NSDSSO

Flexible Image Transport System (*FITS*)

December, 1990

Draft Standard

NSDSSO 100-0.1

NASA Science Data Systems Standards Office
Code 933
NASA Goddard Space Flight Center
Greenbelt MD 20771

The NASA Science Data Systems Standards Office (NSDSSO) has been established to serve the space science communities in evolving cost effective, interoperable, data systems. The NSDSSO performs a number of functions designed to facilitate the recognition, development, adoption, and use of standards by the space science communities.

Approval of an NSDSSO Standard requires verification by the NSDSSO that the following requirements have been met: consensus by the technical panel; proper adjudication of the comments received from the targeted space and Earth science community; and conformance to the accreditation process.

An NSDSSO standard represents the consensus of the technical panel convened by the NASA Science Data Systems Standards Office (NSDSSO) of the National Space Science Data Center (NSSDC) of the National Aeronautics and Space Administration (NASA). Consensus is established when the NSDSSO Accreditation Panel deternines 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. NSDSSO standards are not binding as published, however they may serve as a basis for mandatory standards when adopted by NASA or other organizations.

An NSDSSO 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 NSDSSO 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 cancelled.

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

Robert J. Hanisch, Chair	Space Telescope Science Institute
Barry M. Schlesinger, Secretary	ST Systems Corporation
Lee E. Brotzman	ST Systems Corporation
Edward Kemper	ST Systems Corporation
Peter Teuben	University of Maryland
Michael E. Van Steenberg	NASA Goddard Space Flight Center
Wayne H. Warren Jr.	ST Systems Corporation
Richard A. White	NASA Goddard Space Flight Center

This standard is published and maintained by the NSDSSO. Send comments and orders for NSDSSO documents to:

NSDSSO, Code 933, NASA Goddard Space Flight Center
Greenbelt MD 20771
Internet: NSDSSO@nssdca.gsfc.nasa.gov
SPAN: nssdca::NSDSSO
301-286-3575

Contents

Appendixes

List of Tables

List of Figures

Introduction

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*, 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 structure and contents of the array. The *FITS* concept was later expanded to accommodate more complex data structures. For example, a new structure for image transfer called *random groups* was defined in which the data consists of a series of arrays, with each array accompanied by a set of associated parameters. These structures were formally endorsed by the International Astronomical Union (IAU) in 1982[1]. Provisions for data structures other than simple arrays or groups were later made. These structures appear in *extensions*, consisting of an ASCII header followed by the data structure it describes. One particular extension structure, *ASCII Tables*, was formally approved by the IAU in 1988. At the same time, a set of rules governing all extensions was endorsed, to ensure that *FITS* readers encountering an unfamiliar extension would be able to identify it, interpret it, and pass on to the next.

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* structure is to be understood as a logical structure, and not defined in terms of the physical characteristics of any particular data storage medium or media.

Section 1

Overview

1.1 Purpose

This standard formally defines the *FITS* format for data structuring and exchange. It is intended as a formal codification of the *FITS* structure that has been endorsed by the IAU for transfer of astronomical data, as described in four published papers [2,3,4,5] and in a formal agreement [6] endorsed by the IAU *FITS* Working Group under Commission 5. Minor ambiguities and inconsistencies in *FITS* as described in the original papers are eliminated.

1.2 Scope

This standard specifies the structure and content of *FITS* data sets. It specifies the content and organization of the header and data for all standard *FITS* formats: Basic *FITS*, the ASCII tables extension, and the random groups structure. It also specifies minimum structural requirements for new extensions and general principles governing the creation of new extensions. For headers, it specifies the proper syntax for card images and defines required and reserved keywords. For data, it specifies character and value representations. It defines the general rules to which new extensions are required to conform.

1.3 Applicability

All spacecraft projects and astrophysics data archives under the management of the Astrophysics Division of the National Aeronautics and Space Administration are required to make processed data available to users in the format defined by this standard, unless the Astrophysics Division specifically determines otherwise. The IAU has recommended that all astronomical computer facilities support *FITS* for the interchange of binary data. 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 and Recommendations

Following the definitions in Section 3, this document describes the overall organization of a *FITS* data set and the rules for creating new *FITS* extensions in Section 4. Subsequent sections describe the contents of *FITS* files. Section 5 describes headers, including the required keywords and the conditions under which each is required; also, reserved keywords are specified. Section 6 describes how data are represented, and Section 7 describes data structures. Each section contains information relevant to all *FITS* formats. Throughout the document, deprecation of structures or syntax is noted where relevant. Data sets containing such constructs are valid *FITS*, but these constructs should not be used in new data sets; the old data sets using them remain standard because of the principle that no *FITS* structure, once valid, shall ever become invalid.

Although it is part of the *FITS* standard, the random groups structure has not been widely used. New formats under development will replace it, and use of it is discouraged. Consequently, it is described in Appendix A rather than in the main body of the standard. One new format, *3D binary tables*, is described for informational purposes in Appendix B, which is not part of the formal standard.

While *FITS* is designed as a logical format, the astronomical community has developed recommendations for its expression on various physical media. The accepted recommendations are provided in Appendix C as a guide to *FITS* practices, but they are not part of this standard. Appendix D gives a tabular summary of the *FITS* keywords.

Section 2

References

1. IAU *Information Bulletin* No. 49, p. 14, 1983.

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

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

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

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

6. Wells, D. C. and Grosbøl, P. 1990, "Floating Point Agreement for *FITS*" (available from the authors at National Radio Astronomy Observatory, Edgemont Road Charlottesville, Virginia 22903 USA or European Southern Observatory Karl-Schwarzschild Strasse 2, D-8046 Garching bei Munchen, FRG.)

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

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

9. "American National Standard – IEEE Standard for Binary Floating Point Arithmetic". ANSI/IEEE 754–1985, Published by American National Standards Institute, Inc., New York, 1985.

10. "American National Standard for Information Processing: Unrecorded Magnetic Tape," ANSI X3.40 - 1976, Published by American National Standards Institute, Inc., New York, 1976.

11. "American National Standard for Information Processing: Magnetic Tape Labels and File Structure," ANSI X3.27 - 1978, Published by American National Standards Institute, Inc., New York, 1978.

Section 3

Definitions, Acronyms, and Symbols

␣ Used to designate an ASCII blank.

Array A sequence of data values, of zero or more dimensions.

Array value An element of an array.

ASCII Abbreviation of American National Standard Code for Information Interchange.

Basic FITS The *FITS* structure consisting of the primary header and primary data containing a single array.

Bit A single binary digit.

Byte A string of eight bits treated as a single entity.

Card image A sequence of 80 bytes with values between hexadecimal 20 and 7E inclusive, treated as a logical record.

Conforming extension An extension whose keywords and structure adhere to the requirements for conforming extensions defined in this standard.

Deprecate To express earnest disapproval of (permitted but discouraged).

FITS Element One of the components of a *FITS* data set: the primary HDU, an extension, or the special records collectively.

Entry A single value in a table.

Extension A *FITS* HDU following the primary HDU in a *FITS* file.

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.

FITS Abbreviation of Flexible Image Transport System.

FITS file A file with a structure that matches the specifications in this document.

FITS logical record A record of 23040 bits, corresponding to 2880 8-bit bytes within a *FITS* file.

Floating point A number stored internally as a mantissa and exponent, whose ASCII representation contains an explicit decimal point and may include a power-of-ten exponent.

Group Format One of two possible data organization formats for the primary HDU.

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

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

IAU Abbreviation of International Astronomical Union.

IEEE Abbreviation of Institute of Electrical and Electronic Engineers.

IEEE NaN Abbreviation of IEEE Not-a-Number value.

IEEE special values (-0, $\pm\infty$).

Keyword The first eight bytes of a header card image.

Mandatory keyword A keyword which must be used in all *FITS* data sets or a keyword required in conjunction with particular *FITS* structures.

Matrix A data array of two or more dimensions.

NSDSSO Abbreviation of NASA Science Data Systems Standards Office.

Physical value The value in physical units represented by a member of an array and possibly derived from the array value using a linear transformation.

Picture element A single location within an image array.

Pixel Abbreviation of "picture element".

Primary array The data array contained in the Primary HDU.

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

Random Group A primary HDU data structure, which consists of a sequence of individual values followed by a data array.

Random Group parameters The sequence of individual values that precede the data array in a Random Group.

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

Reference pixel The pixel along a given coordinate axis at which a value and increment are defined.

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, at the end of the *FITS* file, 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.

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

Section 4

Data Set Organization

4.1 Overall Structure

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

- Primary HDU
- Extensions (optional)
- Special records (optional)

Each *FITS* element 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* element shall be the record immediately following the last record of the preceding *FITS* element. The size of a *FITS* logical record shall be 23040 bits, corresponding to 2880 8-bit bytes.

4.2 *FITS* Element Structure

The primary HDU and every extension 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

The first component of a *FITS* data set shall be the primary header. The primary header may, but need not be, followed by primary data. The presence or absence of primary data shall be indicated by the value of the `NAXIS` keyword in the primary header (Section 5.2.1.1).

4.3.1 Primary Header

The header of a primary HDU shall consist of a series of card images in ASCII code. 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

The primary data, if present, may be either in Basic *FITS* array format or random groups format. In Basic *FITS* format, the primary data shall consist of a single data array of 1-999 dimensions. If the data array does not fill the final record, the remaining storage locations shall be filled with zero values with the same data representation as the values in the array. Random groups format, which this document deprecates, is described in Appendix A.

4.4 Extensions

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

4.4.1 Requirements for Conforming Extensions

4.4.1.1 Identity

Each extension shall have a unique name, specified in the header according to the syntax codified in Section 5.2.1.2. To preclude conflict, extension names must be registered with the NSDSSO or IAU Commission 5 who will coordinate name allocations.

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 Section 5.2.1.2.

4.4.1.3 Compatibility with Existing *FITS* Structures

No extension shall be constructed that invalidates existing *FITS* data sets.

4.4.2 Standard Extensions

A standard extension shall be a conforming extension whose structure 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.

4.4.3 Order of Extensions

Standard extensions and other conforming extensions may appear in any order in a *FITS* data set.

4.5 Special Records

The first 8 bytes of special records must not contain the string "`XTENSION`" or 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.

Section 5

Headers

5.1 Card Images

5.1.1 Syntax

Header card images shall consist of a keyword, an optional value, and an optional comment. If a value is present, column 9 shall contain the symbol "=", column 10 shall be blank, and columns 11-80 shall be as specified in the remainder of Section 5.2. If no value is present, column 9 may contain any ASCII text except the symbol "=", and columns 10- 80 may contain any ASCII text.

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 60, "_") and hyphen (hexadecimal 2D, "`-`") are also permitted. No other characters are permitted. If the keyword itself requires less than eight characters it must be filled with ASCII blanks (hexadecimal 20). The value and comments field on a card image may be restricted for particular keywords.

5.1.2.2 Value Indicator (bytes 9-10)

This field shall contain an ASCII "`=␣`" for keywords with an associated value field. If there is no associated value field, byte 9 may not contain "`=`" (hexadecimal 3D); otherwise, this field may contain any ASCII character text.

5.1.2.3 Value/Comment

This field, when used, shall contain the value, if any, of the keyword followed by optional comments. Separation of the value and comments by a slash (hexadecimal 2F, "/"), and a space between the value and the slash are strongly recommended. The value shall be the ASCII representation of a string or constant, structured as specified in Section 5.3. The comment may contain any printable 7-bit ASCII characters (hexadecimal 20 through 7E).

5.2 Keywords

5.2.1 Mandatory Keywords

Mandatory keywords are required as described in the remainder of this subsection. They may be used only as described in this standard.

5.2.1.1 Principal

Principal Mandatory keywords other than `SIMPLE` are required in all *FITS* headers. The `SIMPLE` keyword is required in all primary headers and may not appear elsewhere. The card images of any primary header must contain the keywords shown in Table 5.1 in the order given.

1	`SIMPLE`
2	`BITPIX`
3	`NAXIS`
4	`NAXISn`, $n = 1, \ldots,$ `NAXIS`
	⋮
	(other keywords)
	⋮
last	`END`

Table 5.1: Principal mandatory 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 must not appear as the first keyword in a conforming extension header.

`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 (see Table 5.2).

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

Table 5.2: Interpretation of `BITPIX` value.

`NAXIS` **Keyword** The value field shall contain an integer no greater than 999, representing the number of axes in an ordinary data array. A value of zero signifies that no data follow the header.

`NAXISn` **Keywords** The value field shall contain an integer, representing the number of positions along axis n of an ordinary data array. 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 blank.

5.2.1.2 Conforming Extensions

The use of *FITS* extensions requires the use of a single additional keyword in the primary header.

`EXTEND` **Keyword** If the *FITS* data set contains 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.

The card images of any extension header must use the keywords defined in Table 5.3 in the order specified. 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 the fill described in Section 4.3.2 must be given by the following expression:

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

Table 5.3: Mandatory keywords in conforming extensions.

$$\begin{aligned}\texttt{NBITS} &= |\texttt{BITPIX}| \times \texttt{GCOUNT} \times \\ &\quad (\texttt{PCOUNT} + \texttt{NAXIS1} \times \texttt{NAXIS2} \times \cdots \times \texttt{NAXISm}),\end{aligned}$$

where NBITS is 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. NSDSSO, in collaboration with IAU Commission 5, may specify additional names that must be used only to identify specific types of extensions.

PCOUNT Keyword The value field shall contain an integer that shall be used in any way appropriate to define the data structure, consistent with the equation for NBITS. Usage is analogous to that of the PCOUNT keyword for the Random Groups structure (Section A.1.1), which represents the number of parameters in a parameter-array group.

GCOUNT Keyword The value field shall contain an integer that shall be used in any way appropriate to define the data structure, consistent with the equation for NBITS. Usage is analogous to that of the GCOUNT keyword for the Random Groups structure (Section A.1.1), which represents the number of parameter-array groups.

END Keyword This keyword has no associated value. Columns 9-80 shall be blank.

5.2.1.3 ASCII Tables Extension Keywords

The card images in the header of an ASCII Tables Extension must use the keywords defined in Table 5.4 in the order specified.

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`

Table 5.4: Mandatory keywords in ASCII tables extensions.

`XTENSION` **Keyword** The value field shall contain the character string `'TABLE␣␣␣'`.

`BITPIX` **Keyword** The value field shall contain the integer 8, denoting that the array contains 8-bit printable ASCII characters with the eighth bit, the parity bit, set to zero (i.e. hexadecimal codes 20 through 7E).

`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 an integer, representing the width of the table in characters.

`NAXIS2` **Keyword** The value field shall contain an integer of value 1 or greater, representing the number of rows in the table.

PCOUNT Keyword The value field shall contain the integer 0; no data values precede the table.

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

TFIELDS Keyword The value field shall contain an integer representing the number of fields in each row. The maximum permissible value is 999.

TBCOLn Keywords The value field shall contain an integer specifying the column in which field n starts.

TFORMn Keywords The value field shall contain a character string describing the FORTRAN-77 [7] format in which field n is coded. The formats in Table 5.5 may be used for encoding.

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

Table 5.5: Valid TFORMn format values in TABLE extensions.

Repetition of a format from one field to the next must be indicated by using separate pairs of TBCOLn and TFORMn keywords for each field; format repetition may not be indicated by prefixing the format by a number.

END Keyword This keyword has no associated value. Columns 9-80 shall be blank.

5.2.2 Other Reserved Keywords

These keywords are optional but may be used only as defined in this standard. These keywords apply to any *FITS* structure except where specifically further restricted.

5.2.2.1 Keywords Describing the History or Physical Construction of the HDU

`DATE` **Keyword** The value field shall contain a character string giving the date on which the HDU was created, in the form `DD/MM/YY`, where `DD` shall be 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. For all data sets created after the adoption of this standard, the date shall be that specified by local civil time at the location where the data set was created. 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 creating the data set.

`BLOCKED` **Keyword** This keyword may be used only in the primary header. It must appear immediately after the `EXTEND` card image, or if this card image is not present, after the last `NAXISn` card image or the `NAXIS` card image with value zero. Its presence with the required logical value of `T` advises that the physical block size of the *FITS* data set on which it appears may be an integral multiple of the logical record length. 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. The issuance of this standard deprecates the `BLOCKED` keyword.

5.2.2.2 Keywords Describing Observations

`DATE-OBS` **Keyword** The value field shall contain a character string giving the day on which the observations represented by the array were made, in Universal Time in the form form `DD/MM/YY`, where `DD` shall be 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.

`TELESCOP` **Keyword** The value field shall contain a character string identifying the telescope used to acquire the data contained in the array.

`INSTRUME` **Keyword** The value field shall contain a character string identifying the instrument used to acquire the data contained in the array.

`OBSERVER` **Keyword** The value field shall contain a character string identifying the individual who acquired the data associated with the header. This keyword is appropriate when the data describe the results of observations.

`OBJECT` **Keyword** The value field shall contain a character string giving the name of the object observed.

`EQUINOX` **Keyword** The value field shall contain a character string giving the equinox for the celestial coordinate system in which positions given in either the header or data are expressed.

`EPOCH` **Keyword** The value field shall contain a character string giving the equinox for the celestial coordinate system in which positions given in either the header or data are expressed. This document deprecates the use of the `EPOCH` keyword and thus it shall not be used in data sets created after the adoption of this standard; rather, the `EQUINOX` keyword shall be used.

5.2.2.3 Bibliographic Keywords

`AUTHOR` **Keyword** The value field shall contain a character string identifying the individual 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 a compilation of results 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.

5.2.2.4 Commentary Keywords

`COMMENT` **Keyword** This keyword shall have no associated value; columns 9-80 may contain any ASCII text, except that column 9 may not contain "=". Any number of `COMMENT` records may appear in a header.

`HISTORY` **Keyword** This keyword shall have no associated value; columns 9-80 may contain any ASCII text, except that column 9 may not contain "=". The text should contain a history of steps and procedures associated with the processing of the associated data. Any number of `HISTORY` records may appear in a header.

Keyword Field is Blank Columns 1-8 are blank. Columns 9-80 may contain any ASCII text except that column 9 may not contain "=", as for a keyword with no associated value. Any number of blank records may appear in a header.

5.2.2.5 Array Keywords

These keywords are used to describe the contents of a simple array, either alone, or in a series of random groups. They are optional, but if they appear in the header describing an array, 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 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 array values to the true physical values they represent, using the linear transformation equation defined below. 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.

`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 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 transformation equation is as follows:

$$\text{physical value} = \texttt{BZERO} + \texttt{BSCALE} \times \text{array value}$$

`BUNIT` Keyword The value field shall contain a character string, describing the ultimate physical units in which the quantities of the array are expressed.

`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␣␣␣`" (blanks in columns 6-8). The value field shall contain an integer that specifies the representation of array members whose data values are undefined.

`CTYPEn` Keywords The value field shall contain a character string, giving the physical 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. The reference point need not coincide with the center of a pixel (i.e. need not be an integer), nor lie within the actual data array.

`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 pixel `CRPIXn`.

`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 pixel, `CRPIXn`, in units of the coordinate specified by the `CTYPEn` keyword.

`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 in the array.

`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 in the array.

5.2.2.6 Extension Keywords

These keywords are used to describe an extension.

`EXTNAME` **Keyword** The value field shall contain a character string, to be used to distinguish among different extensions with the same value of `XTENSION` in a *FITS* data set.

`EXTVER` **Keyword** The value field shall contain an integer, to be used to distinguish among different extensions in a *FITS* data set with 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 data set 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 data set should be treated as if the value were 1.

5.2.2.7 ASCII Tables Keywords

In addition to the mandatory keywords defined in section 5.2.1, these keywords may be used to describe the structure of an ASCII Tables data array. They are optional, but if they appear within an ASCII tables extension header, they must be used as defined in this section of this standard.

`TSCALn` Keywords This 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 the linear transformation equation below, 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 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

$$\text{physical value} = \mathtt{TZEROn} + \mathtt{TSCALn} \times \text{field value}.$$

`TNULLn` Keywords The value field shall contain the character string that represents an undefined value for field n. The character value should be in the format given by the `TFORMn` keyword for field n.

`TTYPEn` Keywords The value field shall contain a character string, giving the name of field n. It is recommended that only upper case letters, digits and underscore (hexadecimal code 5F) be used in the name. The use of identical names for different fields should be avoided.

`TUNITn` Keywords The value field shall contain a character string describing the ultimate physical units in which the quantity in field n is expressed.

5.2.3 Additional Keywords

5.2.3.1 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.

5.2.3.2 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 Section 5.2.1.2 shall be used to indicate the possible presence of extensions. No keyword in either the primary or extension header shall explicitly specify the physical block size, other than the `BLOCKED` keyword of Section 5.2.2.1

5.3 Value/Comment

The structure of the value field shall be determined by the type of the variable.

5.3.1 Character String

If the value is a character string, column 11 shall contain a single quote (hexadecimal code 27, "'"); the string shall follow, starting in column 12, followed by a closing single quote (also hexadecimal code 27) that should not occur before column 20 and must occur in or before column 80. Proper interpretation of the data set should not require decoding any more than the first eight characters of a character string. The Character string can only be composed of the printable ASCII characters (hexadecimal codes 20 through 7E).

5.3.2 Logical Variable

If the value is a logical constant it shall appear as a `T` or `F` in column 30.

5.3.3 Integer

If the value is an integer, the ASCII representation shall appear right justified in columns 11-30. For a complex integer, the imaginary part shall be right justified in columns 31-50.

5.3.4 Real Floating Point Number

If the value is a real floating point number, the ASCII representation shall appear in columns 11-30. Letters in the exponential form shall be upper case. The value shall be right justified, and the decimal point must appear. Note: The full precision of 64-bit values can not be expressed as a single value following the rules of this standard.

5.3.5 Complex Floating Point Number

5.3.5.1 Real Part

If the value is a complex floating point number, the ASCII representation of the real part shall appear in the same manner as a real floating point number (see above).

5.3.5.2 Imaginary Part

The ASCII representation of the imaginary part of a complex floating point number shall appear in columns 31 - 50. Letters in the exponential form shall be upper case. The value shall be right justified, and the decimal point must appear. Note: The full precision of 64-bit values can not be expressed as a single value following the rules of this standard.

Section 6

Data Representation

Primary and extension data shall appear in one of the formats described in this section.

6.1 Characters

Each character shall be represented by one byte. A character shall be written in 7-bit ASCII [8], right justified in the byte. The high-order parity bit shall be zero.

6.2 Integers

6.2.1 Eight-bit

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

6.2.2 Sixteen-bit

Sixteen-bit integers shall be twos-complement signed binary integers, contained in two bytes. The sign bit shall be the leftmost bit of the leftmost byte. Significance shall decrease from left to right; the least significant bit shall be the rightmost bit of the rightmost byte.

6.2.3 Thirty-two-bit

Thirty-two bit integers shall be twos-complement signed binary integers, contained in four bytes. The sign bit shall be the leftmost bit of the leftmost byte. Significance shall decrease from left to right; the least significant bit shall be the rightmost bit of the rightmost byte.

6.3 IEEE-754 Floating Point

Transmission of 32- and 64-bit floating point data within the *FITS* format will use the ANSI/IEEE-754 standard [9]. `BITPIX = -32` and `BITPIX = -64` signify 32- and 64-bit IEEE floating point numbers; the absolute value of `BITPIX` is used for computing the sizes of data structures. The full IEEE set of number forms is allowed for *FITS* interchange, including all special values (e.g., the "Not-a-Number" cases). The order of the bytes will be sign and exponent first, followed by the mantissa bytes in order of decreasing significance. The `BLANK` keyword should not be used when `BITPIX = -32` or `-64`.

6.3.1 Thirty-two-bit Floating Point

6.3.1.1 Interpretation

Lowest numbered (leftmost) bits are most significant. The IEEE NaN (Not-a-Number) values shall be used to represent undefined values. All IEEE special values are recognized.

$$value \quad = \quad (-1)^{\mathrm{sign}} \times 2^{(\mathrm{exponent}-127)} \times \mathrm{mantissa}$$

6.3.1.2 Structure

Table 6.1 describes the bit structure of 32-bit floating point values.

Bit Positions (left to right)	Content
1	sign
2 - 9	exponent
10 - 32	mantissa

Table 6.1: Content of 32-bit floating point bit positions.

6.3.2 Sixty-four-bit Floating Point

6.3.2.1 Interpretation

Lowest numbered (leftmost) bits are most significant. The IEEE NaN (Not-a-Number) values shall be used to represent undefined values. All IEEE special values are recognized.

$$value = (-1)^{\text{sign}} \times 2^{(\text{exponent}-1023)} \times \text{mantissa}$$

6.3.2.2 Structure

Table 6.2 describes the bit structure of 64-bit floating point values.

Bit Positions (left to right)	Content
1	sign
2 - 12	exponent
13 - 64	mantissa

Table 6.2: Content of 64-bit floating point bit positions.

Section 7

Data Records

7.1 Primary Array

The data values shall be stored as a byte stream, with no embedded fill or blank space. The first value shall be stored in the first position of the first array record. The first value of each subsequent row of the array shall be stored in the position immediately following the last value of the previous row, i.e., the data values are stored as a byte stream, with no embedded fill or blank space. A primary array, when present, shall consist of an array of 1 to 999 dimensions. Arrays of more than one dimension shall be stored in 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 n 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 Figure 7.1, where m is the value of `NAXIS`. The remainder of the last data record, following the last element of the array shall be filled as prescribed in Section 4.3.2.

7.2 ASCII Tables Extension

Data shall appear as an ASCII Tables 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␣␣␣'`.

7.2.1 Storage

The data shall be stored as a two-dimensional character array. The row length and the number of rows shall be specified by keywords in 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

$$
\begin{array}{c}
\mathrm{A}(1, 1, \ldots, 1), \\
\mathrm{A}(2, 1, \ldots, 1), \\
\vdots, \\
\mathrm{A}(\mathtt{NAXIS1}, 1, \ldots, 1), \\
\mathrm{A}(1, 2, \ldots, 1), \\
\mathrm{A}(2, 2, \ldots, 1), \\
\vdots, \\
\mathrm{A}(\mathtt{NAXIS1}, 2, \ldots, 1), \\
\vdots \\
\mathrm{A}(1, \mathtt{NAXIS2}, \ldots, \mathtt{NAXISm}), \\
\vdots, \\
\mathrm{A}(\mathtt{NAXIS1}, \mathtt{NAXIS2}, \ldots, \mathtt{NAXISm})
\end{array}
$$

Figure 7.1: Arrays of more than one dimension shall be stored in 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.

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.

7.2.2 Fields

Each row in the array shall consist of a sequence of fields, with one entry in each field. For every field, the format of the stored information, 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.

7.2.3 Entries

All data in an ASCII tables extension record shall be 7-bit printable ASCII characters with the eighth (parity) bit set to zero (hexadecimal codes 20 through 7E). The only possible formats shall be FORTRAN I, A, F, E, or D. If values of -0 and +0 must be distinguished, then the sign character should be stored in a separate field in character format. `TNULL` 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.

Section 8

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.

Appendix A

Random Groups Structure

(This Appendix is a part of the NSDSSO *FITS* Standard)

Although it is part of the *FITS* standard, the random groups structure for the primary data has been used almost exclusively for applications in radio interferometry; outside this field, few *FITS* readers can read data in random groups format. The evolving binary tables format will eventually be able to accommodate the structure described by random groups. While existing *FITS* data sets use the format, and it is therefore included in this standard, its use for future applications is deprecated by this document.

A.1 Keywords

A.1.1 Mandatory Keywords

If the primary data are in Random Groups format, the card images of the primary header must use the following keywords in the order given here.

1. `SIMPLE`

2. `BITPIX`

3. `NAXIS`

3. + n (n=1, ..., value of `NAXIS`). `NAXISn`

   ⋮

   (other keywords, including)

   `GROUPS`

PCOUNT

GCOUNT

⋮

last END

A.1.1.1 SIMPLE Keyword

The card image containing this keyword is structured in the same way as for a simple array (Section 5.2.1).

A.1.1.2 BITPIX Keyword

The card image containing this keyword is structured in the same way as for a simple array (Section 5.2.1).

A.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.

A.1.1.4 NAXIS1 Keyword

The value field shall contain the integer 0, as a signature of random groups format.

A.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 each ordinary data array.

A.1.1.6 GROUPS Keyword

The value field shall contain the logical constant T. The value T associated with this keyword implies that the primary array is in random groups format.

A.1.1.7 PCOUNT Keyword

The value field shall contain an integer equal to the number of parameters preceding each group.

A.1.1.8 `GCOUNT` Keyword

The value field shall contain an integer equal to the number of random groups in the data array.

A.1.1.9 `END` Keyword

The card image containing this keyword is structured in the same way as for a simple array (Section 5.2.1).

A.1.2 Reserved Keywords

A.1.2.1 `PTYPEn` Keywords

The value field shall contain a character string, describing the physical quantity or coordinate represented by the value for parameter n, as derived from the quantity stored in the *FITS* data set and the `PSCALn` and `PZEROn` keywords. If the `PTYPEn` keywords for more than one value of n have the same associated value, then the data value for the physical quantity or coordinate so defined is to be obtained by adding the derived data values of the corresponding parameters.

A.1.2.2 `PSCALn` Keywords

This keyword shall be used, along with the `PZEROn` keyword, when the data set parameter values are not the true physical values, to transform the data set parameter values to the true physical values they represent, using the linear transformation equation below with the `PZEROn` keyword. The value field shall contain a floating point number representing the coefficient of the linear term in the scaling equation, the scaling factor between true values and data set parameter values at zero offset.

A.1.2.3 `PZEROn` Keywords

This keyword shall be used, along with the `PSCALn` keyword, when the data set parameter values are not the true physical values, to transform data set parameter values to the true values. The value field shall contain a floating point number, representing the true value corresponding to a data set parameter value of zero. The transformation equation is as follows:

$$\text{physical value} = \texttt{PZEROn} + \texttt{PSCALn} \times \text{data set value}$$

A.2 Data Storage

Values shall be stored as a series 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. 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 member of the array of the previous group. The arrays shall be organized internally in the same way as an ordinary primary array.

A.3 Data Representation

Permissible data representations are the same as those for an ordinary primary array. Parameters and members of associated data arrays shall have the same representation. Should more precision be required for an associated parameter than for a member of a data array, the parameter shall be divided into two 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 array values using the PSCALn and PZEROn keywords.

Appendix B

A Binary Table Extension

(This Appendix is not part of the NSDSSO *FITS* Standard but is included for informational purposes only.)

The new "3D"[1] binary tables format is under consideration by the regional and international *FITS* committees. Its present form is expected to be a subset of the eventual structure that is adopted; some structures and capabilities remain to be added. It is in use at a number of installations, has been implemented under AIPS and exchanged between AIPS and MIDAS, but much of the existing *FITS*-reading software yet cannot decode the format. Because it is becoming widely used, and because it illustrates an application of the rules for conforming extensions, it is described in this Appendix.

B.1 Identification

The interim identification of a binary table is indicated if the primary header of the *FITS* file has the keyword `EXTEND` set to `T` and the first keyword of the associated extension header has `XTENSION=␣'A3DTABLE'`.

B.2 Header

The header of a binary table extension shall conform to the requirements for *FITS* headers specified in Section 5.

B.2.1 Required Keywords

The card images in the header of a binary table extension use the following keywords in the order given in Table B.1.

[1]The term "3D" derives from the capacity to store vectors as fields in binary tables. Thus, the table effectively has depth.

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, ..., `TFIELDS`
	⋮
last	`END`

Table B.1: Required keywords in the interim binary table extension.

B.2.1.1 `XTENSION` Keyword

The value field shall contain the character string `'A3DTABLE'`.

B.2.1.2 `BITPIX` Keyword

The value field shall contain the integer 8, indicating that row size is given in 8-bit bytes.

B.2.1.3 `NAXIS` Keyword

The value field shall contain the integer 2, indicating two axes, rows and columns.

B.2.1.4 `NAXIS1` Keyword

The value field shall contain an integer, representing the width of the table in 8-bit bytes.

B.2.1.5 `NAXIS2` Keyword

The value field shall contain an integer of value 1 or greater, representing the number of rows in the table.

B.2.1.6 PCOUNT Keyword

The value field shall contain the integer 0; no data values precede the table.

B.2.1.7 GCOUNT Keyword

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

B.2.1.8 TFIELDS Keyword

The value field contains an integer representing the number of fields in each row.

B.2.1.9 TFORMn Keywords

The value field contains a character string describing the data type of field n. The following values are permitted, each representing the data type indicated:

Value field	Data Type
rL	Logical
rX	Bit array
rA	ASCII characters
rI	16-bit integer
rJ	32-bit integer
rE	32-bit floating point
rD	64-bit floating point

Table B.2: Valid TFORMn format values in the interim binary table extension.

The lower case r preceding the upper case type indicator is the repeat count, describing how many times the data type appears within field n. A repeat count of zero indicates that the entry defined in the header is omitted in the table.

B.2.1.10 END Keyword

This keyword has no associated value. Columns 9-80 shall be blank.

B.2.2 Reserved Keywords

These keywords are optional but are used in A 3-D Tables extensions only in the manner defined in this document.

B.2.2.1 `TTYPEn` Keyword

The value field contains a character string representing the label or heading for field n.

B.2.2.2 `TUNITn` Keywords

The value field contains a character string representing the physical units of the quantities contained in field n.

B.2.2.3 `TNULL` Keyword.

The value field contains an integer representing the value used in integer fields to represent an undefined value. The IEEE NaN (Not-a-Number) is used for the same purpose in floating point fields.

B.3 Storage

Data are stored as a two dimensional array. The number of bytes in a row and the number of rows are specified by the `NAXIS1` and `NAXIS2` keywords in the associated extension header. Every row in the array must have the same number of bytes. The number of bytes in a row and the number of rows in the table determines the size of the array. The first byte of the first row is at the start of the record immediately following the last header record. The first byte of every subsequent row immediately follows the last byte of the row preceding it.

B.4 Format

Each row in a binary table extension array consists of a sequence of fields, each of which consists of zero or more entries. The location, length and format of each field are the same for every row. The field sequence is specified by the `TFORMn` keywords of the associated header. All entries in a field must be of the same data type. For each field, the number of entries, data type, and (optionally) the field name, are specified in keywords of the associated header records. A separate format keyword must be provided for each field.

B.5 Data Types

Each field in the table contains a binary representation of information in one of the seven following forms:

B.5.1 Logical

A logical value consists of an 8-bit ASCII `T` for true or `F` for false.

B.5.2 Bit Arrays

A bit array consists of an integral number of bytes with trailing bits zero. A bit array shall start in the most significant bit of the first byte; bits will follow in order of decreasing significance

B.5.3 ASCII Characters

Each character shall be represented by one byte. A character shall be written in 7-bit ASCII [8], right justified in the byte. The high-order parity bit shall be zero.

B.5.4 Eight-bit Integers

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

B.5.5 Sixteen-bit Integers

Sixteen-bit integers shall be twos-complement signed binary integers, contained in two bytes. The sign bit shall be the leftmost bit of the leftmost byte. Significance shall decrease from left to right; the least significant bit shall be the rightmost bit of the rightmost byte. The `TNULL` keyword is used if needed to specify the symbol corresponding to an undefined value.

B.5.6 Thirty-two-bit Integers

Thirty-two bit integers shall be twos-complement signed binary integers, contained in four bytes. The sign bit shall be the leftmost bit of the leftmost byte. Significance shall decrease from left to right; the least significant bit shall be the rightmost bit of the rightmost byte. The `TNULL` keyword is used if needed to specify the symbol corresponding to an undefined value.

B.5.7 Thirty-two-bit Floating Point

B.5.7.1 Interpretation

Lowest numbered (leftmost) bits are most significant. The IEEE NaN (Not-a-Number) values shall be used to represent undefined values. All IEEE special values are recognized.

$$value = (-1)^{\mathrm{sign}} \times 2^{(\mathrm{exponent}-127)} \times \mathrm{mantissa}$$

B.5.7.2 Structure

Table 6.1 describes the bit structure of 32-bit floating point values.

B.5.8 Sixty-four-bit Floating Point

B.5.8.1 Interpretation

Lowest numbered (leftmost) bits are most significant. The IEEE NaN (Not-a-Number) values shall be used to represent undefined values. All IEEE special values are recognized.

$$value \; = \; (-1)^{\text{sign}} \times 2^{(\text{exponent}-1023)} \times \text{mantissa}$$

B.5.8.2 Structure

Table 6.2 describes the bit structure of 64-bit floating point values.

Appendix C

Implementation on Physical Media

(This Appendix is not part of the NSDSSO *FITS* Standard, but is included as a guide to recommended practices)

C.1 Block Size

The block size (physical record length) for transport of data should, where possible, equal the logical record length or an integer blocking factor times this record length. Standard values of the blocking factor may be specified for each medium; if not otherwise specified, the expected value is unity.

C.1.1 Magnetic Tape

For nine-track magnetic tapes conforming to the ANSI X3.40–1983 specifications [10], there should be from one to 10 logical records per physical block. The `BLOCKED` keyword (section 5.2.2.1) may be used to warn that there may be more than one logical record per physical block. The last physical block of a *FITS* file should be truncated to the minimum number of *FITS* logical records required to hold the remaining data, in accordance with ANSI X3.27–1978 specifications [11]. With the issuance of this standard the `BLOCKED` keyword is deprecated by this document.

C.1.2 Other Media

For media where the physical block size cannot be equal to or an integral multiple of the *FITS* logical record length of 23040-bits (2880 8-bit bytes), records should be written over multiple blocks. Conventions regarding the relation between physical block size and logical record length of *FITS* data sets have not been otherwise established for other media.

C.2 Physical Properties of Media

The arrangement of digital bits and other physical properties of any medium should be in conformance with the ANSI standard for that medium as specified in the relevant document.

C.3 Labeling

C.3.1 Tape

Tapes may be either ANSI standard labeled or unlabeled. Unlabeled tapes are preferred.

C.3.2 Other Media

Conventions regarding labels for *FITS* data sets have not been established for other media.

C.4 *FITS* File Boundaries

C.4.1 Magnetic Tape

Individual *FITS* files are terminated by a tape-mark.

C.4.2 Other Media

For media where the physical record size cannot be equal to or an integral multiple of the standard *FITS* logical record length, a record of fewer than 23040 bits (2880 8-bit bytes) immediately following the end of the primary header, data, or an extension should be treated as an end-of-file. Otherwise, individual *FITS* data sets should be terminated by a delimiter appropriate to the medium, analogous to the tape end-of-file mark. If more than one *FITS* data set appears on a physical structure, the appropriate end-of-file indicator should immediately precede the start of the primary headers of all data sets after the first. *FITS* data sets should start at the start of a file.

C.5 Multiple Physical Volumes

Storage of a single *FITS* file on more than one tape or on multiple units of any other medium is not supported.

Appendix D

Summary of Keywords

(This Appendix is not part of the NSDSSO *FITS* Standard, but is included for convenient reference).

Principal HDU	Conforming Extension	ASCII Table Extension	Random Groups Extension	Binary Table Extension
SIMPLE	XTENSION	XTENSION †	SIMPLE	XTENSION ‡
BITPIX	BITPIX	BITPIX = 8	BITPIX	BITPIX = 8
NAXIS	NAXIS	NAXIS = 2	NAXIS	NAXIS = 2
NAXISn	NAXISn	NAXIS1	NAXIS1 = 0	NAXIS1
EXTEND	PCOUNT	NAXIS2	NAXISn	NAXIS2
END	GCOUNT	PCOUNT = 0	GROUPS	PCOUNT = 0
	END	GCOUNT = 1	PCOUNT	GCOUNT = 1
		TFIELDS	GCOUNT	TFIELDS
		TBCOLn	END	TFORMn
		TFORMn		END
		END		

† `XTENSION=␣'TABLE␣␣␣'` for the ASCII Table extension.
‡ `XTENSION=␣'A3DTABLE'` for the interim binary table extension.

Table D.1: Mandatory *FITS* keywords for the structures described in this document.

Principal HDU		Conforming	ASCII Table	Random Groups	Binary Table
General	Array	Extension	Extension	Extension	Extension
DATE	BSCALE	EXTNAME	TSCALn	PTYPEn	TTYPEn
ORIGIN	BZERO	EXTVER	TZEROn	PSCALn	TUNITn
BLOCKED	BUNIT	EXTLEVEL	TNULLn	PZEROn	TNULL
AUTHOR	BLANK		TTYPEn		
REFERENC	CTYPEn		TUNITn		
COMMENT	CRPIXn				
HISTORY	CROTAn				
␣␣␣␣␣␣␣␣	CRVALn				
DATE-OBS	CDELTn				
TELESCOP	DATAMAX				
INSTRUME	DATAMIN				
OBSERVER					
OBJECT					
EQUINOX					
EPOCH					

Table D.2: Reserved *FITS* keywords for the structures described in this document. Note that the EPOCH and BLOCKED keywords are deprecated by this document.

Production	Bibliographic	Commentary	Observation	Array
DATE	AUTHOR	COMMENT	DATE-OBS	BSCALE
ORIGIN	REFERENC	HISTORY	TELESCOP	BZERO
BLOCKED		␣␣␣␣␣␣␣␣	INSTRUME	BUNIT
			OBSERVER	BLANK
			OBJECT	CTYPEn
			EQUINOX	CRPIXn
			EPOCH	CROTAn
				CRVALn
				CDELTn
				DATAMAX
				DATAMIN

Table D.3: General Reserved *FITS* keywords described in this document. Note that the EPOCH and BLOCKED keywords are deprecated by this document.

Appendix E

NSDSSO Publications

Document	Title	Date	Status
NSDSSO 100-0.1	FITS Standard	December, 1990	Draft Standard

Table E.1: NSDSSO Publications