Table 16: Valid `TDISPn` format values in `TABLE` extensions.

Field value	Data type
`Aw`	Character
`Iw.m`	Integer
`Bw.m`	Binary, integers only
`Ow.m`	Octal, integers only
`Zw.m`	Hexadecimal, integers only
`Fw.d`	Floating-point, fixed decimal notation
`Ew.dEe`	Floating-point, exponential notation
`ENw.d`	Engineering; E format with exponent multiple of three
`ESw.d`	Scientific; same as EN but non-zero leading digit if not zero
`Gw.dEe`	General; appears as F if significance not lost, else E.
`Dw.dEe`	Floating-point, exponential notation

Notes. `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*.

erators). String comparisons with the `TTYPEn` keyword values *should not* be case sensitive (e.g., 'TIME' and 'Time' *should* be interpreted as the same name).

`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. 4.3.

`TSCALn` keywords. This indexed keyword *shall* be used, along with the `TZEROn` keyword, to linearly scale the values in the table field `n` to transform them into the physical values that they represent using Eq. 7. The value field *shall* contain a floating-point number representing the coefficient of the linear term in the scaling equation. The default value for this keyword is 1.0. This keyword *must not* be used for A-format fields.

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

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

where `field_value` is the value that is actually stored in that table field in the *FITS* file.

`TZEROn` keywords. This indexed keyword *shall* be used, along with the `TSCALn` keyword, to linearly scale the values in the table field `n` to transform them into the physical values that they represent using Eq. 7. 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. This keyword *must not* be used for A-format fields.

`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.

The following 4 keywords *may* be used to specify 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 *shall* be excluded when determining the value of these keywords.

`TDMINn` Keyword. The value field *shall* contain a number giving the minimum physical value 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 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 that specifies the minimum physical value in column `n` that has a valid meaning or interpretation. The column is not required to actually contain any elements that have this value, and the column may contain elements with physical values less than `TLMINn`, however, the interpretation of any such out-of-range column elements is not defined.

`TLMAXn` Keyword. The value field *shall* contain a number that specifies the maximum physical value in column `n` that has a valid meaning or interpretation. The column is not required to actually contain any elements that have this value, and the column may contain elements with physical values 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. For example, if a table contains columns that give the X and Y pixel location of a list of photons that were detected by a photon counting device, then the `TLMINn` and `TLMAXn` keywords could be used to specify the minimum and maximum values that the detector is capable of assigning to the X and Y columns.

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 *d* digits of the string, with leading zeros assumed if necessary. The use of implicit decimal points is *deprecated* and is strongly discouraged because of the possibility that *FITS* reading programs will misinterpret the data value. Therefore, real-formatted fields *should* always contain an explicit decimal point.
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, including embedded space characters, 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

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})

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.

The following 4 keywords *may* be used to specify 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 *shall* be excluded when determining the value of these keywords.

TDMINn Keyword. The value field *shall* contain a number giving the minimum physical value 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 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 that specifies the minimum physical value in column n that has a valid meaning or interpretation. The column is not required to actually contain any elements that have this value, and the column may contain elements with physical values less than TLMINn, however, the interpretation of any such out-of-range column elements is not defined.

TLMAXn Keyword. The value field *shall* contain a number that specifies the maximum physical value in column n that has a valid meaning or interpretation. The column is not required to actually contain any elements that have this value, and the column may contain elements with physical values 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. For example, if a table contains columns that give the X and Y pixel location of a list of photons that were detected by a photon counting device, then the TLMINn and TLMAXn keywords could be used to specify the minimum and maximum values that the detector is capable of assigning to the X and Y columns.

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,