the distinction between -0 and $+0$.

9. `DECD` starts in column 24 and contains a two-digit integer with the absolute value of the number of degrees of declination of the position the star, equinox 1950, Besselian.

10. `DECM` starts in column 26 and contains a two-digit integer, with the minutes of arc of the declination of the star, to be added to the degrees of the previous field, equinox 1950, Besselian.

11. `DECS` starts in column 28 and contains a four-digit integer, which is 100 times the number of seconds of arc of declination of the star to be added to the degrees and minutes of the previous fields, equinox 1950, Besselian This design allows storage in integer fields of declination accurate to 0.01 second of arc.

12. `NUMOBS` starts in column 32 and contains a two-digit integer giving the number of photographic observations used to determine the position.

13. `EPOCH` starts in column 34 and contains a four-digit integer giving the epoch—when the measurement was made—of the AGK position in

hundredths of years after 1900; to get the epoch, the number in this field is multiplied by 0.01 and added to 1900. Because the `EPOCH` keyword has been used in the past to give the equinox of the coordinate system (section 3.1.1.2), it generally better to use a different keyword (`DATE-OBS`, `MJD-OBS`) for the time of observation.

14. `PM_RA_COSD` starts in column 38 and is a five-digit integer containing the product of the annual proper motion in right ascension and the cosine of the declination, in units of thousandths of arcseconds/year, allowing this five-digit field to contain values accurate to 0.001 arcsec/year. Applying the scaling transformation yields values that are in units of arcseconds/year, the value of `TUNIT14`.

15. `PM_DEC` starts in column 43 and is a five-digit integer containing the annual proper motion in declination, in units of thousandths of arcseconds/year, allowing this five-digit field to contain values accurate to 0.001 arcsec/year.

16. `EPOCH_DIFF` starts in column 48 and is a four-digit integer giving the difference between the epoch of the AGK3 position and the epoch of the AGK2 position in hundredths of years.

17. `BD` starts in column 52 and contains a seven-character string giving the *Bonner Durchmusterung* number of the star.

18. `BD_COMP` consists of a one digit integer in column 59 giving a numerical code identification of a component of a multiple star system. The string "0" is the null value that would be in the field if the star is single. As discussed in section 3.4.2, the value of the `TNULL18` keyword is a *character string* and need not be readable in the integer format specified by the `TFORM18` keyword.

19. `DISC_SUM` is a two-digit integer starting in column 60 containing a sum that expresses in one number which discrepancy codes are present. A discrepancy code flags a value that is inconsistent with that in another catalog or is otherwise considered unreliable. The sum is $\sum_V 2^V$ where V are the discrepancy codes for the particular position; all values from 0 to 5 are possible. The meaning is explained by the `COMMENT` keywords immediately following the other column 19 keywords; the `DISC_SUM` following the `COMMENT` allows the text to be associated with the appropriate column even if the keywords are rearranged by the reader or a new table is created using some of the data in the current table and the column number changes. You could also think of the `DISC_SUM` as a hierarchy similar to those in the `HIERARCH` proposal discussed in section 5.7.

20. PM_RA is a six-digit integer starting in column 62 containing the annual proper motion in right ascension, in ten-thousandths of seconds of time per year.

21. RA_RESID is a six-digit integer starting in column 68 containing the difference between the ADK3 right ascension and the ADK2 right ascension in ten-thousandths of seconds of arc, corrected for changes in position due to proper motion.

22. DEC_RESID is a six-digit integer starting in column 74 containing the difference between the ADK3 declination and the ADK2 declination in ten-thousandths of seconds of arc, corrected for changes in position due to proper motion.

Example 6: DIRBE FITS Headers

(The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
00000000011111111112222222222333333333344444444445555555555666666666777
12345678901234567890123456789012345678901234567890123456789012345678909012

SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   32 / number of bits per data pixel
NAXIS   =                    0 / number of data axes
EXTEND  =                    T / FITS dataset may contain extensions
DATE    = '28/09/94'           / FITS file creation date (dd/mm/yy)
DATE-MAP= '10/07/94'           / Date of original file creation (dd/mm/yy)
ORIGIN  = 'CDAC    '           / Cosmology Data Analysis Center
TELESCOP= 'COBE    '           / COsmic Background Explorer satellite
INSTRUME= 'DIRBE   '           / COBE instrument [DIRBE, DMR, FIRAS]
OBJECT  = 'ALL-SKY '           / part of sky given [ALL-SKY, WEEKLY-SKY,
COMMENT                        /                    GAL-SLICE]
EQUINOX =               2000.0 / equinox of coords in following tables
REFERENC= 'DIRBE Explanatory Supplement' /
COMMENT     COBE specific keywords
DATE-BEG= '11/12/89'           / date of initial data represented (dd/mm/yy)
DATE-END= '21/09/90'           / date of final data represented   (dd/mm/yy)
PIXRESOL=                    9 / Quad tree pixel resolution [6, 9]
COMMENT
COMMENT     DIRBE specific keywords
PRODUCT = 'B09_AAM '           / Band 9 Annual Average Sky Map
VERSION = 'Pass 2B '           / Version of Data Reduction Software
WAVE9   = '140. microns'       / nominal wavelength of Band 9
SOLELONG= 'ALL     '           / all available solar elongations included
APPVEC  = 'COMBINED'           / approach vectors combined
ZLREMOV = 'NO      '           / signal from zodiacal dust remains in map
COMMENT
COMMENT
END
```

(8 card images of ASCII blanks)

```
XTENSION= 'BINTABLE'           / Extension type is Binary Table
BITPIX  =                    8 / Binary data
NAXIS   =                    2 / Data are in a table
NAXIS1  =                   27 / Number of 8 bit bytes in each row
```

```
NAXIS2  =               393216 / Number of rows
PCOUNT  =                    0 / Number of bytes of data following table
GCOUNT  =                    1 / Group count (always 1 for bintable extensions)
TFIELDS =                    7 / Number of fields (columns) in the table
COMMENT
TIMVERSN= 'OGIP/93-003'        / OGIP memo number where the convention
COMMENT                        / is described
COMMENT           The times reported in this file are atomic seconds elapsed
COMMENT           since 00:00:00 UTC, 1 January 1981.  Time information is
COMMENT           recorded in a manner consistent with the convention specified
COMMENT           in OGIP/93-003 with the understanding that time is counted
COMMENT           in atomic seconds and the origin of time (MJDREF) is quoted
COMMENT           in ephemeris MJD.
TIMESYS = '1981.00 '           / time system (same as IRAS)
MJDREFI =                44605 / Integer portion of ephemeris MJD
COMMENT                        / corresponding to 0h UTC 1 Jan 1981
MJDREFF =        0.00059240741 / Fractional portion of ephemeris MJD
COMMENT                        / corresponding to 0h UTC 1 Jan 1981
TIMEUNIT= 's       '           / unit for TSTART, TSTOP, TIMEZERO = seconds
TSTART  =        282185275.611 / observation start time in TIMESYS system
TSTOP   =        306753235.358 / observation stop time in TIMESYS system
COMMENT
COMMENT   -------------------------------------------------------------------
COMMENT   *******************************************************************
COMMENT         Annual Average Sky Map -- 140. microns
COMMENT         Mission-averaged photometric measurements and cumulative
COMMENT         sky coverage data for DIRBE Band 9 (140. microns).
COMMENT         The photometry is computed from the weighted average of all
COMMENT         available weekly-averaged maps for the entire mission,
COMMENT         up to the time when the cryogen supply was exhausted.
COMMENT         The foreground signal from Zodiacal dust remains in the map.
COMMENT   *******************************************************************
COMMENT   -------------------------------------------------------------------
COMMENT   ====================================================================
COMMENT   Coordinate Information (Spatial and Temporal):
COMMENT   ===================================================================
COMMENT   -------------------------------------------------------------------
COMMENT       DIRBE Pixel number (resolution 9)
TTYPE1  = 'Pixel_no'           / entire fieldname = Pixel_no
TUNIT1  = '        '           /
TFORM1  = '1J      '           /
COMMENT   ---------------------------------------------------------------------
COMMENT        Sub-pixel containing the mean DIRBE LOS (line-of-sight)
COMMENT         when pixel is divided into 256
COMMENT         sub-pixels (16x16 grid) according to
COMMENT         standard quad-sphere rules.
TTYPE2  = 'PSubPos '           / entire fieldname = Pixel_subpos
TUNIT2  = '        '           /
TFORM2  = '1B      '           /
```

```
COMMENT   --------------------------------------------------------------------
COMMENT       Time in TAI seconds since 01-JAN-1981:00:00:00.000 UTC.
COMMENT       This is the average time of all observations of a pixel
COMMENT       within the time interval TSTART to TSTOP,
COMMENT       regardless of the weight assigned to a particular observation.
COMMENT       For this product, not a physically meaningful quantity.
TTYPE3  = 'Time    '           / entire fieldname = Time
TUNIT3  = '        '           /
TFORM3  = '1D      '           /
COMMENT   -------------------------------------------------------------------
COMMENT   ===================================================================
COMMENT   DIRBE Sky Brightness Information:
COMMENT   ===================================================================
COMMENT   -------------------------------------------------------------------
COMMENT       Cumulative weighted average of the 140. micron photometry.
COMMENT        This average is calculated using the weighted number of
COMMENT        observations from each Weekly Averaged Map ( WtNumObs from
COMMENT        the Weekly Averaged Map) as the weight, such that
COMMENT        annual_average =
COMMENT           sum( weekly_average * weekly_weight )/ sum( weekly_weight)
COMMENT
COMMENT        The photometric measurement is given as a
COMMENT        spectral intensity, I(nu) in MJy/sr, and
COMMENT        is quoted at the nominal wavelength
COMMENT        for a source with nu*I(nu) = constant  -- i.e.,
COMMENT        a color correction is required if the source
COMMENT        spectrum differs from nu*I(nu) = constant.
COMMENT        Data which have been replaced by flagged values
COMMENT        in processing have values .LE. -16375.
COMMENT
TTYPE4  = 'Photomet'           / entire fieldname = Photometry
TUNIT4  = 'MJy/sr  '           /
TFORM4  = '1E      '           /
COMMENT   --------------------------------------------------------------------
COMMENT        The standard deviation of the (weighted) average photometry
COMMENT         values.
TTYPE5  = 'StdDev  '           / entire fieldname = Standard_deviation
TUNIT5  = 'MJy/sr  '           /
TFORM5  = '1E      '           /
COMMENT   --------------------------------------------------------------------
COMMENT        Sum of the weekly weighted number of observations that went into
COMMENT         forming the annual averages.
TTYPE6  = 'WtNumObs'           / entire fieldname = Weighted_num_obs
TUNIT6  = '        '           /
TFORM6  = '1I      '           /
COMMENT   ---------------------------------------------------------------------
COMMENT        Sum of the total number of observations available, regardless
COMMENT         of assigned weight.
TTYPE7  = 'SumNRecs'           / entire fieldname = Sum_numrecs
```

```
TUNIT7  = '        '           /
TFORM7  = '1J      '           /
COMMENT   ------------------------------------------------------------------
END
```

(20 card images of ASCII blanks)

Discussion of Example 6: DIRBE FITS Headers

These primary and binary table headers are for an all sky map (OBJECT keyword) from the (INSTRUME keyword) Diffuse Infrared Background Explorer (DIRBE) on the (TELESCOP keyword) Cosmic Background Explorer (COBE) from the Cosmology Data Analysis Center (ORIGIN keyword). The primary header provides information that applies to all HDUs in the file. There are no primary data (NAXIS keyword); the EXTEND keyword is set to true, advising the reader that extensions may be present, as indeed they are. Since the primary HDU has no data, the value of the BITPIX keyword does not matter, as long as it is one of the permitted values. This particular file was created on September 28, 1994 (DATE keyword), but the original map file was created July 10, 1994 (DATE-MAP keyword). The map coordinates are for equinox 2000 (EQUINOX keyword).

COBE specific keywords (COMMENT) show that the data cover a period from December 11, 1989 (DATE-BEG) through September 21, 1990 (DATE-END) and give the size of the region of space projected on to one pixel (PIXRESOL). DIRBE-specific keywords (COMMENT) identify the (PRODUCT), the (VERSION) of the data reduction software, the nominal (intended) wavelength (WAVE9), and provide information about choices made in processing the data (SOLELONG, APPVEC, and ZLREMOV). The blank cards used as separators have an explicit COMMENT, in contrast with the use of blank lines in Example 5.

The COMMENT following the OBJECT keyword illustrates one approach to the problem of a keyword/comment string that is longer than the 80-columns of a card image.

The binary table (XTENSION keyword) has 393216 rows (NAXIS2) each of which has seven fields (TFIELDS), which take up a total of 27 bytes per row (NAXIS1). There is no heap for variable length arrays (PCOUNT). BITPIX and GCOUNT have the required values as described in section 3.6.1. Like the primary HDU, the extension header uses a COMMENT in the keyword field for blank lines.

Before the table description are a number of keywords describing the time of the observation. The COMMENT lines outline the meaning of the keywords and their values and provide a reference (TIMVERSN) with more details. MJDREFI and MJDREFF give the ephemeris Modified Julian Date corresponding to the zero point of the time system, also specified by TIMESYS. TSTART and TSTOP give the time of the measurements in seconds (TIMEUNIT) after the zero point.

The keywords describing the table follow. Detailed explanatory comments are provided at the top and after the keywords for each field. All fields have a short title (`TTYPE`n) keyword of eight characters or fewer. The titles are mixed case, but FITS software should not attribute any significance to the case; the value is to make it easier for human readers to understand the file. The `COMMENT`s furnish the full field titles and provide detailed explanations. Other `COMMENT`s are used to format the table for easier examination by human readers. The creators of the file have chosen to include a `TUNIT`n keyword for every field, with a blank value if there are no units for the field. To do so is not required but this choice provides a design with a consistent set of keywords for every field. `TFORM`n keywords give the data type of each field. The header keywords describe the following fields:

1. `Pixel_no` is a 32-bit integer giving the pixel number on the map.
2. `PSubPos` is an 8-bit unsigned integer giving the pixel subposition, as explained in the comments.
3. `Time` is a double precision floating point number giving the average time for the observations of a pixel. The comments note that the value is not physically meaningful for this data product. Including this item allows the designers to produce a set of data products with a a common format, simplifying the reading process.
4. `Photomet` is short for "Photometry" and is a single precision floating point number giving a weighted average spectral intensity in MegaJanskys/steradian. The comments provide a precise definition and describe the weighting.
5. `StdDev` is a single precision floating point number giving the standard deviation of the mean in the previous field, in MegaJanskys/steradian.
6. `WtNumObs` is a 16-bit integer giving the sum of the weekly weighted number of observations used in deriving the annual averages.
7. `SumNRecs` is a 32-bit integer giving the total number of observations.

Each row has one unsigned integer, of one byte, one two-byte integer, two four byte-integers, two single precision floating point numbers (four bytes each) and one double precision floating point number (eight bytes), for a total of 27 bytes, equal to the value of the `NAXIS1` keyword.

A description of the DIRBE data products is available at
`http://ssdoo.gsfc.nasa.gov/astro/cobe/dirbe_products.html`.

Example 7: ASCA FITS Header

(The first two lines following are provided to help the reader determine column numbers. They are not part of FITS headers.)

```
0000000001111111111222222222233333333334444444444555555555566666666667777
1234567890123456789012345678901234567890123456789012345678901234567890909012

XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / 8-bit bytes
NAXIS   =                    2 / 2-dimensional binary table
NAXIS1  =                   30 / width of table in bytes
NAXIS2  =                22990 / number of rows in the table
PCOUNT  =                    0 / size of special data area
GCOUNT  =                    1 / one data group (required keyword)
TFIELDS =                   11 / number of fields in each row

TTYPE1  = 'TIME    '           / arrival time of the X-ray
TFORM1  = '1D      '           / data format of the field: 8-byte DOUBLE
TUNIT1  = 's       '           / physical unit of field: seconds

TTYPE2  = 'PI      '           / pulse-height invariant energy of the event
TFORM2  = '1J      '           / data format: 4-byte INTEGER
TUNIT2  = 'channel '           / physical unit of field
TLMIN2  =                    0 / Minimum legal value of PI
TLMAX2  =                 4095 / Maximum legal value of PI

TTYPE3  = 'X       '           / projected X (RA) sky pixel coordinate
TFORM3  = '1I      '           / data format: 2-byte INTEGER
TUNIT3  = 'pixel   '           / physical unit of field
TLMIN3  =                    1 / minimum legal value for column 3
TLMAX3  =                 1280 / maximum legal value for column 3

TTYPE4  = 'Y       '           / projected Y (Dec) sky pixel coordinate
TFORM4  = '1I      '           / data format: 2-byte INTEGER
TUNIT4  = 'pixel   '           / physical unit of field
TLMIN4  =                    1 / minimum legal value for column 4
TLMAX4  =                 1280 / maximum legal value for column 4

TTYPE5  = 'RAWX    '           / raw X detector coordinate of the event
TFORM5  = '1I      '           / data format: 2-byte INTEGER
TUNIT5  = 'pixel   '           / physical unit of field
TLMIN5  =                    6 / minimum legal value for column 5
TLMAX5  =                  426 / maximum legal value for column 5

TTYPE6  = 'RAWY    '           / raw Y detector coordinate of the event
```

```
TFORM6  = '1I      '           / data format: 2-byte INTEGER
TUNIT6  = 'pixel   '           / physical unit of field
TLMIN6  =                    1 / minimum legal value for column 6
TLMAX6  =                  422 / maximum legal value for column 6

TTYPE7  = 'PHA     '           / Pulse-Height-Analyzer energy
TFORM7  = '1I      '           / data format: 2-byte INTEGER
TUNIT7  = 'channel '           / physical unit of field
TLMIN7  =                    0 / minimum legal value for column 7
TLMAX7  =                 4095 / maximum legal value for column 7

TTYPE8  = 'DETX    '           / corrected X detector coordinate
TFORM8  = '1I      '           / data format: 2-byte INTEGER
TUNIT8  = 'pixel   '           / physical unit of field
TLMIN8  =                    1 / minimum legal value for column 8
TLMAX8  =                 1280 / maximum legal value for column 8

TTYPE9  = 'DETY    '           / corrected Y detector coordinate
TFORM9  = '1I      '           / data format: 2-byte INTEGER
TUNIT9  = 'pixel   '           / physical unit of field
TLMIN9  =                    1 / minimum legal value for column 9
TLMAX9  =                 1280 / maximum legal value for column 9

TTYPE10 = 'GRADE   '           / event quality grade
TFORM10 = '1I      '           / data format: 2-byte INTEGER
TUNIT10 = 'pixel   '           / physical unit of field
TLMIN10 =                    0 / minimum legal value for column 10
TLMAX10 =                    4 / maximum legal value for column 10

TTYPE11 = 'CCDID   '           / CCD number
TFORM11 = '1I      '           / data forma: 2-byte INTEGER
TLMIN11 =                    0 / minimum legal value for column 11
TLMAX11 =                    3 / maximum legal value for column 11

                 General Descriptive Keywords
EXTNAME = 'EVENTS  '           / Name of extension
HDUCLASS= 'OGIP    '           / format conforms to OGIP/GSFC conventions
HDUCLAS1= 'EVENTS  '           / Extension contains Events
INSTRUME= 'SIS1    '           / Instrument name
ORIGIN  = 'GSFC    '           / Origin of FITS file
OBJECT  = 'NGC 4151'           / Name of observed object
OBSERVER= 'I. N. Jones'        / Principal Investigator
TELESCOP= 'ASCA    '           / Telescope (mission) name
CREATOR = 'extractor10q'       / Program that created this FITS file
SEQNUM  =             71019020 / Sequential number from ODB
SEQPNUM =                  004 / Number of times sequence processed
PROCVER = 'A5.1.8  '           / Processing script version

                 Instrument mode keywords
```

```
MODAL_ID=                 1894 / Modal Configuration ID
BIT_RATE= 'HIGH    '           / Telemetry rate
OBS_MODE= 'POINTING'           / Observation mode
DATAMODE= 'BRIGHT2 '           / file generated by FAINT

                     Time Related Keywords
DATE    = '20/03/95'           / FITS file creation date (dd/mm/yy)
DATE-OBS= '05/12/93'           / date of observation start (dd/mm/yy)
TIME-OBS= '20:51:57'           / time of observation start (hh:mm:ss)
DATE-END= '06/12/93'           / date of observation end
TIME-END= '04:06:31'           / time of observation end
MJD-OBS =     4.9326869420E+04 / Modified Julian date of the data start time
MJDREF  =              48988.0 / MJD corresponding to SC clock start (1993.0)
TIMEREF = 'LOCAL   '           / Barycentric correction not applied to times
TIMESYS = '1993.0  '           / Time measured from 1993 Jan 1 00:00 UT
TIMEUNIT= 's       '           / unit for time related keywords: seconds
TSTART  =  2.92783178528423011303E+07 / time start
TSTOP   =  2.93043917384999990463E+07 / time stop
TELAPSE =  2.60738856576979160309E+04 / Elapsed time = TSTOP - TSTART
ONTIME  =  9.92793133954703807831E+03 / sum of the good time intervals
EXPOSURE=  9.92793133954703807831E+03 / Exposure
CLOCKAPP=                    F / Is mission time corrected for clock drift?
TASSIGN = 'SATELLITE'          / location of time assignment
TIMEDEL =     8.0000000000E+00 / time increment or time resolution of data
TIMEZERO=                  0.0 / offset to be applied to TIME column
ORBITBEG= '93120513'           / beginning orbit number
ORBITEND= '93120603'           / ending orbit number

              Position Related Keywords
RADECSYS= 'FK5     '           / World Coordinate System
EQUINOX =          2000.000000 / Equinox for coordinate system
RA_NOM  =               182.74 / Nominal RA
DEC_NOM =               39.429 / Nominal declination
RA_PNT  =           182.760681 / File average value of RA (degrees)
DEC_PNT =            39.434547 / File average value of DEC (degrees)
PA_PNT  =           237.685150 / File average value of ROLL (degrees)
RA_PNTE =             0.000556 / File standard deviation of RA (degrees)
DEC_PNTE=             0.000179 / File standard deviation of DEC (degrees)
PA_PNTE =             0.003630 / File standard deviation of ROLL (degrees)
TCRPX3  =       6.40500000E+02 / Sky X axis reference pixel
TCRPX4  =       6.40500000E+02 / Sky Y axis reference pixel
TCRVL3  =       1.82744003E+02 / Sky X coordinate at reference pixel (degrees)
TCRVL4  =       3.94291992E+01 / Sky Y coordinate at reference  pixel (degrees)
TCDLT3  =      -4.41999990E-04 / Sky X pixel scale (degrees/pixel)
TCDLT4  =       4.41999990E-04 / Sky Y pixel scale (degrees/pixel)
TCTYP3  = 'RA---TAN'           / Coordinate projection
TCTYP4  = 'DEC--TAN'           /  Coordinate projection
TCRPX8  =       6.40500000E+02 / Detector X ref pixel (center of address space)
TCRPX9  =       6.40500000E+02 / Detector Y ref pixel (center of address space)
```

```
TCRVL8  =       0.00000000E+00 / Detector X ref pixel value (pixels)
TCRVL9  =       0.00000000E+00 / Detector Y ref pixel value (pixels)
TCDLT8  =       2.70000007E-02 / Detector X pixel scale (mm/pixel)
TCDLT9  =       2.70000007E-02 / Detector Y pixel scale (mm/pixel)
OPTIC3  =       5.06579102E+02 / File mean optical axis X in sky coords (pixels)
OPTIC4  =       5.69040405E+02 / File mean optical axis Y in sky coords (pixels)
OPTIC8  =       6.18277771E+02 / Optical axis X in detector coords (pixels)
OPTIC9  =       7.73833313E+02 / Optical axis Y in detector coords (pixels)
FOV_X_MM=          23.00000000 / Detector X field of view (mm)
FOV_Y_MM=          23.00000000 / Detector Y field of view (mm)
END
```

(32 card images of blanks)

Discussion of Example 7: ASCA FITS Header

This FITS table contains an event list of the X-rays detected with the SIS1 detector on the ASCA X-ray satellite during a particular pointed observation. Each row contains data for a single photon, and the table contains information for all the events in this particular observation. Eleven different parameters are listed for each X-ray event. The header keywords supply additional information for deriving precise times or positions for each event.

This binary table (`XTENSION` keyword, `BITPIX` value of 8, `NAXIS` value of 2, `GCOUNT` value of 1) has 22990 rows (`NAXIS2` keyword) consisting of 30 bytes (`NAXIS1` keyword) distributed over 11 fields (`TFIELDS` keyword) and no variable length array heap (`PCOUNT`).

In this example, by contrast with examples 5 and 6, the keywords describing the structure of the table appear immediately after the required header keywords, and the more general information follows. The field titles (`TTYPE`*n*) are short, upper-case headings of five characters or fewer; the comment fields of the keyword card image explain the title. `TFORM`*n* keywords provide the format, and `TUNIT`*n* keywords provide the units, where appropriate. All units are lowercase; the case is not significant. The quantity in field 11 is a pure number, and the designers of this table have chosen not to have a `TUNIT11` field, a different approach from that of example 6. For many fields, `TLMAX`*n* and `TLMIN`*n* keywords are used, following the HEASARC convention discussed in section 5.6.1.4. The table contains the following fields:

1. `TIME` is a double precision floating point number containing the arrival time (in seconds) of the X-ray photon. The time system is described by the Time Related Keywords later in the file.

2. `PI` is a 4-byte integer containing the gain-corrected pulse-height energy channel number of the photon. The channel numbers run from 0 to 4095.

3. `X` is a 2-byte integer containing the projection of the event sky right ascension onto the X-axis of a rectangular pixel coordinate system; the axis runs from 1 to 1280 (`TLMAX3`, `TLMIN3`).

4. `Y` is a 2-byte integer containing the projection of the event sky declination onto the Y-axis of a rectangular pixel coordinate system; the axis runs from 1 to 1280.

5. RAWX is a 2-byte integer containing the CCD X-axis pixel location of the event with no corrections; the axis runs from 6 to 426.
6. RAWY is a 2-byte integer containing the CCD Y-axis pixel location of the event with no corrections; the axis runs from 1 to 422.
7. PHA is a 2-byte integer containing the pulse-height energy channel number of the data for this photon, with no gain correction. The channel numbers run from 0 to 4095.
8. DETX is a 2-byte integer containing the X-axis pixel location of the event in detector coordinates, corrected for instrument distortion; the axis runs from 1 to 1280.
9. DETY is a 2-byte integer containing the Y-axis pixel location of the event in detector coordinates, corrected for instrument distortion; the axis runs from 1 to 1280.
10. GRADE is a 2-byte integer describing the shape of the event, used as a quality indicator. Grade numbers run from 0 to 4.
11. CCDID is a 2-byte integer identifying the CCD number, which runs from 0 to 3.

Each row has nine two-byte integers, one four byte-integer, and one double precision floating point number (eight bytes), for a total of 30 bytes, equal to the value of the NAXIS1 keyword.

The remaining keywords are grouped. Each group is preceded by a centered descriptive heading; the first eight characters are blanks, and thus the headings are treated as comments.

Under "General Descriptive Keywords" are keywords that identify the extension and its content. The name, corresponding to a table title, is EVENTS (EXTNAME keyword). The HDUCLASS keyword identifies the authors of the file format and conventions as OGIP, and HDUCLAS1 the type of data as EVENTS. Examples of other types are SPECTRUM and GTI (Good Time Intervals). The usage of these two keywords follows OGIP/HEASARC convention; HDUCLASS identifies the source of the observation as one of NGC 4151 (OBJECT keyword) with the SIS1 instrument (INSTRUME) of the ASCA telescope (TELESCOP), with the principal investigator being I. N. Jones (OBSERVER). The ORIGIN, CREATOR, PROCVER, SEQNUM, and SEQPNUM keywords provide information on the origin of the file and its processing history. The "Instrument mode keywords" provide information on the instrument and data configuration.

The "Time Related Keywords", except for the standard reserved `DATE` keyword that identifies the file creation date as March 20, 1996, give the time of observation. The observation started at 57 seconds after 8:51 P. M. on December 5, 1995 (`DATE-OBS`, `TIME-OBS`) and ended at 31 seconds after 4:06 A. M. on December 6, 1993 (`DATE-END`, `TIME-END`). Of these keywords, `DATE-OBS` is a reserved FITS keyword; the others are user-defined keywords but with names chosen by analogy with the reserved keyword. In this case, the `DATE-OBS` keyword refers to the beginning of the observation, but, as noted in section 3.1.1.2, whether the `DATE-OBS` keyword refers to the beginning, middle, or end of the observation is not specified by the FITS rules and must be described in the file, as it is here. The start time is also specified in Modified Julian Date form using the widely used `MJD-OBS`, intended as a standard under the World Coordinates proposal. The `MJDREF`, `TIMEREF`, `TIMESYS`, and `TIMEUNIT` keywords define the time system and scale used for the `TIME` column of the main table and for expressing the start and stop time of the observations using the `TSTART` and `TSTOP` keywords. The `ORBITBEG` and `ORBITEND` keywords identify the orbit numbers of the start and stop of the observation. The other keywords provide information on the elapsed times of observation and on defining the time scales.

The "Position Related Keywords" describe coordinate systems. The `RADECSYS` keyword is the World Coordinates convention keyword defined in section 4.3 and identifies the coordinate system used as the FK5, the system in use since the 1976 IAU; `EQUINOX` identifies the coordinate system as precessed to the year 2000.000000. The next eight keywords describe the direction in which the instrument points. Right ascension and declination are always given in degrees. The `RA_PNT` and `DEC_PNT` keywords are specifically listed among the standard HFWG keywords. The next 14 keywords describe the coordinate systems in which the event positions of columns 3-4 and 8-9 are defined, one of the cases defined in section 5.6.2. Keywords similar to the standard coordinate keywords of section 3.1.1.2 but with one intermediate letter omitted and prefaced by `T` are used. Columns 3 and 4 describe a position in a tangent projection (section 4.2.2.4) of the sky onto a FITS array. For the X-axis, the right ascension at array position 640.5, that is, equivalent to halfway between 640 and 641 (`TCRPX3` keyword), is 182.744003 degrees (`TCRVL3`), and the right ascension changes by 4.4199999×10^{-4} degrees for a change of 1 in the array position, decreasing with increasing index number (`TCDELT3`). For the Y-axis, the declination at array position 640.5 (`TCRPX4` keyword) is +39.4291992 degrees (`TCRVL4`), and changes by 4.4199999×10^{-4}degrees for a change of 1 in the array index, increasing with increasing index number (TCDELT4). Columns 8 and 9 provide positions of the events in a linearized focal plane pixel coordinate system obtained by correcting the raw detector coordinates for instrument distortion. FITS array member (640.5, 640.5) corresponds to (0,0) of the pixel coordinate system (TCRPX8, TCRVL8, TCRPX9, TCRVL9), with the scale on both axes 0.027 mm/unit change in index. The OPTIC*n* keywords provide the array position of the optical axis in both linearized detector coordinates and sky coordinates, and the FOV_X_MM and FOV_Y_MM keywords provide the size of the field of view at the detector, 23 mm $\times$ 23 mm. The FITS conventions and a discussion of the ASCA system are available on separate pages at the HEASARC sites at http://heasarc.gsfc.nasa.gov/ and ftp://legacy.gsfc.nasa.gov/.

Appendix B

IEEE Formats

(The material in this appendix is adapted from the IEEE-754 (1985) floating point standard. It is not intended to be a comprehensive description of the IEEE formats; readers should refer to the IEEE standard.)

FITS recognizes all IEEE basic formats, including the special values. Table B.1 shows the type of IEEE floating point value, regular or special, for all double and single precision hexadecimal byte patterns.

IEEE value	Double Precision	Single Precision
$+0$	0000000000000000	00000000
denormalized	0000000000000001	00000001
	to	to
	000FFFFFFFFFFFFF	007FFFFF
positive underflow	0010000000000000	00800000
positive numbers	0010000000000001	00800001
	to	to
	7FEFFFFFFFFFFFFE	7F7FFFFE
positive overflow	7FEFFFFFFFFFFFFF	7F7FFFFF
$+\infty$	7FF0000000000000	7F800000
NaN[1]	7FF0000000000001	7F800001
	to	to
	7FFFFFFFFFFFFFFF	7FFFFFFF
-0	8000000000000000	80000000
negative	8000000000000001	80000001
denormalized	to	to
	800FFFFFFFFFFFFF	807FFFFF
negative underflow	8010000000000000	80800000
negative numbers	8010000000000001	80800001
	to	to
	FFEFFFFFFFFFFFFE	FF7FFFFE
negative overflow	FFEFFFFFFFFFFFFF	FF7FFFFF
$-\infty$	FFF0000000000000	FF800000
NaN[1]	FFF0000000000001	FF800001
	to	to
	FFFFFFFFFFFFFFFF	FFFFFFFF

[1] Certain values may be designated as *quiet* NaN (no diagnostic when used) or *signaling* (produces diagnostic when used) by particular implementations.

Table B.1: IEEE Floating Point Formats

References

ANSI 1977, "American National Standard for Information Processing: Code for Information Interchange," ANSI X3.4-1977 (ISO 646) (New York: American National Standards Institute, Inc.).

Cotton, W.D., Tody, D. B., and Pence, W. D. 1995, "Binary Table Extension to *FITS*," *Astron. Astrophys. Suppl.*, **113**, 159-166.

Greisen, E. W. 1983, "Non-linear Coordinate Systems in AIPS," *AIPS Memo* **27**.

Greisen, E. W. 1986, "Additional Non-linear Coordinates," *AIPS Memo* **46**.

Greisen, E. W., and Calabretta, M. 1996, "Representation of Celestial Coordinates in FITS" preprint, available in PostScript form at `ftp://fits.cv.nrao.edu/fits/documents/wcs/wcs_all.ps`.

Greisen, E. W., and Harten, R. H. 1981, "An Extension of FITS for Small Arrays of Data," *Astron. Astrophys. Suppl.*, **44**, 371-374.

Grosbøl, P., Harten, R. H., Greisen, E. W., and Wells, D. C. 1988, "Generalized Extensions and Blocking Factors for FITS," *Astron. Astrophys. Suppl.*, **73**, 359-364.

Hanisch, R. J., and Wells, D. C. 1988, "World Coordinate Systems Representations Within the FITS Format" (DRAFT preprint).

Harten, R. H., Grosbøl, P., Greisen, E. W., and Wells, D. C. 1988, "The FITS Tables Extension," *Astron. Astrophys. Suppl.* **73**, 365-372.

IAU Information Bulletin **49**, 14, 1983.

IAU 1988, *Transactions of the IAU Twentieth General Assembly*," ed. D. McNally (Dordrecht:Kluwer).

IEEE 1985, "American National Standard - IEEE Standard for Binary Floating Point Arithmetic," ANSI-IEEE 754-1985 (New York: American National Standards Institute, Inc.).

Jennings, D. G., Pence, W. D., Folk, M., and Schlesinger, B. M. 1996, "A Hierarchical Grouping Convention for FITS, " preprint, available electronically from `http://adfwww.gsfc.nasa.gov/other/convert/group.html`.

Muñoz, J. R. 1989, "IUE data in *FITS* Format," ESA IUE Newsletter, **32**, 12-45.

National Radio Astronomy Observatory 1990, "Going AIPS," (Charlottesville, VA: National Radio Astronomy Observatory).

NOST 1995, "Definition of the Flexible Image Transport System (FITS)", NOST 100-1.1 (Greenbelt: NASA/Goddard Space Flight Center).

Ponz, J. D., Thompson, R. W., and Muñoz, J. R. 1994, "The *FITS* Image Extension," *Astron. Astrophys. Suppl.*, **105**, 53-55.

Seaman, R. and Pence, W. D. 1995, "FITS Checksum Proposal," available in PostScript form at `ftp://iraf.noao.edu/misc/checksum/checksum.ps`.

Warnock, A., Hill, R. S., Pfarr, B. B., and Wells, D. C. 1990, "An Extension of FITS for Data Compression", draft preprint, available electronically at `http://fits.cv.nrao.edu:80/documents/drafts/`.

Wells, D. C., and Greisen, E. W. 1979, "A Flexible Image Transport System," *Proc. 5th Colloquium Astrophysics*, eds. G. Sedmak, M. Capaccioli, and R. J. Allan, p. 445.

Wells, D. C., Greisen, E. W., and Harten, R. H. 1981, "FITS: A Flexible Image Transport System," *Astron. Astrophys. Suppl.*, **44**, 363-370.

Wells, D. C., and Grosbøl, P. 1990, "Floating Point Agreement for FITS," preprint, available from FITS Support Office.

Zarate, N., and Greenfield, P. 1996, "A FITS Image Extension Kernel for

IRAF," *Astronomical Data Analysis Software and Systems V*, eds. G. H. Jacoby and J. Barnes, pp. 331-334.