4.4.2.7 Table Keywords

These keywords are used to describe the contents of ASCII table extensions (Sect. 7.2.2) or binary table extensions (Sect.7.3.2). They are *optional*, but if they appear in the header describing an ASCII or binary table, 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 defined here.

The following 4 keywords *may* be used to describe the actual or allowed minimum and maximum values in numerical columns of a *FITS* ASCII or binary table. These keywords *must* have the same data type as the physical values in the associated column (either an integer or a floating point number). Any undefined elements in the column (or any other IEEE special values in the case of floating point columns in binary tables) *shall* be excluded when determining the value of these keywords.

`TDMINn` Keyword. The value field *shall* contain a number giving the minimum physical value actually contained in column `n` of the table. This keyword is analogous to the `DATAMIN` keyword that is defined for arrays in Sect. 4.4.2.5.

`TDMAXn` Keyword. The value field *shall* contain a number giving the maximum physical value actually contained in column `n` of the table. This keyword is analogous to the `DATAMAX` keyword that is defined for arrays in Sect. 4.4.2.5.

`TLMINn` Keyword. The value field *shall* contain a number giving the minimum legally defined physical value that may be contained in column `n` of the table.

`TLMAXn` Keyword. The value field *shall* contain a number giving the maximum legally defined physical value that may be contained in column `n` of the table.

If the value of `TDMINn` is greater than `TDMAXn`, or `TLMINn` is greater than `TLMAXn`, then the values of the pair of keywords *should* be interpreted as undefined. It is permissible to have values in a column that are less than `TLMINn` or greater than `TLMAXn`, however, the interpretation of any such out-of-range column elements is not defined.

The `TLMINn` and `TLMAXn` keywords are commonly used when constructing histograms of the data values in a column. This is particularly important when images or spectra are stored in "event list" format which tabulates the properties of each detected event. For example, X-ray missions may record images in a table that lists of the X and Y pixel location of each detected X-ray photon. Since such images may be quite sparse and many of the pixels may not contain any recorded events, the `TLMINn` and `TLMAXn` keywords provide information about the range of coordinates that could have been detected.

4.4.3. Additional keywords

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. Any keyword that refers to or depends upon the existence of other specific HDUs in the same or other files should be used with caution because the persistence of those HDUs cannot always be guaranteed.

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 space filled to the width of the field.

TDISPn keywords. The value field of this indexed keyword *shall* contain a character string describing the format recommended for displaying an ASCII text representation of of the contents of field n. This keyword overrides the default display format given by the TFORMn keyword. 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. Only the format codes in Table 16, interpreted as Fortran (ISO 2004) output formats, and discussed in more detail in Sect. 7.3.4, 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.

min and max keywords. To describe the actual or allowed minimum and maximum values in numerical columns of an ASCII table one *shall* use the TDMINn, TDMAXn, TLMINn, TLMAXn keywords defined in section 4.4.2.7.

7.2.3. 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. 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 data block immediately following the last header block. The first character of subsequent rows *shall* follow immediately the character at the end of the previous row, independent of the *FITS* block structure. The positions in the last data block after the last character of the last row of the table *shall* be filled with ASCII spaces.

7.2.4. Fields

Each row in the array *shall* consist of a sequence of from 0 to 999 fields, as specified by the TFIELDS keyword, with one entry in each field. For every field, the Fortran (ISO 2004) format of the information contained (given by the TFORMn keyword), the location in the row of the beginning of the field (given by the TBCOLn keyword), and (*optionally*, but *strongly recommended*) the field name (given by the TTYPEn keyword), *shall* be specified in the associated header. The location and format of fields *shall* be the same for every row. Fields *may* overlap, but this usage is *not recommended*. Only a limited set of ASCII character values *may* appear within any field, depending on the field type as specified below. There *may* be characters in a table row that are not included in any field, (e.g., between fields, or before the first field or after the last field). Any 7-bit ASCII character *may* occur in characters of a table row that are not included in a defined field. A common convention is to include a space character between each field for added legibility if the table row is displayed verbatim. It is also permissible to add control characters, such as a carriage return or line feed character, following the last field in each row as a way of formatting the table if it is printed or displayed by a text editing program.

7.2.5. 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 Fortran (ISO 2004) format, as described below. The only possible formats *shall* be those specified in Table 15. If values of −0 and +0 need to 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 for each field that is appropriate to the form, range of values, and accuracy of the data in that field. This standard does not impose an upper limit on the number of digits of precision, nor any limit on the range of numeric values. Software packages that read or write data according to this standard could be limited, however, in the range of values and exponents that are supported (e.g., to the range that can be represented by 32-bit or 64-bit binary numbers).

The value of each entry *shall* be interpreted as described in the following paragraphs.

Character fields. 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 character string *shall* be composed of the restricted set of ASCII text characters with decimal values in the range 32 through 126 (hexadecimal 20 through 7E).

Integer fields. The value of an integer-formatted (Iw) field is a signed decimal integer contained in columns TBCOLn through TBCOLn+w − 1 consisting of a single *optional* sign (‘+’ or ‘-’) followed by one or more decimal digits (‘0’ through ‘9’). Non-significant space characters may precede and/or follow the integer value within the field. A blank field has value 0. All characters other than leading and trailing spaces, a contiguous string of decimal digits, and a single leading sign character are forbidden.

Real fields. 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 any trailing space characters in the field and right-justifying the remaining characters,
2. interpreting the first non-space 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

Table 19: Usage of TZEROn to represent non-default integer data types.

TFORMn	Native data type	Physical data type	TZEROn	
B	unsigned	signed byte	-128	(-2^7)
I	signed	unsigned 16-bit	32768	(2^{15})
J	signed	unsigned 32-bit	2147483648	(2^{31})
K	signed	unsigned 64-bit	9223372036854775808	(2^{63})

offset specified by the TZEROn keyword to the native data type value that is stored in the table field.

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, J or K, or P or Q array descriptor fields (Sect. 7.3.5) that point to B, I, J or K integer arrays. The keyword *must not* be used if field n is of any other data type. The value of this keyword corresponds to the table column values before applying any transformation indicated by the TSCALn and TZEROn keywords.

If the TSCALn and TZEROn keywords do not have the default values of 1.0 and 0.0, respectively, then the value of the TNULLn keyword *must* equal the actual value in the *FITS* file that is used to represent an undefined element and not the corresponding physical value (computed from Eq. 7). To cite a specific, common example, *unsigned* 16-bit integers are represented in a *signed* integer column (with TFORMn = 'I') by setting TZEROn = 32768 and TSCALn = 1. If it is desired to use elements that have an *unsigned* value (i.e., the physical value) equal to 0 to represent undefined elements in the field, then the TNULLn keyword *must* be set to the value -32768 because that is the actual value stored in the *FITS* file for those elements in the field.

TDISPn keywords. The value field of this indexed keyword *shall* contain a character string describing the format recommended for displaying an ASCII text representation 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 as an unsigned integer. Arrays of type A *may* be terminated with a zero byte. Only the format codes in Table 20, interpreted as Fortran (ISO 2004) output formats, and discussed in more detail in Sect. 7.3.4, 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. In the case of fields of type P or Q, the TDISPn value applies to the data array pointed to by the array descriptor (Sect. 7.3.5), not the values in the array descriptor itself.

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, which is also the minimum allowed 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. The use of this keyword is described in in Sect. 7.3.5.

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 multi-dimensional array with a format of '(*l*,*m*,*n*...)' where *l*, *m*, *n*, ... are the dimensions of the array. The data are ordered such that the array index of the first dimension given (*l*) is the most rapidly varying and that of the last dimension given is the least rapidly varying. The total number of elements in the array equals the product of the dimensions specified in the TDIMn keyword. The size *must* be less than or equal to the repeat count on the TFORMn keyword, or, in the case of columns that have a 'P' or 'Q' TFORMn data type, less than or equal to the array length specified in the variable-length array descriptor (see Sect. 7.3.5). In the special case where the variable-length array descriptor has a size of zero, then the TDIMn keyword is not applicable. If the number of elements in the array implied by the TDIMn is less than the allocated size of the array in the *FITS* file, then the unused trailing elements *should* be interpreted as containing undefined fill values.

A character string is represented in a binary table by a one-dimensional character array, as described under 'Character' in the list of data types in Sect. 7.3.3. For example, a Fortran CHARACTER*20 variable could be represented in a binary table as a character array declared as TFORMn = '20A'. Arrays of strings, i.e., multi-dimensional character arrays, *may* be represented using the TDIMn notation. For example, if TFORMn = '60A' and TDIMn = '(5,4,3)', then the entry consists of a 4×3 array of strings of five characters each.

min and max keywords. To describe the actual or allowed minimum and maximum values in numerical columns of a binary table one *shall* use the TDMINn, TDMAXn, TLMINn, TLMAXn keywords defined in section 4.4.2.7.

7.3.3. 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 in the supplemental data area. The positions in the last data block after the last additional byte, or, if there are no additional bytes, the last character of the last row of the main data table, *shall* be filled by setting all bits to zero.

7.3.3.1. 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. 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

Appendix C: Summary of keywords

This Appendix is not part of the FITS *standard, but is included for convenient reference.*

All of the mandatory and reserved keywords that are defined in the standard, except for the reserved WCS keywords that are discussed separately in Sect. 8, are listed in Tables C.1, C.2, and C.3. An alphabetized list of these keywords and their definitions is available online: `http://heasarc.gsfc.nasa.gov/docs/fcg/standard_dict.html`.

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

Primary HDU	Conforming extension	Image extension	ASCII table extension	Binary table extension	Random groups records
SIMPLE	XTENSION	XTENSION[1]	XTENSION[2]	XTENSION[3]	SIMPLE
BITPIX	BITPIX	BITPIX	BITPIX = 8	BITPIX = 8	BITPIX
NAXIS	NAXIS	NAXIS	NAXIS = 2	NAXIS = 2	NAXIS
NAXISn[4]	NAXISn[4]	NAXISn[4]	NAXIS1	NAXIS1	NAXIS1 = 0
END	PCOUNT	PCOUNT = 0	NAXIS2	NAXIS2	NAXISn[4]
	GCOUNT	GCOUNT = 1	PCOUNT = 0	PCOUNT	GROUPS = T
	END	END	GCOUNT = 1	GCOUNT = 1	PCOUNT
			TFIELDS	TFIELDS	GCOUNT
			TFORMn[5]	TFORMn[5]	END
			TBCOLn[5]	END	
			END		

[1]`XTENSION=␣'IMAGE␣␣␣'` for the image extension. [2]`XTENSION=␣'TABLE␣␣␣'` for the ASCII table extension. [3]`XTENSION=␣'BINTABLE'` for the binary table extension. [4]Runs from 1 through the value of `NAXIS`. [5]Runs from 1 through the value of `TFIELDS`.

Table C.2: Reserved *FITS* keywords for the structures described in this document.

All[1] HDUs		Array[2] HDUs	ASCII table extension	Binary table extension	Random groups records
DATE	EXTNAME	BSCALE	TSCALn	TSCALn	PTYPEn
DATE-OBS	EXTVER	BZERO	TZEROn	TZEROn	PSCALn
ORIGIN	EXTLEVEL	BUNIT	TNULLn	TNULLn	PZEROn
AUTHOR	EQUINOX	BLANK	TTYPEn	TTYPEn	
REFERENC	EPOCH[3]	DATAMAX	TUNITn	TUNITn	
COMMENT	BLOCKED[3]	DATAMIN	TDISPn	TDISPn	
HISTORY	EXTEND[4]		TDMAXn	TDIMn	
␣␣␣␣␣␣␣␣	TELESCOP		TDMINn	THEAP	
OBJECT	INSTRUME		TLMAXn	TDMAXn	
OBSERVER			TLMINn	TDMINn	
				TLMAXn	
				TLMINn	

[1]These keywords are further categorized in Table C.3. [2]Primary HDU, image extension, user-defined HDUs with same array structure. [3]Deprecated. [4]Only permitted in the primary HDU.

Table C.3: General reserved *FITS* keywords described in this document.

Production	Bibliographic	Commentary	Observation
DATE	AUTHOR	COMMENT	DATE-OBS
ORIGIN	REFERENC	HISTORY	TELESCOP
BLOCKED[1]		␣␣␣␣␣␣␣␣	INSTRUME
			OBSERVER
			OBJECT
			EQUINOX
			EPOCH[1]

[1]Deprecated.

H.3 List of modifications to the latest FITS standard

The table keywords described in Sect. 4.4.2.7 were originally introduced as a *FITS* convention since 1993, and registered in 2006. The text of the original convention is reported at `http://fits.gsfc.nasa.gov/registry/colminmax.html`. The differences with this standard concern:

- The exclusion of undefined or IEEE special values when computing maximum and minimum is now *mandatory* while it was *optional*.
- The original text contained usage examples and additional minor explanatory details.