A&A 376, 359–380 (2001)
DOI: 10.1051/0004-6361:20010923

Astronomy & Astrophysics

Definition of the Flexible Image Transport System (FITS)⋆

R. J. Hanisch[1], A. Farris[1], E. W. Greisen[2], W. D. Pence[3], B. M. Schlesinger[4], P. J. Teuben[5], R. W. Thompson[6], and A. Warnock III[7]

[1] Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218, USA
[2] National Radio Astronomy Observatory
[3] NASA Goddard Space Flight Center
[4] Raytheon ITSS
[5] University of Maryland
[6] Computer Sciences Corporation
[7] A/WWW Enterprises

Received 29 May 2001 / Accepted 21 June 2001

Abstract. The Flexible Image Transport System – FITS – has been in use in the astronomical community for over two decades. A newly updated version of the standard has recently been approved by the International Astronomical Union FITS Working Group. This new version of the standard appears here in its entirety. As a preface we briefly describe the process by which the standard evolves and revisions are approved, and note two minor changes to NOST 100–2.0 which were adopted by the IAU FWG.

Key words. instrumentation: miscellaneous – methods: miscellaneous – techniques: miscellaneous – astronomical databases: miscellaneous

Introduction

The Flexible Image Transport System – FITS – was originally developed in the late 1970s to enable the exchange of astronomical image data between computers of different type, with different word lengths and different means of expressing numerical values. Although the IEEE numerical formats have been widely adopted by the computer industry during the past twenty years, and in 1989 the FITS Standard was revised to utilize them, to this day computer manufacturers have yet to agree upon a single standard for bit order. In addition, independent of the numerical values themselves, a standard is essential for expressing the relationship of the data to the instrument with which they were obtained, to the position on the sky or association with wavelength, or with other general descriptive information that collectively constitutes the *metadata* for the observation. FITS has evolved over the years, encompassing new and more complex data structures in accord with the increasing sophistication of new astronomical instruments, and providing support for much more than the "images" implied by the name. Images, spectra, data cubes, text tables, and binary tables are all supported, and with a variety of conventions in nomenclature and structure these basic elements have been combined to accommodate data spanning the range from digital (and digitized) images to output from computational simulations. Moreover, FITS has been immensely successful as a community-wide data format standard. No other scientific community has had anything like the success the astronomy community has had with FITS, and we are envied by many other communities for this cohesiveness. We have a process for amending and adding to the standard that assures broad community participation, and although this sometimes makes the process of change rather slow it helps to assure community support and compliance. All major astronomical software packages read and write FITS format data, and many have adopted FITS not only for exchange with other programs and facilities, but as a native run-time data format. The inherent inefficiencies of FITS (such as sequential header records, which when the allocated space is filled and a new header record is desired to be written, requires all following data to be rewritten) have been offset by the tremendous improvements in CPU and I/O efficiencies of modern desktop computers.

Send offprint requests to: R. J. Hanisch,
e-mail: hanisch@stsci.edu

⋆ Appendices are only available in electronic form at http://www.edpsciences.org

Development of the NOST Standard

In 1987 NASA developed plans for the Astrophysics Data System, an integrated approach to the management of data from all astrophysics missions. Although much of the original plan for the ADS failed to be realized (the current ADS abstract and bibliographic services being a notable exception), the policy decision was reached that all NASA astrophysics mission data sets should be made available to the community in the FITS format. In 1989 the NASA/Science Office of Standards and Technology – NOST – established the FITS Support Office to assist mission data managers in formatting their data in FITS. (Responsibility for the NASA FITS Support Office transitioned to the Astrophysics Data Facility group at GSFC in 1995.) NOST also commissioned the first of the FITS Technical Panels whose task was to recast the FITS papers (published in *Astronomy and Astrophysics Supplements*) into a form acceptable as an official NASA standard. The first draft standard, NOST 100–0.1, was released in December 1990. Since then there have been six revisions of the NOST standard, clarifying ambiguities and adding new features with each version. The accompanying paper represents NOST 100–2.0 and contains all FITS revisions and extensions that have been approved by the IAU-FWG through the end of 1998.

The NOST Technical Panel was responsible for developing the standards documents, making these available for public review and comment, and then evaluating each comment received and making additional revisions to the standard as necessary to address the comment. All comments and the NOST Technical Panel reactions to them were posted on the Internet and distributed by e-mail to all who submitted comments. This process of open review and discussion was audited by the NOST Accreditation Panel to assure that community input was open and unrestricted, and that the Technical Panel was fully accountable to the community. Once approved by the Accreditation Panel, the NOST FITS document becomes the official NASA standard.

Adoption of the standard by the community

FITS is used world-wide in astronomy, and thus a NASA FITS standard is not the final word. Recognizing the value-added of the NOST FITS Technical Panel's work, however, the community has taken the NASA standard as both a practical working document and has officially endorsed it through regional and international organizations. There are three regional FITS committees: the North American FITS Committee, which is convened under the auspices of the Working Group on Astronomical Software of the American Astronomical Society, the European FITS Committee, and the Japanese FITS Committee. Changes to the FITS standard are voted on in these committees and then forwarded for review to the FITS Working Group of the International Astronomical Union. The IAU FWG is the final voice of approval for revisions to the standard.

The NOST Technical Panel worked hard to resolve all discrepancies between the various FITS papers and to clarify all potentially ambiguous text. Nevertheless, some areas of the document may still be unclear to some readers or may be subject to misinterpretation. It is left to future Technical Panels to continue the effort to refine and clarify the document. These Technical Panels will also need to incorporate the results of future FITS negotiations into the document, such as the anticipated World Coordinate Systems (WCS) agreements (available at `http://www.cv.nrao.edu/fits/documents/wcs/`).

The NOST 100-2.0 document was approved by all three of the regional FITS Committees, without dissent, during 1999 and, in a vote taken on 12 October 2000, the IAU FITS Working Group adopted the following resolution, without dissent:

> The IAU FITS Working [IAU-FWG] Group adopts the "Definition of the Flexible Image Transport System (FITS)" [NOST 100–2.0] as the official version of the FITS standards, superseding Astron. Astrophys. Suppl. 44, 363–370 (1981) and the other FITS papers listed in Sect. 2 of NOST 100-2.0, with these interpretations/modifications of its text:
>
> 1. Use of the word *"deprecated"* in the first paragraph of Sect. 7 "Random Groups Structure" is understood to mean that binary table extensions should be used in new astronomical application areas instead of the random groups format where either is appropriate and where there is no historical precedent for random groups. Existing applications of the random groups structure (almost exclusively interferometry) may continue to use random groups as needed indefinitely;
> 2. It is noted that the following sentence in B.2, *"The size implied by the TDIMn keyword will equal the element count specified in the TFORMn keyword."* is not valid in the case of variable length array columns. This sentence should be replaced with wording similar to the following: *"The total number of elements in the array equals the product of the dimensions specified in the TDIMn keyword. This size must be equal to the repeat count on the TFORMn keyword, or, in the case of columns which have a "P" TFORMn datatype, equal to the array length specified in the variable length array descriptor (see Appendix B.1). In the special case where the variable length array descriptor has a size of zero, then the TDIMn keyword is not applicable."*

Acknowledgements. The authors acknowledge the support of NOST, in particular, Don Sawyer, for spearheading the standardization effort. The National Space Science Data Center

and Astrophysics Data Facility at NASA Goddard Space Flight Center have sponsored the FITS Support Office, staffed for many years by Barry Schlesinger and overseen by Richard A. White. The community owes immeasurable thanks to Don Wells, National Radio Astronomy Observatory, for his tireless efforts in building consensus in the FITS community and his leadership of the IAU FITS Working Group. Preben Grosbøl, European Southern Observatory, preceded Don in this role and has also provided leadership within the European community. Bill Pence (HEASARC, NASA Goddard Space Flight Center) has written the most complete FITS I/O software package and this is widely used in the community. Peter Bunclark (IoA, Cambridge) and Arnold Rots (SAO) led the development of the `DATExxxx` keywords for year 2000 compatibility. The authors of the original FITS papers, Don Wells, Eric Greisen, Ron Harten, Preben Grosbøl, Daniel Ponz, Randy Thompson, José Muñoz, Bill Cotton, Doug Tody, and Bill Pence, deserve great credit for their ground-breaking work and ingenuity in adapting FITS to accommodate new data structures.

The members of the regional FITS committees (as of October 1999) and the IAU FITS Working Group are listed below.

North American FITS Committee

Peter Teuben, Chair	U. Maryland
Steve Allen	Lick Observatory
Daniel Durand	HIA/CADC
Allen Farris	STScI
Eric Greisen	NRAO
Arne Henden	USNO
Robert Kibrick	Lick Observatory
William Lupton	Keck Observatory
Eric Mandel	CfA
Robert Narron	IPAC
William Pence	NASA GSFC
Jeffrey Percival	U. Wisconsin
Arnold Rots	CfA
Skip Schaller	Steward Observatory
Barry Schlesinger	Raytheon ITSS
Randall Thompson	Computer Sciences Corp.
Doug Tody	NOAO
Stephen Walton	Cal. State U. Northridge
Archibald Warnock	A/WWW Enterprises
Don Wells	NRAO
Robert Hanisch	STScI (ex officio)

European FITS Committee

Preben Grosbøl, Chair	ESO
Peter Bunclark	IoA, Cambridge
Anatoly Piskunov	IoA, Russian Acad. Sci.
Ernst Raimond	NFRA
Patrick Wallace	RAL

Japanese FITS Committee

Shiro Nishimura, Chair	NAOJ
Osamu Kanamitsu	Fukuoka U.
Yasuhiro Murata	ISAS
Eiji Nishihara	NAOJ
Toshiyuki Sasaki	NAOJ
Shigeomi Yoshida	U. Tokyo

IAU FITS Working Group

Don Wells, Chair	USA
Bill Cotton	USA
John Glaspey	USA
Eric Greisen	USA
Preben Grosbøl	Germany (ESO)
Robert Hanisch	USA
Don Jennings	USA
Osamu Kanamitsu	Japan
Francois Ochsenbein	France
William Pence	USA
Bruce Peterson	Australia
Anatoly Piskunov	Russia
Ernst Raimond	The Netherlands
Peter Teuben	USA
Doug Tody	USA
Pat Wallace	UK
Wayne Warren	USA

Definition of the Flexible Image Transport System (*FITS*)

March 29, 1999

Standard

NOST 100–2.0

NASA/Science Office of Standards and Technology
Code 633.2
NASA Goddard Space Flight Center
Greenbelt, MD 20771
USA

Foreword to NOST 100–2.0

The NASA/Science Office of Standards and Technology (NOST) of the National Space Science Data Center (NSSDC) of the National Aeronautics and Space Administration (NASA) has been established to serve the space science communities in evolving cost effective, interoperable data systems. The NOST performs a number of functions designed to facilitate the recognition, development, adoption, and use of standards by the space science communities.

Approval of a NOST standard requires verification by the NOST that the following requirements have been met: consensus of the Technical Panel, proper adjudication of the comments received from the targeted space and Earth science community, and conformance to the accreditation process.

A NOST standard represents the consensus of the Technical Panel convened by the NOST. Consensus is established when the NOST Accreditation Panel determines that substantial agreement has been reached by the Technical Panel. However, consensus does not necessarily

imply that all members were in full agreement with every item in the standard. NOST standards are not binding as published; however, they may serve as a basis for mandatory standards when adopted by NASA or other organizations.

A NOST standard may be revised at any time, depending on developments in the areas covered by the standard. Also, within five years from the date of its issuance, this standard will be reviewed by the NOST to determine whether it should 1) remain in effect without change, 2) be changed to reflect the impact of new technologies or new requirements, or 3) be retired or canceled.

The Technical Panel that developed this version of the standard consisted of the following members:

Robert J. Hanisch, Chair, Space Telescope Science Institute,
William D. Pence, Secretary, NASA GSFC,
Barry M. Schlesinger, Past Secretary, Raytheon STX,
Allen Farris, Space Telescope Science Institute,
Eric W. Greisen, National Radio Astronomy Observatory,
Peter J. Teuben, University of Maryland,
Randall W. Thompson, Computer Sciences Corporation,
Archibald Warnock, A/WWW Enterprises.

Members of the previous Technical Panels also included:
Lee E. Brotzman, Hughes STX,
Edward Kemper, Hughes STX,
Michael E. Van Steenberg, NASA GSFC,
Wayne H. Warren Jr., Hughes STX,
Richard A. White, NASA Goddard Space Flight Center.

Introduction to NOST 100–2.0

The Flexible Image Transport System (*FITS*) evolved out of the recognition that a standard format was needed for transferring astronomical data from one installation to another. The original form, or Basic *FITS* (Wells et al. 1981), was designed for the transfer of images and consisted of a binary array, usually multidimensional, preceded by an ASCII text header with information describing the organization and contents of the array. The *FITS* concept was later expanded to accommodate more complex data formats. A new format for image transfer, *random groups*, was defined (Greisen & Harten 1981) in which the data would consist of a series of arrays, with each array accompanied by a set of associated parameters. These formats were formally endorsed by the International Astronomical Union (IAU) in 1982 (IAU Information Bulletin No. 49, 1983). Provisions for data structures other than simple arrays or groups were made later. These structures appear in *extensions*, each consisting of an ASCII header followed by the data whose organization it describes. A set of general rules governing such extensions (Grosbøl et al. 1988) and a particular extension, ASCII table (Harten et al. 1988), were endorsed by the IAU General Assembly in 1988 (IAU Information Bulletin No. 61, 1988). At the same General Assembly, an IAU *FITS* Working Group (IAUFWG) was formed (McNally 1988) under IAU Commission 5 (Astronomical Data) with the mandate to maintain the existing *FITS* standards and to review, approve, and maintain future extensions to *FITS*, recommended practices for *FITS*, implementations, and the thesaurus of approved *FITS* keywords. In 1989, the IAUFWG approved a formal agreement (Wells & Grosbøl 1990) for the representation of floating point numbers. In 1994, the IAUFWG endorsed two additional extensions, the image extension (Ponz et al. 1994) and the binary table extension (Cotton et al. 1995). *FITS* was originally designed and defined for 9-track half-inch magnetic tape. However, as improvements in technology have brought forward other data storage and data distribution media, it has generally been agreed that the *FITS* format is to be understood as a logical format and not defined in terms of the physical characteristics of any particular data storage medium. In 1994, the IAUFWG adopted a set of rules (Grosbøl & Wells 1994) governing the relation between the *FITS* logical record size and the physical block size for sequential media and bitstream devices. The IAUFWG also approved in 1997 an agreement (Bunclark & Rots 1997) defining a new format for encoding the date and time in the `DATE`, `DATE-OBS`, and other related `DATExxxx` keywords to correct the ambiguity in the original DATE keyword format beginning in the year 2000.

1. Overview

> *An* archival format *must be utterly portable and self-describing, on the assumption that, apart from the transcription device, neither the software nor the hardware that wrote the data will be available when the data are read.* "Preserving Scientific Data on our Physical Universe", p. 60. Steering Committee for the Study on the Long-Term Retention of Selected Scientific and Technical Records of the Federal Government, [US] National Research Council, National Academy Press 1995.

1.1. Purpose

This standard formally defines the *FITS* format for data structuring and exchange that is to be used where applicable as defined in Sect. 1.3. It is intended as a formal codification of the *FITS* format that has been endorsed by the IAU for transfer of astronomical data, fully consistent with all actions and endorsements of the IAU and the IAU *FITS* Working Group (IAUFWG). Minor ambiguities and inconsistencies in *FITS* as described in the original papers are eliminated.

1.2. Scope

This standard specifies the organization and content of *FITS* data sets, including the header and data, for all standard *FITS* formats: Basic *FITS*, the random groups structure, the ASCII table extension, the image extension, and the binary table extension. It also specifies

minimum structural requirements for new extensions and general principles governing the creation of new extensions. It specifies the relation between physical block sizes and logical records for *FITS* files on bitstream devices and sequential media. For headers, it specifies the proper syntax for card images and defines required and reserved keywords. For data, it specifies character and value representations and the ordering of contents within the byte stream. It defines the general rules to which new extensions are required to conform.

1.3. Applicability

This standard describes an extensible data interchange format particularly well suited for transport and archiving of arrays and tables of astronomical data. The IAU has recommended that all astronomical computer facilities support *FITS* for the interchange of binary data. It has been NASA policy for its astrophysics projects to make their data available in *FITS* format. This standard may also be used to define the format for data transport in other disciplines, as may be determined by the appropriate authorities.

1.4. Organization of this document

Section 3 is a glossary of definitions, acronyms, and symbols. In Sect. 4, this document describes the overall organization of a *FITS* file, the contents of the first (primary) header and data, the rules for creating new *FITS* extensions, and the relation between physical block sizes and logical records for *FITS* files on bitstream devices and sequential media. The next two sections provide additional details on the header and data, with a particular focus on the primary header. Section 5 provides details about header card image syntax and specifies those keywords required and reserved in a primary header. Section 6 describes how different data types are represented in *FITS*. The following sections describe the headers and data of two standard *FITS* structures, the now deprecated random groups records (Sect. 7) and the current standard extensions: ASCII table, image, and binary table (Sect. 8). Throughout the document, deprecation of structures or syntax is noted where relevant. Files containing deprecated features are valid *FITS*, but these features should not be used in new files; the old files using them remain standard because of the principle that no change in *FITS* shall cause a valid *FITS* file to become invalid.

The Appendixes contain material that is not part of the standard[1]. The first, Appendix A, provides a formal expression of the keyword/value syntax for header card images described in Sect. 5.2. Appendix B provides examples of widely accepted *FITS* conventions that are not part of the formal *FITS* standard. It describes three conventions in use with the binary table extension – one for handling multidimensional arrays, one for including variable length arrays, and one for arrays of substrings. Appendix C describes aspects of the implementation of *FITS* on physical media not covered by the blocking agreement. Appendix D is the appendix to the agreement endorsed by the IAUFWG for a new format for keywords expressing dates. The new format uses a four-digit value for the year, and thus eliminates any ambiguity in dates from the year 2000 and after. This appendix is not part of the formal agreement. It contains a detailed discussion of time systems. It has been slightly reformatted for stylistic compatibility with the remainder of this document. Appendix E lists the differences between this standard and the specifications of prior publications; it also identifies those ambiguities in the documents endorsed by the IAU on which this standard provides specific rules. The next four appendixes provide reference information: a tabular summary of the *FITS* keywords (Appendix F), a list of the ASCII character set and a subset designated *ASCII text* (Appendix G), a description of the IEEE floating point format (Appendix H), and a list of the extension type names that have been reserved as of the date this document was issued (Appendix I). Appendix J is a list of NOST documents, including earlier versions of this standard.

[1] The Appendixes appear only in the electronic version of this article.

2. References

[The references for this document have been moved to the end of the article, in conformance with the editorial policies of the journal. This section is retained in order to maintain consistency in section numbering with the NASA Standard.]

3. Definitions, acronyms, and symbols

␣ Used to designate an ASCII blank.

ANSI American National Standards Institute.

Array A sequence of data values. This sequence corresponds to the elements in a rectilinear, n-dimension matrix ($0 \leq n \leq 999$).

Array value The value of an element of an array in a *FITS* file, without the application of the associated linear transformation to derive the physical value.

ASCII American National Standard Code for Information Interchange.

ASCII blank The ASCII character for blank which is represented by hexadecimal 20 (decimal 32).

ASCII character Any member of the 7-bit ASCII character set.

ASCII NULL Hexadecimal 00.

ASCII text ASCII characters hexadecimal 20–7E.

Basic *FITS* The *FITS* structure consisting of the primary header followed by a single primary data array.

Bit A single binary digit.

Byte An ordered sequence of eight consecutive bits treated as a single entity.

Card image A sequence of 80 bytes containing ASCII text, treated as a logical entity.

Conforming extension An extension whose keywords and organization adhere to the requirements for conforming extensions defined in Sect. 4.4.1 of this standard.

DAT 4 mm Digital Audio Tape.

Deprecated This term is used to refer to obsolete structures that should not be used for new applications but remain valid.

Entry A single value in a table.

Extension A *FITS* HDU appearing after the primary HDU in a *FITS* file.

Extension name The identifier used to distinguish a particular extension HDU from others of the same type, appearing as the value of the `EXTNAME` keyword.

Extension type An extension format.

Field A set of zero or more table entries collectively described by a single format.

File A sequence of one or more records terminated by an end-of-file indicator appropriate to the medium.

FITS Flexible Image Transport System.

FITS file A file with a format that conforms to the specifications in this document.

FITS structure One of the components of a *FITS* file: the primary HDU, the random groups records, an extension, or, collectively, the special records following the last extension.

Floating point A computer representation of a real number.

Fraction The field of the mantissa (or significand) of a floating point number that lies to the right of its implied binary point.

Group parameter value The value of one of the parameters preceding a group in the random groups structure, without the application of the associated linear transformation.

GSFC Goddard Space Flight Center.

HDU Header and Data Unit. A data structure consisting of a header and the data the header describes. Note that an HDU may consist entirely of a header with no data records.

Header A series of card images organized within one or more *FITS* logical records that describes structures and/or data which follow it in the *FITS* file.

Heap A supplemental data area, currently defined to follow the table in a binary table extension.

IAU International Astronomical Union.

IAUFWG International Astronomical Union *FITS* Working Group.

IUE International Ultraviolet Explorer.

IEEE Institute of Electrical and Electronic Engineers.

IEEE NaN IEEE Not-a-Number value.

IEEE special values Floating point number byte patterns that have a special, reserved meaning, such as -0, $\pm\infty$, $\pm$underflow, $\pm$overflow, $\pm$denormalized, $\pm$NaN (see Appendix H).

Indexed keyword A keyword that is of the form of a fixed root with an appended positive integer count.

Keyword The first eight bytes of a header card image.

Logical record A record comprising 2880 8-bit bytes.

Mandatory keyword A keyword that must be used in all *FITS* files or a keyword required in conjunction with particular *FITS* structures.

Mantissa Also known as significand. The component of an IEEE floating point number consisting of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.

Matrix A data array of two or more dimensions.

NOST NASA/Science Office of Standards and Technology.

Physical value The value in physical units represented by an element of an array and possibly derived from the array value using the associated, but optional, linear transformation.

Picture element A single location within an array.

Pixel Picture element.

Primary data array The data array contained in the primary HDU.

Primary HDU The first HDU in a *FITS* file.

Primary header The first header in a *FITS* file, containing information on the overall contents of the file as well as on the primary data array.

Record A sequence of bits treated as a single logical entity.

Reference point The point along a given coordinate axis, given in units of pixel number, at which a value and increment are defined.

Repeat count The number of values represented in a binary table field.

Reserved keyword An optional keyword that may be used only in the manner defined in this standard.

Special records A series of 23040-bit (2880 8-bit byte) records, following the primary HDU, whose internal structure does not otherwise conform to that for the primary HDU or to that specified for a conforming extension in this standard.

Standard extension A conforming extension whose header and data content are specified explicitly in this standard.

Type name The value of the `XTENSION` keyword, used to identify the type of the extension in the data following.

Valid value A member of a data array or table corresponding to an actual physical quantity.

4. FITS file organization

4.1. Overall

A *FITS* file shall be composed of the following *FITS* structures, in the order listed:

- Primary HDU;
- Conforming Extensions (optional);
- Other special records (optional).

Each *FITS* structure shall consist of an integral number of *FITS* logical records. The primary HDU shall start with the first record of the *FITS* file. The first record of each subsequent *FITS* structure shall be the record immediately following the last record of the preceding *FITS* structure. The size of a *FITS* logical record shall be 23 040 bits, corresponding to 2880 8-bit bytes.

*4.2. Individual **FITS** structures*

The primary HDU and every extension HDU shall consist of an integral number of header records consisting of ASCII text, which may be followed by an integral number of data records. The first record of data shall be the record immediately following the last record of the header.

4.3. Primary header and data array

The first component of a *FITS* file shall be the primary header. The primary header may, but need not be, followed by a primary data array. The presence or absence of a primary data array shall be indicated by the values of the `NAXIS` or `NAXISn` keywords in the primary header (Sect. 5.4.1.1).

4.3.1. Primary header

The header of a primary HDU shall consist of a series of card images in ASCII text. All header records shall consist of 36 card images. Card images without information shall be filled with ASCII blanks (hexadecimal 20).

4.3.2. Primary data array

In *FITS* format, the primary data array shall consist of a single data array of 0–999 dimensions. The random groups convention in the primary data array is a more complicated structure (see Sect. 7). The data values shall be a byte stream with no embedded fill or blank space. The first value shall be in the first position of the first primary data array record. The first value of each subsequent row of the array shall be in the position immediately following the last value of the previous row. Arrays of more than one dimension shall consist of a sequence such that the index along axis 1 varies most rapidly, that along axis 2 next most rapidly, and those along subsequent axes progressively less rapidly, with that along axis `m`, where `m` is the value of `NAXIS`, varying least rapidly; i.e., the elements of an array $A(x_1, x_2, \ldots, x_m)$ shall be in the order shown in Fig. 1. The index count along each axis shall begin with 1 and increment by 1 up to the value of the `NAXISn` keyword (Sect. 5.4.1.1).

If the data array does not fill the final record, the remainder of the record shall be filled by setting all bits to zero.

A(1, 1, ..., 1),
A(2, 1, ..., 1),
⋮,
A(NAXIS1, 1, ..., 1),
A(1, 2, ..., 1),
A(2, 2, ..., 1),
⋮,
A(NAXIS1, 2, ..., 1),
⋮,
A(1, NAXIS2, ..., NAXISm),
⋮,
A(NAXIS1, NAXIS2, ..., NAXISm)

Fig. 1. Arrays of more than one dimension shall consist of a sequence such that the index along axis 1 varies most rapidly and those along subsequent axes progressively less rapidly. Except for the location of the first element, array structure is independent of record structure.

4.4. Extensions

4.4.1. Requirements for conforming extensions

All extensions, whether or not further described in this standard, shall fulfill the following requirements to be in conformance with this *FITS* standard.

4.4.1.1. Identity. Each extension type shall have a unique type name, specified in the header according to the syntax codified in Sect. 5.4.1.2. To preclude conflict, extension type names must be registered with the IAUFWG. The *FITS* Support Office shall maintain and provide a list of the registered extensions.

4.4.1.2. Size specification. The total number of bits in the data of each extension shall be specified in the header for that extension, in the manner prescribed in Sect. 5.4.1.2.

4.4.1.3. Compatibility with existing ***FITS*** files. No extension shall be constructed that invalidates existing *FITS* files.

4.4.2. Standard extensions

A standard extension shall be a conforming extension whose organization and content are completely specified in this standard. Only one *FITS* format shall be approved for each type of data organization. Each standard extension shall have a unique name given by the value of the `XTENSION` keyword (see Appendix I).

4.4.3. Order of extensions

An extension may follow the primary HDU or another conforming extension. Standard extensions and other conforming extensions may appear in any order in a *FITS* file.

4.5. Special records

The first 8 bytes of special records must not contain the string "XTENSION". It is recommended that they not contain the string "SIMPLE␣␣". The records must have the standard *FITS* 23040-bit record length. The contents of special records are not otherwise specified by this standard.

4.6. Physical blocking

4.6.1. Bitstream devices

For bitstream devices, including but not restricted to logical file systems, *FITS* files shall be written with fixed blocks of a physical block size equal to the 23040-bit *FITS* logical record size.

4.6.2. Sequential media

4.6.2.1. Fixed block. For fixed block length sequential media, including but not restricted to optical disks (accessed as a sequential set of records), QIC format 1/4-inch cartridge tapes, and local area networks, *FITS* files shall be written as a bitstream, using the fixed block size of the medium. If the end of the last logical record does not coincide with the end of a physical fixed block, all bits in the remainder of the physical block containing the last logical record shall be set to zero. After an end-of-file mark has been detected in the course of reading a *FITS* file, subsequent incomplete *FITS* logical records should be disregarded.

4.6.2.2. Variable block. For variable block length sequential media, including but not restricted to 1/2-inch 9-track tapes, DAT 4 mm cartridge tapes, and 8 mm cartridge tapes, *FITS* files may be written with an integer blocking factor equal to 1–10 logical records per physical block.

5. Headers

5.1. Card images

5.1.1. Syntax

Header card images shall consist of a keyword, a value indicator (optional unless a value is present), a value (optional), and a comment (optional). Except where specifically stated otherwise in this standard, keywords may appear in any order.

A formal syntax, giving a complete definition of the syntax of *FITS* card images, is given in Appendix A. It is intended as an aid in interpreting the text defining the standard.

5.1.2. Components

5.1.2.1. Keyword (bytes 1–8). The keyword shall be a left justified, 8-character, blank-filled, ASCII string with no embedded blanks. All digits (hexadecimal 30 to 39,"0123456789") and upper case Latin alphabetic characters (hexadecimal 41 to 5A, "ABCDEFG HIJKLMN OPQRST UVWXYZ") are permitted; no lower case characters shall be used. The underscore (hexadecimal 5F, "_") and hyphen (hexadecimal 2D, "−") are also permitted. No other characters are permitted. For indexed keywords with a single index the counter shall not have leading zeroes.

5.1.2.2. Value indicator (bytes 9–10). If this field contains the ASCII characters "=␣", it indicates the presence of a value field associated with the keyword, unless it is a commentary keyword as defined in Sect. 5.4.2.4. If the value indicator is not present or if it is a commentary keyword then Cols. 9–80 may contain any ASCII text.

5.1.2.3. Value/comment (bytes 11–80). This field, when used, shall contain the value, if any, of the keyword, followed by optional comments. The value field may be a null field; i.e., it may consist entirely of spaces. If the value field is null, the value associated with the keyword is undefined. If a comment is present, it must be preceded by a slash (hexadecimal 2F, "/"). A space between the value and the slash is strongly recommended. The value shall be the ASCII text representation of a string or constant, in the format specified in Sect. 5.2. The comment may contain any ASCII text.

5.2. Value

The structure of the value field shall be determined by the type of the variable. The value field represents a single value and not an array of values. The value field must be in one of two formats: fixed or free. The fixed format is required for values of mandatory keywords and recommended for values of all others. This standard imposes no requirements on case sensitivity of character strings other than those explicitly specified.

5.2.1. Character string

If the value is a fixed format character string, Col. 11 shall contain a single quote (hexadecimal code 27, "'"); the string shall follow, starting in Col. 12, followed by a closing single quote (also hexadecimal code 27) that should not occur before Col. 20 and must occur in or before Col. 80. The character string shall be composed only of ASCII text. A single quote is represented within a string as two successive single quotes, e.g., O'HARA = 'O''HARA'. Leading blanks are significant; trailing blanks are not.

Free format character strings follow the same rules as fixed format character strings except that the starting and closing single quote characters may occur anywhere within Cols. 11–80. Any columns preceding the starting quote character and after Col. 10 must contain the space character.

Note that there is a subtle distinction between the following 3 keywords:

```
KEYWORD1= ''                    / null string keyword
KEYWORD2= '   '                 / blank keyword
KEYWORD3=                       / undefined keyword
```

The value of `KEYWORD1` is a null, or zero length string whereas the value of the `KEYWORD2` is a blank string (nominally a single blank character because the first blank in the string is significant, but trailing blanks are not). The value of `KEYWORD3` is undefined and has an indeterminate datatype as well, except in cases where the data type of the specified keyword is explicitly defined in this standard.

The maximum allowed length of a keyword string is 68 characters (with the opening and closing quote characters in Cols. 11 and 80, respectively). In general, no length limit less than 68 is implied for character-valued keywords.

5.2.2. Logical

If the value is a fixed format logical constant, it shall appear as a `T` or `F` in Col. 30. A logical value is represented in free format by a single character consisting of `T` or `F`. This character must be the first non-blank character in Cols. 11–80. The only characters that may follow this single character are spaces, or a slash followed by an optional comment (see Sect. 5.1.2.3).

5.2.3. Integer number

If the value is a fixed format integer, the ASCII representation shall be right justified in columns 11–30. An integer consists of a "+" (hexadecimal 2B) or "−" (hexadecimal 2D) sign, followed by one or more ASCII digits (hexadecimal 30 to 39), with no embedded spaces. The leading "+" sign is optional. Leading zeros are permitted, but are not significant. The integer representation described here is always interpreted as a signed, decimal number.

A free format integer value follows the same rules as fixed format integers except that it may occur anywhere within Cols. 11–80.

5.2.4. Real floating point number

If the value is a fixed format real floating point number, the ASCII representation shall appear, right justified, in Cols. 11–30.

A floating point number is represented by a decimal number followed by an optional exponent, with no embedded spaces. A decimal number consists of a "+" (hexadecimal 2B) or "−" (hexadecimal 2D) sign, followed by a sequence of ASCII digits containing a single decimal point (".") , representing an integer part and a fractional part of the floating point number. The leading "+" sign is optional. At least one of the integer part or fractional part must be present. If the fractional part is present, the decimal point must also be present. If only the integer part is present, the decimal point may be omitted. The exponent, if present, consists of an exponent letter followed by an integer. Letters in the exponential form ("E" or "D") shall be upper case. Note: the full precision of 64-bit values cannot be expressed over the whole range of values using the fixed format.

A free format floating point value follows the same rules as fixed format floating point values except that it may occur anywhere within Cols. 11–80.

5.2.5. Complex integer number

There is no fixed format for complex integer numbers.

If the value is a complex integer number, the value must be represented as a real part and an imaginary part, separated by a comma and enclosed in parentheses. Spaces may precede and follow the real and imaginary parts. The real and imaginary parts are represented as integers (Sect. 5.2.3). Such a representation is regarded as a single value for the complex integer number. This representation may be located anywhere within Cols. 11–80.

5.2.6. Complex floating point number

There is no fixed format for complex floating point numbers.

If the value is a complex floating point number, the value must be represented as a real part and an imaginary part, separated by a comma and enclosed in parentheses. Spaces may precede and follow the real and imaginary parts. The real and imaginary parts are represented as floating point values (Sect. 5.2.4). Such a representation is regarded as a single value for the complex floating point number. This representation may be located anywhere within Cols. 11–80.

5.3. Units

The units of all *FITS* header keyword values, with the exception of measurements of angles, should conform with the recommendations in the IAU Style Manual (McNally 1988). For angular measurements given as floating point values and specified with reserved keywords, degrees are the recommended units (with the units, if specified, given as `'deg'`).

5.4. Keywords

5.4.1. Mandatory keywords

Mandatory keywords are required in every HDU as described in the remainder of this subsection. They may

Table 1. Mandatory keywords for primary header.

1	SIMPLE
2	BITPIX
3	NAXIS
4	NAXISn, n = 1, ..., NAXIS
	⋮
	(other keywords)
	⋮
last	END

Table 2. Interpretation of valid BITPIX value.

Value	Data Represented
8	Character or unsigned binary integer
16	16-bit twos-complement binary integer
32	32-bit twos-complement binary integer
−32	IEEE single precision floating point
−64	IEEE double precision floating point

be used only as described in this standard. Values of the mandatory keywords must be written in fixed format.

5.4.1.1. Principal. The SIMPLE keyword is required to be the first keyword in the primary header of all *FITS* files. Principal mandatory keywords other than SIMPLE are required in all *FITS* headers. The card images of any primary header must contain the keywords shown in Table 1 in the order given. No other keywords may intervene between the SIMPLE keyword and the last NAXISn keyword.

The total number of bits in the primary data array, exclusive of fill that is needed after the data to complete the last record (Sect. 4.3.2), is given by the following expression:

$$N_{\rm bits} = |\texttt{BITPIX}| \times (\texttt{NAXIS1} \times \texttt{NAXIS2} \times \cdots \times \texttt{NAXISm}), \quad (1)$$

where $N_{\rm bits}$ is non-negative and the number of bits excluding fill, m is the value of NAXIS, and BITPIX and the NAXISn represent the values associated with those keywords.

SIMPLE keyword. The value field shall contain a logical constant with the value T if the file conforms to this standard. This keyword is mandatory for the primary header and is not permitted in extension headers. A value of F signifies that the file does not conform to this standard.

BITPIX keyword. The value field shall contain an integer. The absolute value is used in computing the sizes of data structures. It shall specify the number of bits that represent a data value. The only valid values of BITPIX are given in Table 2.

NAXIS keyword. The value field shall contain a non-negative integer no greater than 999, representing the number of axes in the associated data array. A value of zero signifies that no data follow the header in the HDU.

NAXISn keywords. The value field of this indexed keyword shall contain a non-negative integer, representing the number of elements along axis n of a data array. The NAXISn must be present for all values n = 1,...,NAXIS, and for no other values of n. A value of zero for any of the NAXISn signifies that no data follow the header in the HDU. If NAXIS is equal to 0, there should not be any NAXISn keywords.

END keyword. This keyword has no associated value. Columns 9–80 shall be filled with ASCII blanks.

Table 3. Mandatory keywords in conforming extensions.

1	XTENSION
2	BITPIX
3	NAXIS
4	NAXISn, n = 1, ..., NAXIS
	⋮
	(other keywords, including ...)
	PCOUNT
	GCOUNT
	⋮
last	END

5.4.1.2. Conforming extensions. All conforming extensions must use the keywords defined in Table 3 in the order specified. No other keywords may intervene between the XTENSION keyword and the last NAXISn keyword. This organization is required for any conforming extension, whether or not further specified in this standard.

The total number of bits in the extension data array exclusive of fill that is needed after the data to complete the last record such as that for the primary data array (Sect. 4.3.2) is given by the following expression:

$$N_{\rm bits} = |\texttt{BITPIX}| \times \texttt{GCOUNT} \times (\texttt{PCOUNT} + \texttt{NAXIS1} \times \texttt{NAXIS2} \times \cdots \times \texttt{NAXISm}), \quad (2)$$

where $N_{\rm bits}$ is non-negative and the number of bits excluding fill, m is the value of NAXIS, and BITPIX, GCOUNT, PCOUNT, and the NAXISn represent the values associated with those keywords.

XTENSION keyword. The value field shall contain a character string giving the name of the extension type. This keyword is mandatory for an extension header and must not appear in the primary header. For an extension that is not a standard extension, the type name must not be the same as that of a standard extension.

The IAUFWG may specify additional type names that must be used only to identify specific types of extensions; the full list shall be available from the *FITS* Support Office.

PCOUNT keyword. The value field shall contain an integer that shall be used in any way appropriate to define the data structure, consistent with Eq. (2).

GCOUNT keyword. The value field shall contain an integer that shall be used in any way appropriate to define the data structure, consistent with Eq. (2).

EXTEND keyword. The use of extensions necessitates a single additional keyword in the primary header of the *FITS* file. If the *FITS* file may contain extensions, a card image with the keyword EXTEND and the value field containing the logical value T must appear in the primary header immediately after the last NAXISn card image, or, if NAXIS=0, the NAXIS card image. The presence of this keyword with the value T in the primary header does not require that extensions be present.

5.4.2. Other reserved keywords

These keywords are optional but may be used only as defined in this standard. They apply to any *FITS* structure with the meanings and restrictions defined below. Any *FITS* structure may further restrict the use of these keywords.

5.4.2.1. Keywords describing the history or physical construction of the HDU

DATE keyword. Starting January 1, 2000, the following format shall be used. *FITS* writers should commence writing the value of the DATE keyword in this format starting January 1, 1999 and before January 1, 2000. The value field shall contain a character string giving the date on which the HDU was created, in the form YYYY-MM-DD, or the date and time when the HDU was created, in the form YYYY-MM-DDThh:mm:ss[.sss...], where YYYY shall be the four-digit calendar year number, MM the two-digit month number with January given by 01 and December by 12, and DD the two-digit day of the month. When both date and time are given, the literal T shall separate the date and time, hh shall be the two-digit hour in the day, mm the two-digit number of minutes after the hour, and ss[.sss...] the number of seconds (two digits followed by an optional fraction) after the minute. No fields may be defaulted and no leading zeroes omitted. The decimal part of the seconds field is optional and may be arbitrarily long, so long as it is consistent with the rules for value formats of Sect. 5.2.

The value of the DATE keyword shall always be expressed in UTC when in this format, for all data sets created on earth.

The following format may appear on files written before January 1, 2000. The value field contains a character string giving the date on which the HDU was created, in the form DD/MM/YY, where DD is the day of the month, MM the month number with January given by 01 and December by 12, and YY the last two digits of the year, the first two digits being understood to be 19. Specification of the date using Universal Time is recommended but not assumed.

Copying of a *FITS* file does not require changing any of the keyword values in the file's HDUs.

ORIGIN keyword. The value field shall contain a character string identifying the organization or institution responsible for creating the *FITS* file.

BLOCKED keyword. This keyword may be used only in the primary header. It shall appear within the first 36 card images of the *FITS* file. (Note: this keyword thus cannot appear if NAXIS is greater than 31, or if NAXIS is greater than 30 and the EXTEND keyword is present.) Its presence with the required logical value of T advises that the physical block size of the *FITS* file on which it appears may be an integral multiple of the logical record length, and not necessarily equal to it. Physical block size and logical record length may be equal even if this keyword is present or unequal if it is absent. It is reserved primarily to prevent its use with other meanings. Since the issuance of version 1 of this standard, the BLOCKED keyword has been deprecated.

5.4.2.2. Keywords describing observations

DATE-OBS keyword. The format of the value field for DATE-OBS keywords shall follow the prescriptions for the DATE keyword (Sect. 5.4.2.1). Either the 4-digit year format or the 2-digit year format may be used for observation dates from 1900 through 1999 although the 4-digit format is preferred.

When the format with a four-digit year is used, the default interpretations for time shall be UTC for dates beginning 1972-01-01 and UT before. Other date and time scales are permissible. The value of the DATE-OBS keyword shall be expressed in the principal time system or time scale of the HDU to which it belongs; if there is any chance of ambiguity, the choice shall be clarified in comments. The value of DATE-OBS shall be assumed to refer to the start of an observation, unless another interpretation is clearly explained in the comment field. Explicit specification of the time scale is recommended. By default, times for TAI and times that run simultaneously with TAI, e.g., UTC and TT, will be assumed to be as measured at the detector (or, in practical cases, at the observatory). For coordinate times such as TCG, TCB, and TDB which are tied to an unambiguous coordinate system, the default shall be time as if the observation had taken place at the origin of the coordinate time system. Conventions may be developed that use other time systems. Appendix D of this document contains the appendix to the agreement on a four digit year, which discusses time systems in some detail.

When the value of DATE-OBS is expressed in the two-digit year form, allowed for files written before

January 1, 2000 with a year in the range 1900–1999, there is no default assumption as to whether it refers to the start, middle or end of an observation.

DATExxxx keywords. The value fields for all keywords beginning with the string DATE whose value contains date, and optionally time, information shall follow the prescriptions for the DATE-OBS keyword.

TELESCOP keyword. The value field shall contain a character string identifying the telescope used to acquire the data associated with the header.

INSTRUME keyword. The value field shall contain a character string identifying the instrument used to acquire the data associated with the header.

OBSERVER keyword. The value field shall contain a character string identifying who acquired the data associated with the header.

OBJECT keyword. The value field shall contain a character string giving a name for the object observed.

EQUINOX keyword. The value field shall contain a floating point number giving the equinox in years for the celestial coordinate system in which positions are expressed.

EPOCH keyword. The value field shall contain a floating point number giving the equinox in years for the celestial coordinate system in which positions are expressed. Starting with Version 1, this standard has deprecated the use of the EPOCH keyword and thus it shall not be used in *FITS* files created after the adoption of this standard; rather, the EQUINOX keyword shall be used.

5.4.2.3. Bibliographic keywords

AUTHOR keyword. The value field shall contain a character string identifying who compiled the information in the data associated with the header. This keyword is appropriate when the data originate in a published paper or are compiled from many sources.

REFERENC keyword. The value field shall contain a character string citing a reference where the data associated with the header are published.

5.4.2.4. Commentary keywords

COMMENT keyword. This keyword shall have no associated value; Cols. 9–80 may contain any ASCII text. Any number of COMMENT card images may appear in a header.

HISTORY keyword. This keyword shall have no associated value; Cols. 9–80 may contain any ASCII text. The text should contain a history of steps and procedures associated with the processing of the associated data. Any number of HISTORY card images may appear in a header.

Keyword hield is blank. Columns 1–8 contain ASCII blanks. Columns 9–80 may contain any ASCII text. Any number of card images with blank keyword fields may appear in a header.

5.4.2.5. Array keywords.

These keywords are used to describe the contents of an array, either alone or in a series of random groups (Sect. 7). They are optional, but if they appear in the header describing an array or groups, they must be used as defined in this section of this standard. They shall not be used in headers describing other structures unless the meaning is the same as that for a primary or groups array.

BSCALE keyword. This keyword shall be used, along with the BZERO keyword, when the array pixel values are not the true physical values, to transform the primary data array values to the true physical values they represent, using Eq. (3). The value field shall contain a floating point number representing the coefficient of the linear term in the scaling equation, the ratio of physical value to array value at zero offset. The default value for this keyword is 1.0.

BZERO keyword. This keyword shall be used, along with the BSCALE keyword, when the array pixel values are not the true physical values, to transform the primary data array values to the true values. The value field shall contain a floating point number representing the physical value corresponding to an array value of zero. The default value for this keyword is 0.0.

The transformation equation is as follows:

$$\text{physical\_value} = \texttt{BZERO} + \texttt{BSCALE} \times \text{array\_value}. \quad (3)$$

BUNIT keyword. The value field shall contain a character string, describing the physical units in which the quantities in the array, after application of BSCALE and BZERO, are expressed. These units must follow the prescriptions of Sect. 5.3.

BLANK keyword. This keyword shall be used only in headers with positive values of BITPIX (i.e., in arrays with integer data). Columns 1–8 contain the string, "BLANK␣␣␣" (ASCII blanks in Cols. 6–8). The value field shall contain an integer that specifies the representation of array values whose physical values are undefined.

CTYPEn keywords. The value field shall contain a character string, giving the name of the coordinate represented by axis n.

CRPIXn keywords. The value field shall contain a floating point number, identifying the location of a reference point along axis n, in units of the axis index. This value is based upon a counter that runs from 1 to NAXISn with an increment of 1 per pixel. The reference point value need not be that for the center of a pixel nor lie within the actual data array. Use comments to indicate the location of the index point relative to the pixel.

CRVALn keywords. The value field shall contain a floating point number, giving the value of the coordinate specified by the CTYPEn keyword at the reference point CRPIXn. Units must follow the prescriptions of Sect. 5.3.

CDELTn keywords. The value field shall contain a floating point number giving the partial derivative of the coordinate specified by the CTYPEn keywords with respect to the pixel index, evaluated at the reference point CRPIXn, in units of the coordinate specified by the CTYPEn keyword. These units must follow the prescriptions of Sect. 5.3.

CROTAn keywords. This keyword is used to indicate a rotation from a standard coordinate system described by the CTYPEn to a different coordinate system in which the values in the array are actually expressed. Rules for such rotations are not further specified in this standard; the rotation should be explained in comments. The value field shall contain a floating point number giving the rotation angle in degrees between axis n and the direction implied by the coordinate system defined by CTYPEn.

DATAMAX keyword. The value field shall always contain a floating point number, regardless of the value of BITPIX. This number shall give the maximum valid physical value represented by the array, exclusive of any special values.

DATAMIN keyword. The value field shall always contain a floating point number, regardless of the value of BITPIX. This number shall give the minimum valid physical value represented by the array, exclusive of any special values.

5.4.2.6. Extension keywords. These keywords are used to describe an extension and should not appear in the primary header.

EXTNAME keyword. The value field shall contain a character string, to be used to distinguish among different extensions of the same type, i.e., with the same value of XTENSION, in a *FITS* file.

EXTVER keyword. The value field shall contain an integer, to be used to distinguish among different extensions in a *FITS* file with the same type and name, i.e., the same values for XTENSION and EXTNAME. The values need not start with 1 for the first extension with a particular value of EXTNAME and need not be in sequence for subsequent values. If the EXTVER keyword is absent, the file should be treated as if the value were 1.

EXTLEVEL keyword. The value field shall contain an integer, specifying the level in a hierarchy of extension levels of the extension header containing it. The value shall be 1 for the highest level; levels with a higher value of this keyword shall be subordinate to levels with a lower value. If the EXTLEVEL keyword is absent, the file should be treated as if the value were 1.

5.4.3. Additional keywords

5.4.3.1. Requirements. New keywords may be devised in addition to those described in this standard, so long as they are consistent with the generalized rules for keywords and do not conflict with mandatory or reserved keywords.

5.4.3.2. Restrictions. No keyword in the primary header shall specify the presence of a specific extension in a *FITS* file; only the EXTEND keyword described in Sect. 5.4.1.2 shall be used to indicate the possible presence of extensions. No keyword in either the primary or extension header shall explicitly refer to the physical block size, other than the deprecated BLOCKED keyword of Sect. 5.4.2.1.

6. Data representation

Primary and extension data shall be represented in one of the formats described in this section. *FITS* data shall be interpreted to be a byte stream. Bytes are in order of decreasing significance. The byte that includes the sign bit shall be first, and the byte that has the ones bit shall be last.

6.1. Characters

Each character shall be represented by one byte. A character shall be represented by its 7-bit ASCII (ANSI 1977) code in the low order seven bits in the byte. The high-order bit shall be zero.

6.2. Integers

6.2.1. Eight-bit

Eight-bit integers shall be unsigned binary integers, contained in one byte.

6.2.2. Sixteen-bit

Sixteen-bit integers shall be twos-complement signed binary integers, contained in two bytes.

6.2.3. Thirty-two-bit

Thirty-two-bit integers shall be twos-complement signed binary integers, contained in four bytes.

6.2.4. Unsigned integers

Unsigned sixteen-bit integers can be represented in *FITS* files by subtracting 32 768 from each value (thus offsetting the values into the range of a signed sixteen-bit integer) before writing them to the *FITS* file. The `BZERO` keyword (or the `TZEROn` keyword in the case of binary table columns with `TFORMn = 'I'`) must also be included in the header with its value set to 32 768 so that *FITS* reading software will add this offset to the raw values in the *FITS* file, thus restoring them to the original unsigned integer values. Unsigned thirty-two-bit integers can be represented in *FITS* files in a similar way by applying an offset of 2147483648 (2^{31}) to the data values.

6.3. IEEE-754 floating point

Transmission of 32- and 64-bit floating point data within the *FITS* format shall use the ANSI/IEEE-754 standard (IEEE 1985). `BITPIX = -32` and `BITPIX = -64` signify 32- and 64-bit IEEE floating point numbers, respectively; the absolute value of `BITPIX` is used for computing the sizes of data structures. The full IEEE set of number forms is allowed for *FITS* interchange, including all special values.

The `BLANK` keyword should not be used when `BITPIX = -32` or `-64`; rather, the IEEE NaN should be used to represent an undefined value. Use of the `BSCALE` and `BZERO` keywords is not recommended.

Appendix H has additional details on the IEEE format.

7. Random groups structure

Although it is standard *FITS*, the random groups structure has been used almost exclusively for applications in radio interferometry; outside this field, few *FITS* readers can read data in random groups format. The binary table extension (Sect. 8.3) can accommodate the structure described by random groups. While existing *FITS* files use the format, and it is therefore included in this standard, its use for future applications has been deprecated since the issue of Version 1 of this standard.

7.1. Keywords

7.1.1. Mandatory keywords

The `SIMPLE` keyword is required to be the first keyword in the primary header of all *FITS* files, including those with random groups records. If the random groups format records follow the primary header, the card images of the primary header must use the keywords defined in Table 4 in the order specified. No other keywords may intervene between the `SIMPLE` keyword and the last `NAXISn` keyword.

Table 4. Mandatory keywords in primary header preceding random groups.

1	`SIMPLE`
2	`BITPIX`
3	`NAXIS`
4	`NAXIS1`
5	`NAXISn`, `n`=2, ..., value of `NAXIS`
	⋮
	(other keywords, which must include ...)
	`GROUPS`
	`PCOUNT`
	`GCOUNT`
	⋮
last	`END`

The total number of bits in the random groups records exclusive of the fill described in Sect. 7.2 is given by the following expression:

$$N_{\rm bits} = |\texttt{BITPIX}| \times \texttt{GCOUNT} \times (\texttt{PCOUNT} + \texttt{NAXIS2} \times \texttt{NAXIS3} \times \cdots \times \texttt{NAXISm}), \quad (4)$$

where $N_{\rm bits}$ is non-negative and the number of bits excluding fill, `m` is the value of `NAXIS`, and `BITPIX`, `GCOUNT`, `PCOUNT`, and the `NAXISn` represent the values associated with those keywords.

7.1.1.1. `SIMPLE` keyword. The card image containing this keyword is structured in the same way as if a primary data array were present (Sect. 5.4.1).

7.1.1.2. `BITPIX` keyword. The card image containing this keyword is structured as prescribed in Sect. 5.4.1.

7.1.1.3. `NAXIS` keyword. The value field shall contain an integer ranging from 1 to 999, representing one more than the number of axes in each data array.

7.1.1.4. `NAXIS1` keyword. The value field shall contain the integer 0, a signature of random groups format indicating that there is no primary data array.

7.1.1.5. `NAXISn` Keywords (`n` = 2, ..., value of `NAXIS`). The value field shall contain an integer, representing the number of positions along axis `n-1` of the data array in each group.

7.1.1.6. `GROUPS` keyword. The value field shall contain the logical constant `T`. The value `T` associated with this keyword implies that random groups records are present.

7.1.1.7. PCOUNT keyword. The value field shall contain an integer equal to the number of parameters preceding each array in a group.

7.1.1.8. GCOUNT keyword. The value field shall contain an integer equal to the number of random groups present.

7.1.1.9. END keyword. The card image containing this keyword is structured as described in Sect. 5.4.1.

7.1.2. Reserved keywords

7.1.2.1. PTYPEn keywords. The value field shall contain a character string giving the name of parameter n. If the PTYPEn keywords for more than one value of n have the same associated name in the value field, then the data value for the parameter of that name is to be obtained by adding the derived data values of the corresponding parameters. This rule provides a mechanism by which a random parameter may have more precision than the accompanying data array elements; for example, by summing two 16-bit values with the first scaled relative to the other such that the sum forms a number of up to 32-bit precision.

7.1.2.2. PSCALn keywords. This keyword shall be used, along with the PZEROn keyword, when the nth *FITS* group parameter value is not the true physical value, to transform the group parameter value to the true physical values it represents, using Eq. (5). The value field shall contain a floating point number representing the coefficient of the linear term in Eq. (5), the scaling factor between true values and group parameter values at zero offset. The default value for this keyword is 1.0.

7.1.2.3. PZEROn keywords. This keyword shall be used, along with the PSCALn keyword, when the nth *FITS* group parameter value is not the true physical value, to transform the group parameter value to the physical value. The value field shall contain a floating point number, representing the true value corresponding to a group parameter value of zero. The default value for this keyword is 0.0. The transformation equation is as follows:

$$\text{physical\_value} = \texttt{PZEROn} + \texttt{PSCALn} \times \text{group\_parameter\_value}. \quad (5)$$

7.2. Data sequence

Random groups data shall consist of a set of groups. The number of groups shall be specified by the GCOUNT keyword in the associated header record. Each group shall consist of the number of parameters specified by the PCOUNT keyword followed by an array with the number of elements $N_{\rm elem}$ given by the following expression:

$$N_{\rm elem} = (\texttt{NAXIS2} \times \texttt{NAXIS3} \times \cdots \times \texttt{NAXISm}), \quad (6)$$

where $N_{\rm elem}$ is the number of elements in the data array in a group, m is the value of NAXIS, and the NAXISn represent the values associated with those keywords.

The first parameter of the first group shall appear in the first location of the first data record. The first element of each array shall immediately follow the last parameter associated with that group. The first parameter of any subsequent group shall immediately follow the last element of the array of the previous group. The arrays shall be organized internally in the same way as an ordinary primary data array. If the groups data do not fill the final record, the remainder of the record shall be filled with zero values in the same way as a primary data array (Sect. 4.3.2). If random groups records are present, there shall be no primary data array.

7.3. Data representation

Permissible data representations are those listed in Sect. 6. Parameters and elements of associated data arrays shall have the same representation. Should more precision be required for an associated parameter than for an element of a data array, the parameter shall be divided into two or more addends, represented by the same value for the PTYPEn keyword. The value shall be the sum of the physical values, which may have been obtained from the group parameter values using the PSCALn and PZEROn keywords.

8. Standard extensions

8.1. The ASCII table extension

Data shall appear as an ASCII table extension if the primary header of the *FITS* file has the keyword EXTEND set to T and the first keyword of that extension header has XTENSION=␣'TABLE␣␣␣'.

8.1.1. Mandatory keywords

The header of an ASCII table extension must use the keywords defined in Table 5. The first keyword must be XTENSION; the seven keywords following XTENSION (BITPIX ... TFIELDS) must be in the order specified with no intervening keywords.

XTENSION keyword. The value field shall contain the character string value text 'TABLE␣␣␣'.

BITPIX keyword. The value field shall contain the integer 8, denoting that the array contains ASCII characters.

NAXIS keyword. The value field shall contain the integer 2, denoting that the included data array is two-dimensional: rows and columns.

Table 5. Mandatory keywords in ASCII table extensions.

1	XTENSION
2	BITPIX
3	NAXIS
4	NAXIS1
5	NAXIS2
6	PCOUNT
7	GCOUNT
8	TFIELDS
	⋮
	(other keywords, which must include ...)
	TBCOLn, n=1, 2, ..., k where k is the value of TFIELDS
	TFORMn, n=1, 2, ..., k where k is the value of TFIELDS
	⋮
last	END

NAXIS1 keyword. The value field shall contain a non-negative integer, giving the number of ASCII characters in each row of the table.

NAXIS2 keyword. The value field shall contain a non-negative integer, giving the number of rows in the table.

PCOUNT keyword. The value field shall contain the integer 0.

GCOUNT keyword. The value field shall contain the integer 1; the data records contain a single table.

TFIELDS keyword. The value field shall contain a non-negative integer representing the number of fields in each row. The maximum permissible value is 999.

TBCOLn keywords. The value field of this indexed keyword shall contain an integer specifying the column in which field n starts. The first column of a row is numbered 1.

TFORMn keywords. The value field of this indexed keyword shall contain a character string describing the format in which field n is encoded. Only the formats in Table 6, interpreted as ANSI FORTRAN-77 (ANSI 1978) input formats and discussed in more detail in Sect. 8.1.5, are permitted for encoding. Format codes must be specified in upper case. Other format editing codes common to ANSI FORTRAN-77 such as repetition, positional editing, scaling, and field termination are not permitted. All values in numeric fields have a number base of ten (i.e., they are decimal); binary, octal, hexadecimal, and other representations are not permitted.

Table 6. Valid TFORMn format values in TABLE extensions.

Field Value	Data Type
Aw	Character
Iw	Decimal integer
Fw.d	Single precision real
Ew.d	Single precision real, exponential notation
Dw.d	Double precision real, exponential notation

END keyword. This keyword has no associated value. Columns 9–80 shall contain ASCII blanks.

8.1.2. Other reserved keywords

In addition to the mandatory keywords defined in Sect. 8.1.1, the following keywords may be used to describe the structure of an ASCII table data array. They are optional, but if they appear within an ASCII table extension header, they must be used as defined in this section of this standard.

TSCALn keywords. This indexed keyword shall be used, along with the TZEROn keyword, when the quantity in field n does not represent a true physical quantity. The value field shall contain a floating point number representing the coefficient of the linear term in Eq. (7), which must be used to compute the true physical value of the field. The default value for this keyword is 1.0. This keyword may not be used for A-format fields.

TZEROn keywords. This indexed keyword shall be used, along with the TSCALn keyword, when the quantity in field n does not represent a true physical quantity. The value field shall contain a floating point number representing the zero point for the true physical value of field n. The default value for this keyword is 0.0. This keyword may not be used for A-format fields.

The transformation equation used to compute a true physical value from the quantity in field n is

$$\text{physical\_value} = \texttt{TZEROn} + \texttt{TSCALn} \times \text{field\_value}. \quad (7)$$

TNULLn keywords. The value field for this indexed keyword shall contain the character string that represents an undefined value for field n. The string is implicitly blank filled to the width of the field.

TTYPEn keywords. The value field for this indexed keyword shall contain a character string, giving the name of field n. It is recommended that only letters, digits, and underscore (hexadecimal code 5F, "_") be used in the name. String comparisons with the values of TTYPEn keywords should not be case sensitive. The use of identical names for different fields should be avoided.

TUNITn keywords. The value field shall contain a character string describing the physical units in which the quantity in field n, after any application of TSCALn and TZEROn, is expressed. Units must follow the prescriptions in Sect. 5.3.

8.1.3. Data sequence

The table is constructed from a two-dimensional array of ASCII characters. The row length and the number of rows shall be those specified, respectively, by the NAXIS1 and NAXIS2 keywords of the associated header records. The number of characters in a row and the number of rows in the table shall determine the size of the character array. Every row in the array shall have the same number of characters. The first character of the first row shall be at the start of the record immediately following the last header record. The first character of subsequent rows shall follow immediately the character at the end of the previous row, independent of the record structure. The positions in the last data record after the last character of the last row of the data array shall be filled with ASCII blanks.

8.1.4. Fields

Each row in the array shall consist of a sequence of fields, with one entry in each field. For every field, the ANSI FORTRAN-77 format of the information contained, location in the row of the beginning of the field and (optionally) the field name, shall be specified in keywords of the associated header records. A separate format keyword must be provided for each field. The location and format of fields shall be the same for every row. Fields may overlap. There may be characters in a table row that are not included in any field.

8.1.5. Entries

All data in an ASCII table extension field shall be ASCII text in a format that conforms to the rules for fixed field input in ANSI FORTRAN-77 (ANSI 1978) format, as described below, including implicit decimal points. The only possible formats shall be those specified in Table 6. If values of -0 and $+0$ must be distinguished, then the sign character should appear in a separate field in character format. TNULLn keywords may be used to specify a character string that represents an undefined value in each field. The characters representing an undefined value may differ from field to field but must be the same within a field. Writers of ASCII tables should select a format appropriate to the form, range of values, and accuracy of the data in the table.

The value of a character-formatted (Aw) field is a character string of width w containing the characters in columns TBCOLn through TBCOLn+w − 1.

The value of an integer-formatted (Iw) field is an integer number determined by removing all blanks from columns TBCOLn through TBCOLn+w − 1 and interpreting the remaining, right-justified characters as a signed decimal integer. A blank field has value 0. All characters other than blanks, the decimal integers ("0" through "9") and a single leading sign character ("+" and "−") are forbidden.

The value of a real-formatted field (Fw.d, Ew.d, Dw.d) is a real number determined from the w characters from columns TBCOLn through TBCOLn+w − 1. The value is formed by

1. Discarding all blank characters and right-justifying the non-blank characters;
2. Interpreting the first non-blank characters as a numeric string consisting of a single optional sign ("+" or "−") followed by one or more decimal digits ("0" through "9") optionally containing a single decimal point (".") . The numeric string is terminated by the end of the right-justified field or by the occurrence of any character other than a decimal point (".") and the decimal integers ("0" through "9"). If the string contains no explicit decimal point, then the implicit decimal point is taken as immediately preceding the rightmost d digits of the string, with leading zeros assumed if necessary;
3. If the numeric string is terminated by a
   (a) "+" or "−", interpreting the following string as an exponent in the form of a signed decimal integer, or
   (b) "E", or "D", interpreting the following string as an exponent of the form E or D followed by an optionally signed decimal integer constant;
4. The exponent string, if present, is terminated by the end of the right-justified string;
5. Characters other than those specified above are forbidden.

The numeric value of the table field is then the value of the numeric string multiplied by ten (10) to the power of the exponent string, i.e., value = numeric_string $\times$ $10^{(\text{exponent\_string})}$. The default exponent is zero and a blankfield has value zero. There is no difference between the F, D, and E formats; the content of the string determines its interpretation. Numbers requiring more precision and/or range than the local computer can support may be represented. It is good form to specify a D format in TFORMn for a column of an ASCII table when that column will contain numbers that cannot be accurately represented in 32-bit IEEE binary format (see Appendix H).

Note that the above definitions allow for embedded blanks anywhere in integer-formatted and real-formatted fields and implicit decimal points in real-formatted fields. *FITS* reading tasks will have to honor these flexibilities. However, since these flexibilities are likely to cause confusion and possible misinterpretation, it is recommended that *FITS* writing tasks write tables with explicit decimal points and no embedded or trailing blanks whenever possible.

Table 7. Mandatory keywords in image extensions.

1	XTENSION
2	BITPIX
3	NAXIS
4	NAXISn, n = 1, ..., NAXIS
5	PCOUNT
6	GCOUNT
	⋮
	(other keywords ...)
	⋮
last	END

8.2. Image extension

Data shall appear as an image extension if the primary header of the *FITS* file has the keyword EXTEND set to T and the first keyword of that extension header has XTENSION=␣'IMAGE␣␣␣'.

8.2.1. Mandatory keywords

The XTENSION keyword is required to be the first keyword of all image extensions. The card images in the header of an image extension must use the keywords defined in Table 7 in the order specified. No other keywords may intervene between the XTENSION and GCOUNT keywords.

XTENSION keyword. The value field shall contain the character string value text 'IMAGE␣␣␣'.

BITPIX keyword. The value field shall contain an integer. The absolute value is used in computing the sizes of data structures. It shall specify the number of bits that represent a data value. The only valid values of BITPIX are given in Table 2.

NAXIS keyword. The value field shall contain a non-negative integer no greater than 999, representing the number of axes in the associated data array. A value of zero signifies that no data follow the header in the image extension.

NAXISn keywords. The value field of this indexed keyword shall contain a non-negative integer, representing the number of elements along axis n of a data array. The NAXISn must be present for all values n = 1, ..., NAXIS, and for no other values of n. A value of zero for any of the NAXISn signifies that no data follow the header in the image extension. If NAXIS is equal to 0, there should not be any NAXISn keywords.

PCOUNT keyword. The value field shall contain the integer 0.

Table 8. Mandatory keywords in binary table extensions.

1	XTENSION
2	BITPIX
3	NAXIS
4	NAXIS1
5	NAXIS2
6	PCOUNT
7	GCOUNT
8	TFIELDS
	⋮
	(other keywords, which must include ...)
	TFORMn, n=1, 2, ..., k where k is the value of TFIELDS
	⋮
last	END

GCOUNT keyword. The value field shall contain the integer 1; each image extension contains a single array.

END keyword. This keyword has no associated value. Columns 9–80 shall be filled with ASCII blanks.

8.2.2. Units

The units of all header keyword values in an image extension shall follow the prescriptions in Sect. 5.3.

8.2.3. Data sequence

The data format shall be identical to that of a primary data array as described in Sect. 4.3.2.

8.3. Binary table extension

Data shall appear as a binary table extension if the primary header of the *FITS* file has the keyword EXTEND set to T and the first keyword of that extension header has XTENSION=␣'BINTABLE'.

8.3.1. Mandatory keywords

The XTENSION keyword is the first keyword of all binary table extensions. The seven keywords following (BITPIX ... TFIELDS) must be in the order specified in Table 8, with no intervening keywords.

XTENSION keyword. The value field shall contain the character string 'BINTABLE'.

BITPIX keyword. The value field shall contain the integer 8, denoting that the array is an array of 8-bit bytes.

NAXIS keyword. The value field shall contain the integer 2, denoting that the included data array is two-dimensional: rows and columns.

NAXIS1 keyword. The value field shall contain a non-negative integer, giving the number of 8-bit bytes in each row of the table.

NAXIS2 keyword. The value field shall contain a non-negative integer, giving the number of rows in the table.

PCOUNT keyword. The value field shall contain the number of bytes that follow the table in the associated extension data.

GCOUNT keyword. The value field shall contain the integer 1; the data records contain a single table.

TFIELDS keyword. The value field shall contain a non-negative integer representing the number of fields in each row. The maximum permissible value is 999.

TFORMn keywords. The value field of this indexed keyword shall contain a character string of the form rTa. The repeat count r is the ASCII representation of a non-negative integer specifying the number of elements in field n. The default value of r is 1; the repeat count need not be present if it has the default value. A zero element count, indicating an empty field, is permitted. The data type T specifies the data type of the contents of field n. Only the data types in Table 9 are permitted. The format codes must be specified in upper case. For fields of type P, the only permitted repeat counts are 0 and 1. The additional characters a are optional and are not further defined in this standard. Table 9 lists the number of bytes each data type occupies in a table row. The first field of a row is numbered 1. The total number of bytes $n_{\rm row}$ in a table row, given by

$$n_{\rm row} = \sum_{i=1}^{\rm TFIELDS} r_i b_i \qquad (8)$$

where r_i is the repeat count for field i, b_i is the number of bytes for the data type in field i, and TFIELDS is the value of that keyword, must equal the value of NAXIS1.

END keyword. This keyword has no associated value. Columns 9–80 shall contain ASCII blanks.

8.3.2. Other reserved keywords

In addition to the mandatory keywords defined in Sect. 8.3.1, these keywords may be used to describe the structure of a binary table data array. They are optional, but if they appear within a binary table extension header, they must be used as defined in this section of this standard.

TTYPEn keywords. The value field for this indexed keyword shall contain a character string, giving the name of field n.

Table 9. Valid TFORMn data types in BINTABLE extensions.

TFORMn value	Description	8-bit Bytes
L	Logical	1
X	Bit	*
B	Unsigned byte	1
I	16-bit integer	2
J	32-bit integer	4
A	Character	1
E	Single precision floating point	4
D	Double precision floating point	8
C	Single precision complex	8
M	Double precision complex	16
P	Array Descriptor	8

* Number of 8-bit bytes needed to contain all bits.

It is recommended that only letters, digits, and underscore (hexadecimal code 5F, "_") be used in the name. String comparisons with the values of TTYPEn keywords should not be case sensitive. The use of identical names for different fields should be avoided.

TUNITn keywords. The value field shall contain a character string describing the physical units in which the quantity in field n, after any application of TSCALn and TZEROn, is expressed. Units must follow the prescriptions in Sect. 5.3.

TNULLn keywords. The value field for this indexed keyword shall contain the integer that represents an undefined value for field n of data type B, I, or J. The keyword may not be used if field n is of any other data type.

TSCALn keywords. This indexed keyword shall be used, along with the TZEROn keyword, when the quantity in field n does not represent a true physical quantity. It may not be used if the format of field n is A, L, or X. The interpretation for fields of type P is not defined. A proposed interpretation is described in Appendix B.1. For fields with all other data types, the value field shall contain a floating point number representing the coefficient of the linear term in Eq. (7), which is used to compute the true physical value of the field, or, in the case of the complex data types C and M, of the real part of the field, with the imaginary part of the scaling factor set to zero. The default value for this keyword is 1.0.

TZEROn keywords. This indexed keyword shall be used, along with the TSCALn keyword, when the quantity in field n does not represent a true physical quantity. It may not be used if the format of field n is A, L, or X. The interpretation for fields of type P is not defined. A proposed interpretation is described in Appendix B.1. For fields with all other data types, the value field shall contain a floating point number representing the true physical value corresponding to a value of zero in field n of the *FITS* file,

Table 10. Valid TDISPn format values in BINTABLE extensions. w is the width in characters of displayed values, m is the minimum number of digits displayed, d is the number of digits to right of decimal, and e is number of digits in exponent. The .m and Ee fields are optional.

Field Value	Data Type
Aw	Character
Lw	Logical
Iw.m	Integer
Bw.m	Binary, integers only
Ow.m	Octal, integers only
Zw.m	Hexadecimal, integers only
Fw.d	Single precision real
Ew.dEe	Single precision real, exponential notation
ENw.d	Engineering; E format with exponent multiple of 3
ESw.d	Scientific; same as EN but nonzero leading digit if not zero
Gw.dEe	General; appears as F if significance not lost, else E.
Dw.dEe	Double precision real, exponential notation

or, in the case of the complex data types C and M, in the real part of the field, with the imaginary part set to zero. The default value for this keyword is 0.0. Equation (7) is used to compute a true physical value from the quantity in field n.

TDISPn keywords. The value field of this indexed keyword shall contain a character string describing the format recommended for the display of the contents of field n. If the table value has been scaled, the physical value, derived using Eq. (7), shall be displayed. All elements in a field shall be displayed with a single, repeated format. For purposes of display, each byte of bit (type X) and byte (type B) arrays is treated a an unsigned integer. Arrays of type A may be terminated with a zero byte. Only the format codes in Table 10, discussed in Sect. 8.3.4, are permitted for encoding. The format codes must be specified in upper case. If the Bw.m, Ow.m, and Zw.m formats are not readily available to the reader, the Iw.m display format may be used instead, and if the ENw.d and ESw.d formats are not available, Ew.d may be used. The meaning of this keyword is not defined for fields of type P in this standard but may be defined in conventions using such fields.

THEAP keyword. The value field of this keyword shall contain an integer providing the separation, in bytes, between the start of the main data table and the start of a supplemental data area called the heap. The default value shall be the product of the values of NAXIS1 and NAXIS2. This keyword shall not be used if the value of PCOUNT is zero. A proposed application of this keyword is presented in Appendix B.1.

TDIMn keywords. The value field of this indexed keyword shall contain a character string describing how to interpret the contents of field n as a multidimensional array, providing the number of dimensions and the length along each axis. The form of the value is not further specified by this standard. A proposed convention is described in Appendix B.2.

8.3.3. Data sequence

The data in a binary table extension shall consist of a Main Data Table which may, but need not, be followed by additional bytes. The positions in the last data record after the last additional byte, or, if there are no additional bytes, the last character of the last row of the data array, shall be filled by setting all bits to zero.

8.3.3.1. Main data table. The table is constructed from a two-dimensional byte array. The number of bytes in a row shall be specified by the value of the NAXIS1 keyword and the number of rows shall be specified by the NAXIS2 keyword of the associated header records. Within a row, fields shall be stored in order of increasing column number, as determined from the n of the TFORMn keywords. The number of bytes in a row and the number of rows in the table shall determine the size of the byte array. Every row in the array shall have the same number of bytes. The first row shall begin at the start of the record immediately following the last header record. Subsequent rows shall begin immediately following the end of the previous row, with no intervening bytes, independent of the record structure. Words need not be aligned along word boundaries.

Each row in the array shall consist of a sequence of fields. The number of elements in each field and their data type shall be specified in keywords of the associated header records. A separate format keyword must be provided for each field. The location and format of fields shall be the same for every row. Fields may be empty, if the repeat count specified in the value of the TFORMn keyword of the header is 0. The following data types, and no others, are permitted.

Logical. If the value of the TFORMn keyword specifies data type L, the contents of field n shall consist of ASCII T indicating true or ASCII F, indicating false. A 0 byte (hexadecimal 0) indicates an invalid value.

Bit array. If the value of the TFORMn keyword specifies data type X, the contents of field n shall consist of a sequence of bits starting with the most significant bit; the bits following shall be in order of decreasing significance, ending with the least significant bit. A bit array shall be composed of an integral number of bytes, with those bits following the end of the data set to zero. No null value is defined for bit arrays.

Character. If the value of the TFORMn keyword specifies data type A, field n shall contain a character string of zero

or more members, composed of ASCII text. This character string may be terminated before the length specified by the repeat count by an ASCII NULL (hexadecimal code 00). Characters after the first ASCII NULL are not defined. A string with the number of characters specified by the repeat count is not NULL terminated. Null strings are defined by the presence of an ASCII NULL as the first character.

Unsigned 8-Bit integer. If the value of the TFORMn keyword specifies data type B, the data in field n shall consist of unsigned 8-bit integers, with the most significant bit first, and subsequent bits in order of decreasing significance. Null values are given by the value of the associated TNULLn keyword.

16-Bit integer. If the value of the TFORMn keyword specifies data type I, the data in field n shall consist of twos-complement signed 16-bit integers, contained in two bytes. The most significant byte shall be first. Within each byte the most significant bit shall be first, and subsequent bits shall be in order of decreasing significance. Null values are given by the value of the associated TNULLn keyword. Unsigned integers can be represented using the convention described in Sect. 6.2.4.

32-Bit integer. If the value of the TFORMn keyword specifies data type J, the data in field n shall consist of twos-complement signed 32-bit integers, contained in four bytes. The most significant byte shall be first, and subsequent bytes shall be in order of decreasing significance. Within each byte, the most significant bit shall be first, and subsequent bits shall be in order of decreasing significance. Null values are given by the value of the associated TNULLn keyword. Unsigned integers can be represented using the convention described in Sect. 6.2.4.

Single precision floating point. If the value of the TFORMn keyword specifies data type E, the data in field n shall consist of ANSI/IEEE-754 (IEEE 1985) 32-bit floating point numbers, as described in Appendix H. All IEEE special values are recognized. The IEEE NaN is used to represent invalid values.

Double precision floating point. If the value of the TFORMn keyword specifies data type D, the data in field n shall consist of ANSI/IEEE-754 (IEEE 1985) 64-bit double precision floating point numbers, as described in Appendix H. All IEEE special values are recognized. The IEEE NaN is used to represent invalid values.

Single precision complex. If the value of the TFORMn keyword specifies data type C, the data in field n shall consist of a sequence of pairs of 32-bit single precision floating point numbers. The first member of each pair shall represent the real part of a complex number, and the second member shall represent the imaginary part of that complex number. If either member contains a NaN, the entire complex value is invalid.

Double precision complex. If the value of the TFORMn keyword specifies data type M, the data in field n shall consist of a sequence of pairs of 64-bit double precision floating point numbers. The first member of each pair shall represent the real part of a complex number, and the second member of the pair shall represent the imaginary part of that complex number. If either member contains a NaN, the entire complex value is invalid.

Array descriptor. If the value of the TFORMn keyword specifies data type P, the data in field n shall consist of not more than one pair of 32-bit integers. The meaning of these integers is not defined by this standard. The proposed application of this data type is described in Appendix B.1.

8.3.3.2. Bytes following main table. The main data table shall be followed by zero or more bytes, as specified by the value of the PCOUNT keyword. The meaning of these bytes is not further defined by this standard. One proposed application is described in Appendix B.1.

8.3.4. Data display

Character data are encoded under format code Aw. If the character datum has length less than or equal to w, it is represented on output right-justified in a string of w characters. If the character datum has length greater than w, the first w characters of the datum are represented on output in a string of w characters. Character data are not surrounded by single or double quotation marks unless those marks are themselves part of the data value.

Logical data are encoded under format code Lw. Logical data are represented on output with the character T for true or F for false right justified in a blank-filled string of w characters. A null value may be represented by a completely blank string of w characters.

Integer data (including bit X and byte B type fields) are encoded under format codes Iw.m, Bw.m, Ow.m, and Zw.m. The default value of m is one and the ".m" is optional. The first letter of the code specifies the number base for the encoding with I for decimal (10), B for binary (2), O for octal (8), and Z for hexadecimal (16). Hexadecimal format uses the upper-case letters A through F to represent decimal values 10 through 15. The output field consists of w characters containing zero or more leading blanks followed by a minus if the internal datum is negative followed by the magnitude of the internal datum in the form of an unsigned integer constant in the specified number base with only as many leading zeros as are needed to have at least m numeric digits. Note that $m \leq w$ is allowed if all values are positive, but $m < w$ is required if any values are negative.

If the number of digits required to represent the integer datum exceeds w, then the output field consists of a string of w asterisk (*) characters.

Real data are encoded under format codes Fw.d, Ew.dEe, Dw.dEe, ENw.d, and ESw.d. In all cases, the output is a string of w characters including the decimal point, any sign characters, and any exponent including the exponent's indicators, signs, and values. If the number of digits required to represent the real datum exceeds w, then the output field consists of a string of w asterisk (*) characters. In all cases, d specifies the number of digits to appear to the right of the decimal point. The F format code output field consists of $w - d - 1$ characters containing zero or more leading blanks followed by a minus if the internal datum is negative followed by the absolute magnitude of the internal datum in the form of an unsigned integer constant. These characters are followed by a decimal point (".") and d characters giving the fractional part of the internal datum, rounded by the normal rules of arithmetic to d fractional digits. For the E and D format codes, an exponent is taken such that the fraction $0.1 \leq |\text{datum}|/10^{\text{exponent}} < 1.0$. The fraction (with appropriate sign) is output with an F format of width $w - e - 2$ characters with d characters after the decimal followed by an E or D followed by the exponent as a signed $e + 1$ character integer with leading zeros as needed. The default value of e is 2 when the Ee portion of the format code is omitted. If the exponent value will not fit in $e + 1$ characters but will fit in $e + 2$ then the E (or D) is omitted and the wider field used. If the exponent value will not fit (with a sign character) in $e + 2$ characters, then the entire w-character output field is filled with asterisks (*). The ES format code is processed in the same manner as the E format code except that the exponent is taken so that $1.0 \leq \text{fraction} < 10$. The EN format code is processed in the same manner as the E format code except that the exponent is taken to be an integer multiple of 3 and so that $1.0 \leq \text{fraction} < 1000.0$. All real format codes have number base 10. There is no difference between E and D format codes on input other than an implication with the latter of greater precision in the internal datum.

The Gw.dEe format code may be used with data of any type. For data of type integer, logical, or character, it is equivalent to Iw, Lw, or Aw, respectively. For data of type real, it is equivalent to an F format (with different numbers of characters after the decimal) when that format will accurately represent the value and is equivalent to an E format when the number (in absolute value) is either very small or very large. Specifically, for real values outside the range $0.1 - 0.5\times10^{-d-1} \leq \text{value} < 10^{d} - 0.5$, it is equivalent to Ew.dEe. For real values within the above range, it is equivalent to Fw′.d′ followed by $2 + e$ blanks, where $w' = w - e - 2$ and $d' = d - k$ for $k = 0, 1, \ldots, d$ if the real datum value lies in the range $10^{k-1}\,(1 - 0.5\times10^{-d}) \leq \text{value} \leq 10^{k}\,(1 - 0.5\times10^{-d})$.

Complex data are encoded with any of the real data formats as described above. The same format is used for the real and imaginary parts. It is recommended that the 2 values be separated by a comma and enclosed in parentheses with a total field width of $2w + 3$.

9. Restrictions on changes

Any structure that is a valid *FITS* structure shall remain a valid *FITS* structure at all future times. Use of certain valid *FITS* structures may be deprecated by this or future *FITS* standard documents.

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.)

ANSI 1978, American National Standard for Information Processing: Programming Language FORTRAN, ANSI X3.9–1978 (ISO 1539) (New York: American National Standards Institute, Inc.)

Bunclark, P., & Rots, A. 1997 (ftp://nssdc.gsfc.nasa.gov/pub/fits/year2000_agreement.txt)

Cotton, W. D., Tody, D. B., & Pence, W. D. 1995, A&AS, 113, 159

Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 371

Grosbøl, P., Harten, R. H., Greisen, E. W., & Wells, D. C. 1988, A&AS, 73, 359

Grosbøl, P., & Wells, D. C. 1994 (ftp://nssdc.gsfc.nasa.gov/pub/fits/blocking94.txt)

Harten, R. H., Grosbøl, P., Greisen, E. W., & Wells, D. C. 1988, A&AS, 73, 365

IAU Info. Bull. 1983, 49

IAU Info. Bull. 1988, 61

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., & Schlesinger, B. M. 1997 (http://fits.gsfc.nasa.gov/group.html)

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

Muñoz, J. R. 1989, ESA IUE Newslett., 32, 12

NRAO 1990, Going AIPS, National Radio Astronomy Observatory, Charlottesville, VA

Ponz, J. D., Thompson, R. W., & Muñoz, J. R. 1994, A&AS, 105, 53

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

Wells, D. C., & Grosbøl, P. 1990 (ftp://nssdc.gsfc.nasa.gov/pub/fits/fp_agree.ps)