4.2.2. Logical

If the value is a fixed-format logical constant, it *shall* appear as an uppercase `T` or `F` in byte 30. A logical value is represented in free-format by a single character consisting of an uppercase `T` or `F` as the first non-space character in bytes 11 through 80.

4.2.3. Integer number

If the value is a fixed-format integer, the ASCII representation *shall* be right-justified in bytes 11 through 30. An integer consists of a '+' (decimal 43 or hexadecimal 2B) or '−' (decimal 45 or hexadecimal 2D) sign, followed by one or more contiguous ASCII digits (decimal 48 to 57 or hexadecimal 30 to 39), with no embedded spaces. The leading '+' sign is *optional*. Leading zeros are permitted, but are not significant. The integer representation *shall* always be interpreted as a signed, decimal number. This standard does not limit the range of an integer keyword value, however, software packages that read or write data according to this standard could be limited in the range of values that are supported (e.g., to the range that can be represented by a 32-bit or 64-bit signed binary integer).

A free-format integer value follows the same rules as fixed-format integers except that the ASCII representation *may* occur anywhere within bytes 11 through 80.

4.2.4. Real floating-point number

If the value is a fixed-format real floating-point number, the ASCII representation *shall* be right-justified in bytes 11 through 30.

A floating-point number is represented by a decimal number followed by an *optional* exponent, with no embedded spaces. A decimal number *shall* consist of a '+' (decimal 43 or hexadecimal 2B) or '–' (decimal 45 or hexadecimal 2D) sign, followed by a sequence of ASCII digits containing a single decimal point ('.'), representing an integer part and a fractional part of the floating-point number. The leading '+' sign is *optional*. At least one of the integer part or fractional part *must* be present. If the fractional part is present, the decimal point *must* also be present. If only the integer part is present, the decimal point *may* be omitted, in which case the floating-point number is indistinguishable from an integer. The exponent, if present, consists of an exponent letter followed by an integer. Letters in the exponential form ('E' or 'D')[2] *shall* be upper case. The full precision of 64-bit values cannot be expressed over the whole range of values using the fixed-format. This standard does not impose an upper limit on the number of digits of precision, nor any limit on the range of floating-point keyword 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 a 32-bit or 64-bit floating-point number).

A free-format floating-point value follows the same rules as a fixed-format floating-point value except that the ASCII representation *may* occur anywhere within bytes 11 through 80.

[2] The 'D' exponent form is traditionally used when representing values that have more decimals of precision or a larger magnitude than can be represented by a single precision 32-bit floating-point number, but otherwise there is no distinction between 'E' or 'D'.

Table 3: IAU-recommended basic units.

Quantity	Unit	Meaning	Notes
SI base & supplementary units			
length	`m`	meter	
mass	`kg`	kilogram	`g` gram allowed
time	`s`	second	
plane angle	`rad`	radian	
solid angle	`sr`	steradian	
temperature	`K`	kelvin	
electric current	`A`	ampere	
amount of substance	`mol`	mole	
luminous intensity	`cd`	candela	
IAU-recognized derived units			
frequency	`Hz`	hertz	s^{-1}
energy	`J`	joule	N m
power	`W`	watt	$J\ s^{-1}$
electric potential	`V`	volt	$J\ C^{-1}$
force	`N`	newton	$kg\ m\ s^{-2}$
pressure, stress	`Pa`	pascal	$N\ m^{-2}$
electric charge	`C`	coulomb	A s
electric resistance	`Ohm`	ohm	$V\ A^{-1}$
electric conductance	`S`	siemens	$A\ V^{-1}$
electric capacitance	`F`	farad	$C\ V^{-1}$
magnetic flux	`Wb`	weber	V s
magnetic flux density	`T`	tesla	$Wb\ m^{-2}$
inductance	`H`	henry	$Wb\ A^{-1}$
luminous flux	`lm`	lumen	cd sr
illuminance	`lx`	lux	$lm\ m^{-2}$

4.2.5. Complex integer number

There is no fixed-format for complex integer numbers.[3]

If the value is a complex integer number, the value *must* be represented as a real part and an imaginary part, separated by a comma and enclosed in parentheses e.g., `(123, 45)`. Spaces *may* precede and follow the real and imaginary parts. The real and imaginary parts are represented in the same way as integers (Sect. 4.2.3). Such a representation is regarded as a single value for the complex integer number. This representation *may* be located anywhere within bytes 11 through 80.

4.2.6. Complex floating-point number

There is no fixed-format for complex floating-point numbers.[3]

If the value is a complex floating-point number, the value *must* be represented as a real part and an imaginary part, separated by a comma and enclosed in parentheses, e.g., `(123.23, -45.7)`. Spaces *may* precede and follow the real and imaginary parts. The real and imaginary parts are represented in the same way as floating-point values (Sect. 4.2.4). Such a representation is regarded as a single value for the complex floating-point number. This representation *may* be located anywhere within bytes 11 through 80.

4.2.7. **Date**

There is strictly no such thing as a data type for *date valued* keywords, however a pseudo data type of *datetime* is defined in

[3] This requirement differs from the wording in the original *FITS* papers. See Appendix H.

Sect. 9.1.2 and *must* be used to write ISO-8601 *datetime* strings as character strings.

If a keyword need to express a *time* in JD or MJD (see Sect. 9), this can be formatted as an arbitrary precision number, eventually separating the integer and fractional part as specified in Sect. 9.2.2.

4.3. Units

When a numerical keyword value represents a physical quantity, it is *recommended* that units be provided. Units *shall be* represented with a string of characters composed of the restricted ASCII text character set. Unit strings can be used as values of keywords (e.g., for the reserved keywords `BUNIT`, and `TUNITn`), as an entry in a character string column of an ASCII or binary table extension, or as part of a keyword comment string (see Sect. 4.3.2, below).

The units of all *FITS* header keyword values, with the exception of measurements of angles, *should* conform with the recommendations in the IAU Style Manual (McNally 1988). For angular measurements given as floating-point values and specified with reserved keywords, the units *should* be degrees (i.e., `deg`). If a requirement exists within this standard for the units of a keyword, then those units *must* be used.

The units for fundamental physical quantities recommended by the IAU are given in Table 3, and additional units that are commonly used in astronomy are given in Table 4. Further specifications for time units are given in Sect. 9.3. The recommended plain text form for the IAU-recognized *base units* are given in column 2 of both tables.[4] All base units strings *may* be preceded, with no intervening spaces, by a single character (two for deca) taken from Table 5 and representing scale factors mostly in steps of 10^3. Compound prefixes (e.g., `ZYeV` for 10^{45} eV) *must not* be used.

4.3.1. Construction of units strings

Compound units strings *may* be formed by combining strings of base units (including prefixes, if any) with the recommended syntax described in Table 6. Two or more base units strings (called `str1` and `str2` in Table 6) *may* be combined using the restricted set of (explicit or implicit) operators that provide for multiplication, division, exponentiation, raising arguments to powers, or taking the logarithm or square-root of an argument. Note that functions such as `log` actually require dimensionless arguments, so that `log(Hz)`, for example, actually means `log(`*x*`/1 Hz)`. The final units string is the compound string, or a compound of compounds, preceded by an *optional* numeric multiplier of the form `10**`*k*, `10^`*k*, or `10`±*k* where *k* is an integer, *optionally* surrounded by parentheses with the sign character required in the third form in the absence of parentheses. Creators of *FITS* files are encouraged to use the numeric multiplier only when the available standard scale factors of Table 5 will not suffice. Parentheses are used for symbol grouping and are strongly *recommended* whenever the order of operations might be subject to misinterpretation. A space character implies multiplication which can also be conveyed explicitly with an asterisk or a period. Therefore, although spaces are allowed as symbol separators, their use is discouraged. Note that, per IAU convention, case is significant throughout. The IAU style manual forbids the use of more than one slash (‘`/`’) character in a units string. However, since normal mathematical precedence rules apply in this context, more than one slash *may* be used but is discouraged.

Table 5: Prefixes for multiples and submultiples.

Submult	Prefix	Char	Mult	Prefix	Char
10^{-1}	deci	`d`	10	deca	`da`
10^{-2}	centi	`c`	10^{2}	hecto	`h`
10^{-3}	milli	`m`	10^{3}	kilo	`k`
10^{-6}	micro	`u`	10^{6}	mega	`M`
10^{-9}	nano	`n`	10^{9}	giga	`G`
10^{-12}	pico	`p`	10^{12}	tera	`T`
10^{-15}	femto	`f`	10^{15}	peta	`P`
10^{-18}	atto	`a`	10^{18}	exa	`E`
10^{-21}	zepto	`z`	10^{21}	zetta	`Z`
10^{-24}	yocto	`y`	10^{24}	yotta	`Y`

A unit raised to a power is indicated by the unit string followed, with no intervening spaces, by the *optional* symbols `**` or `^` followed by the power given as a numeric expression, called `expr` in Table 6. The power *may* be a simple integer, with or without sign, *optionally* surrounded by parentheses. It *may* also be a decimal number (e.g., 1.5, 0.5) or a ratio of two integers (e.g., 7/9), with or without sign, which *must* be surrounded by parentheses. Thus meters squared *may* be indicated by `m**(2)`, `m**+2`, `m+2`, `m2`, `m^2`, `m^(+2)`, etc. and per meter cubed *may* be indicated by `m**-3`, `m-3`, `m^(-3)`, `/m3`, and so forth. Meters to the three-halves power *may* be indicated by `m(1.5)`, `m^(1.5)`, `m**(1.5)`, `m(3/2)`, `m**(3/2)`, and `m^(3/2)`, but *not* by `m^3/2` or `m1.5`.

4.3.2. Units in comment fields

If the units of the keyword value are specified in the comment of the header keyword, it is *recommended* that the units string be enclosed in square brackets (i.e., enclosed by ‘`[`’ and ‘`]`’) at the beginning of the comment field, separated from the slash (‘`/`’) comment field delimiter by a single space character. An example, using a non-standard keyword, is

```
EXPTIME = 1200. / [s] exposure time in seconds
```

This widespread, but *optional*, practice suggests that square brackets *should* be used in comment fields only for this purpose. Nonetheless, software *should not* depend on units being expressed in this fashion within a keyword comment, and software *should not* depend on any string within square brackets in a comment field containing a proper units string.

[4] These tables are reproduced from the first in a series of papers on world coordinate systems (Greisen & Calabretta 2002), which provides examples and expanded discussion.

Table 9: Example of a primary array header.

Keyword records		
SIMPLE	=	T / file does conform to FITS standard
BITPIX	=	16 / number of bits per data pixel
NAXIS	=	2 / number of data axes
NAXIS1	=	250 / length of data axis 1
NAXIS2	=	300 / length of data axis 2
OBJECT	=	'Cygnus X-1'
DATE	=	'2006-10-22'
END		

The total number of bits in the extension data array (exclusive of fill that is needed after the data to complete the last 2880-byte data block) is given by the following expression:

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

where $N_{\rm bits}$ *must* be non-negative and 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. If $N_{\rm bits} > 0$, then the data array *shall* be contained in an integral number of 2880-byte *FITS* data blocks. The header of the next *FITS* extension in the file, if any, *shall* start with the first *FITS* block following the data block that contains the last bit of the current extension data array.

4.4.2. Other reserved keywords

The reserved keywords described below are *optional*, but if present in the header they *must* be used only as defined in this standard. They apply to any *FITS* structure with the meanings and restrictions defined below. Any *FITS* structure *may* further restrict the use of these keywords.

4.4.2.1. General descriptive keywords

DATE keyword. The value field *shall* contain a character string giving the date on which the HDU was created, in the form YYYY-MM-DD, or the date and time when the HDU was created, in the form YYYY-MM-DDThh:mm:ss[.sss...], where YYYY *shall* be the four-digit calendar year number, MM the two-digit month number with January given by 01 and December by 12, and DD the two-digit day of the month. When both date and time are given, the literal T *shall* separate the date and time, hh *shall* be the two-digit hour in the day, mm the two-digit number of minutes after the hour, and ss[.sss...] the number of seconds (two digits followed by an *optional* fraction) after the minute. Default values *must not* be given to any portion of the date/time string, and leading zeros *must not* be omitted. The decimal part of the seconds field is *optional* and *may* be arbitrarily long, so long as it is consistent with the rules for value formats of Sect. 4.2. Otherwise said, the format for DATE keywords written after January 1, 2000 *shall* be the ISO-8601 *datetime* form described in Sect. 9.1.2. See also Sect. 9.5.

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

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

When a newly created HDU is substantially a verbatim copy of another HDU, the value of the DATE keyword in the original HDU *may* be retained in the new HDU instead of updating the value to the current date and time.

Table 10: Mandatory keywords in conforming extensions.

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

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

EXTEND keyword. The value field *shall* contain a logical value indicating whether the *FITS* file is allowed to contain conforming extensions following the primary HDU. This keyword *may* only appear in the primary header and *must not* appear in an extension header. If the value field is T then there *may* be conforming extensions in the *FITS* file following the primary HDU. This keyword is only advisory, so its presence with a value T does not require that the *FITS* file contains extensions, nor does the absence of this keyword necessarily imply that the file does not contain extensions. Earlier versions of this standard stated that the EXTEND keyword *must* be present in the primary header if the file contained extensions, but this is no longer required.

BLOCKED keyword. This keyword is deprecated and *should not* be used in new *FITS* files. It is reserved primarily to prevent its use with other meanings. As previously defined, this keyword, if used, was *required* to appear only within the first 36 keywords in the primary header. Its presence with the required logical value of T advised that the physical block size of the *FITS* file on which

it appears *may* be an integral multiple of the *FITS* block length and not necessarily equal to it.

4.4.2.2. Keywords describing observations

`DATE-OBS` keyword. The format of the value field for `DATE-OBS` keywords *shall* follow the prescriptions for the `DATE` keyword (Sect. 4.4.2 and Sect. 9.1.2). Either the four-digit year format or the two-digit year format *may* be used for observation dates from 1900 through 1999 although the four-digit format is *recommended*.

When the format with a four-digit year is used, the default interpretations for time *should* be UTC for dates beginning 1972-01-01 and UT before. Other date and time scales are permissible. The value of the `DATE-OBS` keyword *shall* be expressed in the principal time system or time scale of the HDU to which it belongs; if there is any chance of ambiguity, the choice *should* be clarified in comments. The value of `DATE-OBS` *shall* be assumed to refer to the start of an observation, unless another interpretation is clearly explained in the comment field. Explicit specification of the time scale is *recommended*. By default, times for TAI and times that run simultaneously with TAI, e.,g., UTC and TT, will be assumed to be as measured at the detector (or, in practical cases, at the observatory). For coordinate times such as TCG, TCB and TDB, the default *shall* be to include light-time corrections to the associated spatial origin, namely the geocenter for TCG and the solar-system barycenter for the other two. Conventions *may* be developed that use other time systems. Time scales are now discussed in detail in Sect. 9.2.1 and Table 30.

When the value of `DATE-OBS` is expressed in the two-digit year form, allowed for files written before January 1, 2000 with a year in the range 1900-1999, there is no default assumption as to whether it refers to the start, middle or end of an observation.

`DATExxxx` keywords. The value fields for all keywords beginning with the string `DATE` whose value contains date, and *optionally* time, information *shall* follow the prescriptions for the `DATE-OBS` keyword. See also Sect. 9.1.2 for the *datetime* format, and Sect. 9.5 for further global time keywords specified by the Standard.

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

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

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

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

4.4.2.3. Bibliographic keywords

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

`REFERENC` keyword. The value field *shall* contain a character string citing a reference where the data associated with the header are published. It is *recommended* that either the 19-digit bibliographic identifier[7] used in the Astrophysics Data System bibliographic databases (`http://adswww.harvard.edu/`) or the Digital Object Identifier (`http://doi.org`) be included in the value string when available (e.g., `'1994A&AS..103..135A'` or `'doi:10.1006/jmbi.1998.2354'`).

[7] This bibliographic convention (Schmitz et al. 1995) was initially developed for use within NED (NASA/IPAC Extragalactic Database) and SIMBAD (operated at CDS, Strasbourg, France).

5. Data Representation

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

5.1. Characters

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

5.2. Integers

5.2.1. Eight-bit

Eight-bit integers *shall* be unsigned binary integers, contained in one byte with decimal values ranging from 0 to 255.

5.2.2. Sixteen-bit

Sixteen-bit integers *shall* be two's complement signed binary integers, contained in two bytes with decimal values ranging from -32768 to +32767.

5.2.3. Thirty-two-bit

Thirty-two-bit integers *shall* be two's complement signed binary integers, contained in four bytes with decimal values ranging from -2147483648 to +2147483647.

5.2.4. Sixty-four-bit

Sixty-four-bit integers *shall* be two's complement signed binary integers, contained in eight bytes with decimal values ranging from -9223372036854775808 to +9223372036854775807.

5.2.5. Unsigned integers

The *FITS* format does not support a native unsigned integer data type (except for the unsigned 8-bit byte data type) therefore unsigned 16-bit, 32-bit, or 64-bit binary integers cannot be stored directly in a *FITS* data array. Instead, the appropriate offset *must* be applied to the unsigned integer to shift the value into the range of the corresponding signed integer, which is then stored in the *FITS* file. The `BZERO` keyword *shall* record the amount of the offset needed to restore the original unsigned value. The `BSCALE` keyword *shall* have the default value of 1.0 in this case, and the appropriate `BZERO` value, as a function of `BITPIX`, is specified in Table 11.

This same technique *must* be used when storing unsigned integers in a binary table column of signed integers (Sect. 7.3.2). In this case the `TSCALn` keyword (analogous to `BSCALE`) *shall* have the default value of 1.0, and the appropriate `TZEROn` value (analogous to `BZERO`) is specified in Table 19.

5.3. IEEE-754 floating point

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

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

Appendix E has additional details on the IEEE format.

5.4. **Time**

There is strictly no such thing as a data type for *time valued* data, but rules to encode time values are given in Sect. 9 and in more details in Rots et al. (2015).

6. Random groups structure

The random groups structure allows a collection of 'groups', where a group consists of a subarray of data and a set of associated parameter values, to be stored within the *FITS* primary data array. Random groups have been used almost exclusively for applications in radio interferometry; outside this field, there is little support for reading or writing data in this format. Other than the existing use for radio interferometry data, the random groups structure is deprecated and *should not* be further used. For other applications, the binary table extension (Sect. 7.3) provides a more extensible and better documented way of associating groups of data within a single data structure.

6.1. Keywords

6.1.1. Mandatory keywords

The `SIMPLE` keyword is required to be the first keyword in the primary header of all *FITS* files, including those with random groups records. If the random groups format records follow the primary header, the keyword records of the primary header *must* use the keywords defined in Table 12 in the order specified. No other keywords *may* intervene between the `SIMPLE` keyword and the last `NAXISn` keyword.

`SIMPLE` keyword. The keyword record containing this keyword is structured in the same way as if a primary data array were present (Sect. 4.4.1).

`BITPIX` keyword. The keyword record containing this keyword is structured as prescribed in Sect. 4.4.1.

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

7.3.6. Variable-length-array guidelines

While the above description is sufficient to define the required features of the variable-length array implementation, some hints regarding usage of the variable-length array facility might also be useful.

Programs that read binary tables should take care to not assume more about the physical layout of the table than is required by the specification. For example, there are no requirements on the alignment of data within the heap. If efficient runtime access is a concern one might want to design the table so that data arrays are aligned to the size of an array element. In another case one might want to minimize storage and forgo any efforts at alignment (by careful design it is often possible to achieve both goals). Variable-length array data *may* be stored in the heap in any order, i.e., the data for row $N+1$ are not necessarily stored at a larger offset than that for row N. There *may* be gaps in the heap where no data are stored. Pointer aliasing is permitted, i.e., the array descriptors for two or more arrays *may* point to the same storage location (this could be used to save storage if two or more arrays are identical).

Byte arrays are a special case because they can be used to store a 'typeless' data sequence. Since *FITS* is a machine-independent storage format, some form of machine-specific data conversion (byte swapping, floating-point format conversion) is implied when accessing stored data with types such as integer and floating, but byte arrays are copied to and from external storage without any form of conversion.

An important feature of variable-length arrays is that it is possible that the stored array length *may* be zero. This makes it possible to have a column of the table for which, typically, no data are present in each stored row. When data are present the stored array can be as large as necessary. This can be useful when storing complex objects as rows in a table.

Accessing a binary table stored on a random access storage medium is straightforward. Since the rows of data in the main data table are fixed in size they can be randomly accessed given the row number, by computing the offset. Once the row has been read in, any variable-length array data can be directly accessed using the element count and offset given by the array descriptor stored in that row.

Reading a binary table stored on a sequential access storage medium requires that a table of array descriptors be built up as the main data table rows are read in. Once all the table rows have been read, the array descriptors are sorted by the offset of the array data in the heap. As the heap data are read, arrays are extracted sequentially from the heap and stored in the affected rows using the back pointers to the row and field from the table of array descriptors. Since array aliasing is permitted, it might be necessary to store a given array in more than one field or row.

Variable-length arrays are more complicated than regular static arrays and might not be supported by some software systems. The producers of *FITS* data products should consider the capabilities of the likely recipients of their files when deciding whether or not to use this format, and as a general rule should use it only in cases where it provides significant advantages over the simpler fixed-length array format. In particular, the use of variable-length arrays might present difficulties for applications that ingest the *FITS* file via a sequential input stream because the application cannot fully process any rows in the table until after the entire fixed-length table and potentially the entire heap has been transmitted as outlined in the previous paragraph.

8. World coordinate systems

Representations of the mapping between image coordinates and physical (i.e., world) coordinate systems (WCSs) may be represented within *FITS* HDUs. The keywords that are used to express these mappings are now rigorously defined in a series of papers on world coordinate systems (Greisen & Calabretta 2002), celestial coordinate systems (Calabretta & Greisen 2002), spectral coordinate systems (Greisen et al. 2006), and time coordinate systems (Rots et al. 2015). An additional spherical projection, called HEALPix, is defined in reference (Calabretta & Roukema 2007). These WCS papers have been formally approved by the IAUFWG and therefore are *incorporated by reference* as an official part of this Standard. The reader should refer to these papers for additional details and background information that cannot be included here. Various updates and corrections to the primary WCS papers have been compiled by the authors, and are reflected in this section. Therefore, where conflicts exist, the description in this Standard will prevail.

8.1. Basic concepts

Rather than store world coordinates separately for each datum, the regular lattice structure of a *FITS* image offers the possibility of defining rules for computing world coordinates at each point. As stated in Sect. 3.3.2 and depicted in Fig. 1, image array data are addressed via *integral array indices* that range in value from 1 to `NAXIS`*j* on axis j. Recognizing that image data values may have an extent, for example an angular separation, spectral channel width or time span, and thus that it may make sense to interpolate between them, these integral array indices may be generalized to floating-point *pixel coordinates*. Integral pixel coordinate values coincide with the corresponding array indices, while fractional pixel coordinate values lie between array indices and thus imply interpolation. Pixel coordinate values are defined at all points within the image lattice and outside it (except along *conventional* axes, see Sect. 8.5). They form the basis of the world coordinate formalism in *FITS* depicted schematically in Fig. 2.

The essence of representing world coordinate systems in *FITS* is the association of various reserved keywords with elements of a transformation (or a series of transformations), or with parameters of a projection function. The conversion from pixel coordinates in the data array to world coordinates is simply a matter of applying the specified transformations (in order) via the appropriate keyword values; conversely, defining a WCS for an image amounts to solving for the elements of the transformation matrix(es) or coefficients of the function(s) of interest and recording them in the form of WCS keyword values. The description of the WCS systems and their expression in *FITS* HDUs is quite extensive and detailed, but is aided by a careful choice of notation. Key elements of the notation are summarized in Table 21, and are used throughout this section. The formal definitions of the keywords appear in the following subsections.

The conversion of image pixel coordinates to world coordinates is a multi-step process, as illustrated in Fig. 2.

For all coordinate types, the first step is a linear transformation applied via matrix multiplication of the vector of pixel coordinate elements, p_j:

$$q_i = \sum_{j=1}^{N} m_{ij}(p_j - r_j) \tag{9}$$

Table 21: WCS and celestial coordinates notation.

Variable(s)	Meaning	Related FITS keywords
i	Index variable for world coordinates	
j	Index variable for pixel coordinates	
a	Alternative WCS version code	
p_j	Pixel coordinates	
r_j	Reference pixel coordinates	CRPIX*ja*
m_{ij}	Linear transformation matrix	CD*i_ja* or PC*i_ja*
s_i	Coordinate scales	CDELT*ia*
(x, y)	Projection plane coordinates	
(ϕ, θ)	Native longitude and latitude	
(α, δ)	Celestial longitude and latitude	
(ϕ_0, θ_0)	Native longitude and latitude of the fiducial point	PV*i*_1*a*† , PV*i*_2*a*†
(α_0, δ_0)	Celestial longitude and latitude of the fiducial point	CRVAL*ia*
(α_p, δ_p)	Celestial longitude and latitude of the native pole	
(ϕ_p, θ_p)	Native longitude and latitude of the celestial pole	LONPOLE*a* (=PV*i*_3*a*†), LATPOLE*a* (=PV*i*_4*a*†)

Notes. † Associated with *longitude* axis *i*.

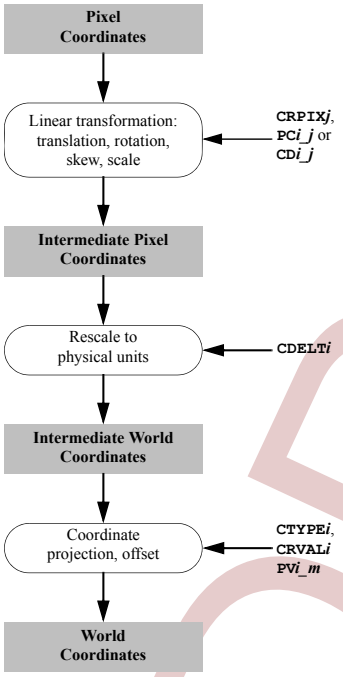

Fig. 2: A schematic view of converting pixel coordinates to world coordinates.

where r_j are the pixel coordinate elements of the reference point, j indexes the pixel axis, and i the world axis. The m_{ij} matrix is a non-singular, square matrix of dimension $N \times N$, where N is the number of world coordinate axes. The elements q_i of the resulting *intermediate pixel coordinate* vector are offsets, in dimensionless pixel units, from the reference point along axes coincident with those of the *intermediate world coordinates*. Thus, the conversion of q_i to the corresponding intermediate world coordinate element x_i is a simple scale:

$$x_i = s_i q_i. \tag{10}$$

There are three conventions for associating *FITS* keywords with the above transformations. In the first formalism, the matrix elements m_{ij} are encoded in the PC*i_j* keywords and the scale factors s_i are encoded in the CDELT*i* keywords, which *must* have non-zero values. In the second formalism Eqs. (9) and (10) are combined as

$$x_i = \sum_{j=1}^{N} (s_i m_{ij})(p_j - r_j) \tag{11}$$

and the CD*i_j* keywords encode the product $s_i m_{ij}$. The third convention was widely used before the development of the two previously described conventions and uses the CDELT*i* keywords to define the image scale and the CROTA2 keyword to define a bulk rotation of the image plane. Use of the CROTA2 keyword is now deprecated, and instead the newer PC*i_j* or CD*i_j* keywords are *recommended* because they allow for skewed axes and fully general rotation of multi-dimensional arrays. The CDELT*i* and CROTA2 keywords *may* co-exist with the CD*i_j* keywords (but the CROTA2 *must not* occur with the PC*i_j* keywords) as an aid to old *FITS* interpreters, but these keywords *must* be ignored by software that supports the CD*i_j* keyword convention. In all these formalisms the reference pixel coordinates r_j are encoded in the CRPIX*i* keywords, and the world coordinates at the reference point are encoded in the CRVAL*i* keywords. For additional details, see Greisen & Calabretta (2002).

The third step of the process, computing the final world coordinates, depends on the type of coordinate system, which is indicated with the value of the CTYPE*i* keyword. For some simple, linear cases an appropriate choice of normalization for the scale factors allows the world coordinates to be taken directly (or by applying a constant offset) from the x_i (e.g., some spectra). In other cases it is more complicated, and may require the application of some non-linear algorithm (e.g., a projection, as for celestial coordinates), which may require the specification of additional parameters. Where necessary, numeric parameter values for non-linear algorithms *must* be specified via PV*i_m* keywords and character-valued parameters will be specified via PS*i_m* keywords, where *m* is the parameter number.

The application of these formalisms to coordinate systems of interest is discussed in the following sub-sections: Sect. 8.2 describes general WCS representations (see Greisen & Calabretta 2002), Sect. 8.3 describes celestial coordinate systems (see Calabretta & Greisen 2002)), Sect. 8.4 describes spectral coordinate systems (see Greisen et al. 2006), and Sect. 9 describes the representation of time coordinates (see Rots et al. 2015).

8.2. World coordinate system representations

A variety of keywords have been reserved for computing the coordinate values that are to be associated with any pixel location within an array. The full set is given in Table 22; those in most common usage are defined in detail below for convenience. Coordinate system specifications may appear in HDUs that contain simple images in the primary array or in an image extension. Images may also be stored in a multi-dimensional vector cell of a binary table, or as a tabulated list of pixel locations (and optionally, the pixel value) in a table. In these last two types of image representations, the WCS keywords have a different naming convention which reflects the needs of the tabular data structure and the 8-character limit for keyword lengths, but otherwise follow exactly the same rules for type, usage, and default values. See reference Calabretta & Greisen (2002) for example usage of these keywords. All forms of these reserved keywords *must* be used only as specified in this Standard.

The keywords given below constitute a complete set of fundamental attributes for a WCS description. Although their inclusion in an HDU is optional, *FITS* writers *should* include a complete set of keywords when describing a WCS. In the event that some keywords are missing, default values *must* be assumed, as specified below.

`WCSAXES` – [integer; default: `NAXIS`, or larger of WCS indexes *i* or *j*]. Number of axes in the WCS description. This keyword, if present, *must* precede all WCS keywords except `NAXIS` in the HDU. The value of `WCSAXES` *may* exceed the number of pixel axes for the HDU.

`CTYPE`*i* – [character; indexed; default: '␣' (i.e. a linear, undefined axis)]. Type for the intermediate coordinate axis *i*. Any coordinate type that is not covered by this standard or an officially recognized *FITS* convention *shall* be taken to be linear. All non-linear coordinate system names *must* be expressed in '4–3' form: the first four characters specify the coordinate type, the fifth character is a hyphen ('-'), and the remaining three characters specify an algorithm code for computing the world coordinate value. Coordinate types with names of less than four characters are padded on the right with hyphens, and algorithm codes with less than three characters are padded on the right with blanks[9]. Algorithm codes *should* be three characters.

`CUNIT`*i* – [character; indexed; default: '␣' (i.e., undefined)]. Physical units of `CRVAL` and `CDELT` for axis *i*. Note that units *should* always be specified (see Sect. 4.3). Units for celestial coordinate systems defined in this Standard *must* be degrees.

`CRPIX`*j* – [floating point; indexed; default: 0.0]. Location of the reference point in the image for axis *j* corresponding to r_j in Eq. (9). Note that the reference point *may* lie outside the image and that the first pixel in the image has pixel coordinates (1.0, 1.0, . . .).

`CRVAL`*i* – [floating point; indexed; default: 0.0]. World Coordinate value at the reference point of axis *i*.

`CDELT`*i* – [floating point; indexed; default: 1.0]. Increment of the world coordinate at the reference point for axis *i*. The value *must not* be zero.

`CROTA`*i* – [floating point; indexed; default: 0.0]. The amount of rotation from the standard coordinate system to a different coordinate system. Further use of this of this keyword is deprecated, in favor of the newer formalisms that use the `CD`*i_j* or `PC`*i_j* keywords to define the rotation.

`PC`*i_j* – [floating point; defaults: 1.0 when $i = j$, 0.0 otherwise]. Linear transformation matrix between pixel axes *j* and intermediate coordinate axes *i*. The `PC`*i_j* matrix *must not* be singular.

`CD`*i_j* – [floating point; defaults: 0.0, but see below]. Linear transformation matrix (with scale) between pixel axes *j* and intermediate coordinate axes *i*. This nomenclature is equivalent to `PC`*i_j* when `CDELT`*i* is unity. The `CD`*i_j* matrix *must not* be singular. Note that the `CD`*i_j* formalism is an exclusive alternative to `PC`*i_j*, and the `CD`*i_j* and `PC`*i_j* keywords *must not* appear together within an HDU.

In addition to the restrictions noted above, if any `CD`*i_j* keywords are present in the HDU, all other unspecified `CD`*i_j* keywords *shall* default to zero. If no `CD`*i_j* keywords are present then the header *shall* be interpreted as being in `PC`*i_j* form whether or not any `PC`*i_j* keywords are actually present in the HDU.

Some non-linear algorithms that describe the transformation between pixel and intermediate coordinate axes require parameter values. A few non-linear algorithms also require character-valued parameters, e.g., table lookups require the names of the table extension and the columns to be used. Where necessary parameter values *must* be specified via the following keywords:

`PV`*i_m* – [floating point]. Numeric parameter values for intermediate world coordinate axis *i*, where *m* is the parameter number. Leading zeros *must not* be used, and *m* may have only values in the range 0 through 99, and that are defined for the particular non-linear algorithm.

`PS`*i_m* – [character]. Character-valued parameters for intermediate world coordinate axis *i*, where *m* is the parameter number. Leading zeros *must not* be used, and *m* may have only values in the range 0 through 99, and that are defined for the particular non-linear algorithm.

The following keywords, while not essential for a complete specification of an image WCS, can be extremely useful for readers to interpret the accuracy of the WCS representation of the image.

`CRDER`*i* – [floating point; default: 0.0]. Random error in coordinate *i*, which *must* be non-negative.

`CSYER`*i* – [floating point; default: 0.0]. Systematic error in coordinate *i*, which *must* be non-negative.

These values *should* give a representative average value of the error over the range of the coordinate in the HDU. The total error in the coordinates would be given by summing the individual errors in quadrature.

8.2.1. Alternative WCS axis descriptions

In some cases it is useful to describe an image with more than one coordinate type[10]. Alternative WCS descriptions *may* be added to the header by adding the appropriate sets of WCS keywords, and appending to all keywords in each set an alphabetic

[9] Example: '`RA---UV `'.

[10] Examples include the frequency, velocity, and wavelength along a spectral axis (only one of which, of course, could be linear), or the position along an imaging detector in both meters and degrees on the sky.

Table 22: Reserved WCS keywords (continues on next page)

			BINTABLE vector		Pixel list	
Keyword Description	Global	Image	Primary	Alternative	Primary	Alternative
Coordinate dimensionality		WCSAXES*a*	WCAX*na*		...	
Axis type		CTYPE*ia*	*i*CTYP*n*	*i*CTY*na*	TCTYP*n*	TCTY*na*
Axis units		CUNIT*ia*	*i*CUNI*n*	*i*CUN*na*	TCUNI*n*	TCUN*na*
Reference value		CRVAL*ia*	*i*CRVL*n*	*i*CRV*na*	TCRVL*n*	TCRV*na*
Coordinate increment		CDELT*ia*	*i*CDLT*n*	*i*CDE*na*	TCDLT*n*	TCDE*na*
Reference point		CRPIX*ja*	*j*CRPX*n*	*j*CRP*na*	TCRPX*n*	TCRP*na*
Coordinate rotation[1]		CROTA*i*	*i*CROT*n*		TCROT*n*	
Transformation matrix[2]		PC*i_ja*	*ij*PC*na*		TPC*n_ka* or TP*n_ka*	
Transformation matrix[2]		CD*i_ja*	*ij*CD*na*		TCD*n_ka* or TC*n_ka*	
Coordinate parameter		PV*i_ma*	*i*PV*n_ma* or *i*V*n_ma*		TPV*n_ma* or TV*n_ma*	
Coordinate parameter array		...	*i*V*n_Xa*		...	
Coordinate parameter		PS*i_ma*	*i*PS*n_ma* or *i*S*n_ma*		TPS*n_ma* or TS*n_ma*	
Coordinate name		WCSNAME*a*	WCSN*na*		WCS*na* or TWCS*na*	
Coordinate axis name		CNAME*ia*	*i*CNA*na*		TCNA*na*	
Random error		CRDER*ia*	*i*CRD*na*		TCRD*na*	
Systematic error		CSYER*ia*	*i*CSY*na*		TCSY*na*	
WCS cross-reference target		...	WCST*na*		...	
WCS cross reference		...	WCSX*na*		...	
Coordinate rotation		LONPOLE*a*	LONP*na*		LONP*na*	
Coordinate rotation		LATPOLE*a*	LATP*na*		LATP*na*	
Coordinate epoch		EQUINOX*a*	EQUI*na*		EQUI*na*	
Coordinate epoch[3]	EPOCH		EPOCH		EPOCH	
Reference frame	RADECSYS[4]	RADESYS*a*	RADE*na*		RADE*na*	
Line rest frequency (Hz)	RESTFREQ[4]	RESTFRQ*a*	RFRQ*na*		RFRQ*na*	
Line rest vacuum wavelength (m)		RESTWAV*a*	RWAV*na*		RWAV*na*	
Spectral reference frame		SPECSYS*a*	SPEC*na*		SPEC*na*	
Spectral reference frame		SSYSOBS*a*	SOBS*na*		SOBS*na*	
Spectral reference frame		SSYSSRC*a*	SSRC*na*		SSRC*na*	
Observation X (m)	OBSGEO-X[5]		OBSGX*n*		OBSGX*n*	
Observation Y (m)	OBSGEO-Y[5]		OBSGY*n*		OBSGY*n*	
Observation Z (m)	OBSGEO-Z[5]		OBSGZ*n*		OBSGZ*n*	
Radial velocity (m s^{-1})		VELOSYS*a*	VSYS*na*		VSYS*na*	
Redshift of source		ZSOURCE*a*	ZSOU*na*		ZSOU*na*	
Angle of true velocity		VELANGL*a*	VANG*na*		VANG*na*	
Date-time related keywords (see Sect.9)						
Date of HDU creation	DATE					
Date/time of observation	DATE-OBS		DOBS*n*		DOBS*n*	
	MJD-OBS		MJDOB*n*		MJDOB*n*	
	BEPOCH					
	JEPOCH					
Average date/time of observation	DATE-AVG		DAVG*n*		DAVG*n*	
	MJD-AVG		MJDA*n*		MJDA*n*	
Start date/time of observation	DATE-BEG					
	MJD-BEG					
	TSTART					
End date/time of observation	DATE-END					
	MJD-END					
	TSTOP					
Net exposure duration	XPOSURE					
Wall clock exposure duration	TELAPSE					
Time scale	TIMESYS	CTYPE*ia*	*i*CTYP*n*	*i*CTY*na*	TCTYP*n*	TCTY*na*
Time zero-point (MJD)	MJDREF[6]					
Time zero-point (JD)	JDREF[6]					
Time zero-point (ISO)	DATEREF					
Reference position	TREFPOS		TRPOS*n*		TRPOS*n*	
Reference direction	TREFDIR		TRDIR*n*		TRDIR*n*	
Solar system ephemeris	PLEPHEM					
Time unit	TIMEUNIT	CUNIT*ia*	*i*CUNI*n*	*i*CUN*na*	TCUNI*n*	TCUN*na*
Time offset	TIMEOFFS					
Time absolute error	TIMSYER	CSYER*ia*	*i*CSYE*n*	*i*CSY*na*	TCSY*n*	TCSY*na*
Time relative error	TIMRDER	CRDER*ia*	*i*CRDE*n*	*i*CRD*na*	TCRD*n*	TCRD*na*
Time resolution	TIMEDEL					
Time location in pixel	TIMEPIXR					
Phase axis zero point		CZPHS*ia*	*i*CZPH*n*	*i*CZP*na*	TCZPH*n*	TCZP*na*
Phase axis period		CPER*ia*	*i*CPER*n*	*i*CPR*na*	TCPER*n*	TCPR*na*

Foootnotes for Table 22

Notes. The indexes *j* and *i* are pixel and intermediate world coordinate axis numbers, respectively. Within a table, the index *n* refers to a column number, and *m* refers to a coordinate parameter number. The index *k* also refers to a column number. The indicator *a* is either blank (for the primary coordinate description) or a character A through Z that specifies the coordinate version. See text.
(1) CROTA*i* form is deprecated but still in use. It *must not* be used with PC*i_j*, PV*i_m*, and PS*i_m*. (2) PC*i_j* and CD*i_j* forms of the transformation matrix are mutually exclusive, and *must not* appear together in the same HDU. (3) EPOCH is deprecated. Use EQUINOX instead. (4) These 8-character keywords are deprecated; the 7-character forms, which can include an alternate version code letter at the end, *should* be used instead. (5) For the purpose of time reference position, geodetic latitude/longitude/elevation OBSGEO-B, OBSGEO-L, OBSGEO-H or an orbital ephemeris keyword OBSORBIT can be also used (see Sect. 9.2.3). (6) [M]JDREF can be split in integer and fractional values [M]JDREFI and [M]JDREFF as explained in Sect. 9.2.2.

code in the range A through Z. Keywords that may be used in this way to specify a coordinate system version are indicated in Table 22 with the suffix *a*. All implied keywords with this encoding are *reserved keywords*, and *must* only be used in *FITS* HDUs as specified in this Standard. The axis numbers *must* lie in the range 1 through 99, and the coordinate parameter *m must* lie in the range 0 through 99, both with no leading zeros.

The *primary* version of the WCS description is that specified with *a* as the blank character[11]. Alternative axis descriptions are optional, but *must not* be specified unless the primary WCS description is also specified. If an alternative WCS description is specified, all coordinate keywords for that version *must* be given even if the values do not differ from those of the primary version. Rules for the default values of alternative coordinate descriptions are the same as those for the primary description. The alternative descriptions are computed in the same fashion as the primary coordinates. The type of coordinate depends on the value of CTYPE*ia*, and may be linear in one of the alternative descriptions and non-linear in another.

The alternative version codes are selected by the *FITS* writer; there is no requirement that the codes be used in alphabetic sequence, nor that one coordinate version differ in its parameter values from another. An optional keyword WCSNAME*a* is also defined to name, and otherwise document, the various versions of WCS descriptions:

WCSNAME*a* – [character; default for *a*: '␣' (i.e., blank, for the primary WCS, else a character A through Z that specifies the coordinate version]. Name of the world coordinate system represented by the WCS keywords with the suffix *a*. Its primary function is to provide a means by which to specify a particular WCS if multiple versions are defined in the HDU.

8.3. Celestial coordinate system representations

The conversion from intermediate world coordinates (x, y) in the plane of projection to celestial coordinates involves two steps: a spherical projection to native longitude and latitude (ϕ, θ), defined in terms of a convenient coordinate system (i.e., *native spherical coordinates*), followed by a spherical rotation of these native coordinates to the required celestial coordinate system (α, δ). The algorithm to be used to define the spherical projection *must* be encoded in the CTYPE*i* keyword as the three-letter algorithm code, the allowed values for which are specified in Table 23 and defined in references Calabretta & Greisen (2002) and Calabretta & Roukema (2007). The target celestial coordinate system is also encoded into the left-most portion of the CTYPE*i* keyword as the coordinate type.

For the final step, the parameter LONPOLE*a* must be specified, which is the native longitude of the celestial pole, ϕ_p. For certain projections (such as cylindricals and conics, which are less commonly used in astronomy), the additional keyword LATPOLE*a* must be used to specify the native latitude of the celestial pole. See Calabretta & Greisen (2002) for the transformation equations and other details.

The accepted celestial coordinate systems are: the standard equatorial (RA-- and DEC-), and others of the form *x*LON and *x*LAT for longitude-latitude pairs, where *x* is G for Galactic, E for ecliptic, H for helioecliptic and S for supergalactic coordinates. Since the representation of planetary, lunar, and solar coordinate systems could exceed the 26 possibilities afforded by the single character *x*, pairs of the form *yz*LN and *yz*LT *may* be used as well.

RADESYS*a* – [character; default: FK4, FK5, or ICRS: see below]. Name of the reference frame of equatorial or ecliptic coordinates, whose value *must* be one of those specified in Table 24. The default value is FK4 if the value of EQUINOX*a* < 1984.0, FK5 if EQUINOX*a* ≥ 1984.0, or ICRS if EQUINOX*a* is not given.

EQUINOX*a* – [floating point; default: see below]. Epoch of the mean equator and equinox in years, whose value *must* be non-negative. The interpretation of epoch depends upon the value of RADESYS*a* if present: *Besselian* if the value is FK4 or FK4-NO-E, *Julian* if the value is FK5; *not applicable* if the value is ICRS or GAPPT.

EPOCH – [floating point]. This keyword is deprecated and *should not* be used in new *FITS* files. It is reserved primarily to prevent its use with other meanings. The EQUINOX keyword *shall* be used instead. The value field of this keyword was previously defined to contain a floating-point number giving the equinox in years for the celestial coordinate system in which positions are expressed.

DATE-OBS – [floating point]. This reserved keyword is defined in Sect. 4.4.2.

MJD-OBS – [floating point; default: DATE-OBS if given, otherwise no default]. Modified Julian Date (JD – 2,400,000.5) of the observation, whose value corresponds (by default) to the *start* of the observation, unless another interpretation is explained in the comment field. No specific time system (e.g. UTC, TAI, etc.) is defined for this or any of the other time-related keywords. It is *recommended* that the TIMESYS keyword, as defined in Sect. 9.2.1 be used to specify the time system. See also Sect. 9.5.

LONPOLE*a* – [floating point; default: ϕ_0 if $\delta_0 \geq \theta_0$, $\phi_0 + 180°$ otherwise]. Longitude in the native coordinate system of the celestial system's north pole. Normally, ϕ_0 is zero unless a

[11] There are a number of keywords (e.g. *ij*PC*na*) where the *a* could be pushed off the 8-char keyword name for plausible values of *i*, *j*, *k*, *n*, and *m*. In such cases *a* is still said to be 'blank' although it is not the blank character.

Table 23: Reserved celestial coordinate algorithm codes.

	Default			
Code	ϕ_0	θ_0	Properties[1]	Projection name
			Zenithal (azimuthal) projections	
AZP	0°	90°	Sect. 5.1.1	Zenithal perspective
SZP	0°	90°	Sect. 5.1.2	Slant zenithal perspective
TAN	0°	90°	Sect. 5.1.3	Gnomonic
STG	0°	90°	Sect. 5.1.4	Stereographic
SIN	0°	90°	Sect. 5.1.5	Slant orthographic
ARC	0°	90°	Sect. 5.1.6	Zenithal equidistant
ZPN	0°	90°	Sect. 5.1.7	Zenithal polynomial
ZEA	0°	90°	Sect. 5.1.8	Zenithal equal-area
AIR	0°	90°	Sect. 5.1.9	Airy
			Cylindrical projections	
CYP	0°	0°	Sect. 5.2.1.	Cylindrical perspective
CEA	0°	0°	Sect. 5.2.2	Cylindrical equal area
CAR	0°	0°	Sect. 5.2.3	Plate carrée
MER	0°	0°	Sect. 5.2.4	Mercator
			Pseudo-cylindrical and related projections	
SFL	0°	0°	Sect. 5.3.1	Samson-Flamsteed
PAR	0°	0°	Sect. 5.3.2	Parabolic
MOL	0°	0°	Sect. 5.3.3	Mollweide
AIT	0°	0°	Sect. 5.3.4	Hammer-Aitoff
			Conic projections	
COP	0°	θ_a	Sect. 5.4.1	Conic perspective
COE	0°	θ_a	Sect. 5.4.2	Conic equal-area
COD	0°	θ_a	Sect. 5.4.3	Conic equidistant
COO	0°	θ_a	Sect. 5.4.4	Conic orthomorphic
			Polyconic and pseudoconic projections	
BON	0°	0°	Sect. 5.5.1	Bonne's equal area
PCO	0°	0°	Sect. 5.5.2	Polyconic
			Quad-cube projections	
TSC	0°	0°	Sect. 5.6.1	Tangential spherical cube
CSC	0°	0°	Sect. 5.6.2	COBE quadrilateralized spherical cube
QSC	0°	0°	Sect. 5.6.3	Quadrilateralized spherical cube
			HEALPix grid projection	
HPX	0°	0°	Sect. 6[2]	HEALPix grid

[1] Refer to the indicated section in Calabretta & Greisen (2002) for a detailed description. [2] This projection is defined in Calabretta & Roukema (2007).

non-zero value has been set for PV*i*_1*a*, which is associated with the *longitude* axis. This default applies for all values of θ_0, including $\theta_0 = 90°$, although the use of non-zero values of θ_0 are discouraged in that case.

LATPOLE*a* – [floating point; default: 90°, or no default if $(\theta_0, \delta_0, \phi_p - \phi_0) = (0, 0, \pm 90°)$]. Latitude in the native coordinate system of the celestial system's north pole, or equivalently, the latitude in the celestial coordinate system of the native system's north pole. May be ignored or omitted in cases where LONPOLE*a* completely specifies the rotation to the target celestial system.

8.4. Spectral coordinate system representations

This section discusses the conversion of intermediate world coordinates to spectral coordinates with common axes such as frequency, wavelength, and apparent radial velocity (represented here with the coordinate variables ν, λ, or v). The key point for constructing spectral WCS in *FITS* is that one of these coordinates *must* be sampled linearly in the dispersion axis; the others are derived from prescribed, usually non-linear transformations. Frequency and wavelength axes *may* also be sampled linearly in their logarithm.

Following the convention for the CTYPE*ia* keyword, when *i* is the spectral axis the first four characters *must* specify a code for the coordinate type; for non-linear algorithms the fifth character *must* be a hyphen, and the next three characters *must* specify a predefined algorithm for computing the world coordinates from the intermediate physical coordinates. The coordinate type *must* be one of those specified in Table 25. When the algorithm is linear, the remainder of the CTYPE*ia* keyword *must* be blank. When the algorithm is non-linear, the 3-letter algorithm code *must* be one of those specified in Table 26. The relationships between the basic physical quantities ν, λ, and v, as well as the relationships between various derived quantities are given in reference Greisen et al. (2006).

The generality of the algorithm for specifying the spectral coordinate system and its representation suggests that some additional description of the coordinate may be helpful beyond what can be encoded in the first four characters of the CTYPE*ia* keyword; CNAME*ia* is reserved for this purpose. Note that this keyword provides a name for an axis in a particular WCS, while the WCSNAME*a* keyword names the particular WCS as a whole. In order to convert between some form of radial velocity and

Table 24: Allowed values of RADESYS*a*.

Value	Definition
ICRS	International Celestial Reference System
FK5	Mean place, new (IAU 1984) system
FK4[1]	Mean place, old (Bessel-Newcomb) system
FK4-NO-E[1]	Mean place: but without eccentricity terms
GAPPT	Geocentric apparent place, IAU 1984 system

(1) New FITS files should avoid using these older reference systems.

either frequency or wavelength, the keywords RESTFRQ*a* and RESTWAV*a*, respectively, are reserved.

CNAME*ia* – [character; default: default: '␣' (i.e. a linear, undefined axis)]. Spectral coordinate description which *must not* exceed 68 characters in length.

RESTFRQ*a* – [floating point; default: none]. Rest frequency of the of the spectral feature of interest. The physical unit *must* be Hz.

RESTWAV*a* – [floating point; default: none]. Vacuum rest wavelength of the of the spectral feature of interest. The physical unit *must* be m.

One or the other of RESTFRQ*a* or RESTWAV*a* *should* be given when it is meaningful to do so.

8.4.1. Spectral coordinate reference frames

Frequencies, wavelengths, and apparent radial velocities are always referred to some selected standard of rest (i.e., reference frame). While the spectra are obtained they are, of necessity, in the observer's rest frame. The velocity correction from topocentric (the frame in which the measurements are usually made) to standard reference frames (which *must* be one of those given in Table 27) are dependent on the dot product with time-variable velocity vectors. That is, the velocity with respect to a standard reference frame depends upon direction, and the velocity (and frequency and wavelength) with respect to the local standard of rest is a function of the celestial coordinate within the image. The keywords SPECSYS*a* and SSYSOBS*a* are reserved and, if used, *must* describe the reference frame in use for the spectral axis coordinate(s) and the spectral reference frame that was held constant during the observation, respectively. In order to compute the velocities it is necessary to have the date and time of the observation; the keywords DATE-AVG and MJD-AVG are reserved for this purpose. See also Sect. 9.5.

DATE-AVG – [character; default: none]. Calendar date of the mid-point of the observation, expressed in the same way as the DATE-OBS keyword.

MJD-AVG – [floating point; default: none]. Modified Julian Date (JD – 2,400,000.5) of the mid-point of the observation.

SPECSYS*a* – [character; default: none]. The reference frame in use for the spectral axis coordinate(s). Valid values are given in Table 27.

SSYSOBS*a* – [character; default: TOPOCENT]. The spectral reference frame that is constant over the range of the non-spectral world coordinates. Valid values are given in Table 27.

The transformation from the rest frame of the observer to a standard reference frame requires a specification of the location

Table 26: Non-linear spectral algorithm codes.

Code[1]	Regularly sampled in	Expressed as
F2W	Frequency	Wavelength
F2V		Apparent radial velocity
F2A		Air wavelength
W2F	Wavelength	Frequency
W2V		Apparent radial velocity
W2A		Air wavelength
V2F	Apparent radial vel.	Frequency
V2W		Wavelength
V2A		Air wavelength
A2F	Air wavelength	Frequency
A2W		Wavelength
A2V		Apparent radial velocity
LOG	Logarithm	Any four-letter type code
GRI	Detector	Any type code from Table 25
GRA	Detector	Any type code from Table 25
TAB	Not regular	Any four-letter type code

(1) Characters 6 through 8 of the value of the keyword CTYPE*ia*.

on Earth[12] of the instrument used for the observation in order to calculate the diurnal Doppler correction due to the Earth's rotation. The location, if specified, *shall* be represented as a geocentric Cartesian triple with respect to a standard ellipsoidal geoid at the time of the observation. While the position can often be specified with an accuracy of a meter or better, for most purposes positional errors of several kilometers will have negligible impact on the computed velocity correction. For details, see reference Greisen et al. (2006).

OBSGEO-X*a* – [floating point; default: none]. $X-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

OBSGEO-Y*a* – [floating point; default: none]. $Y-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

OBSGEO-Z*a* – [floating point; default: none]. $Z-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

Information on the relative radial velocity between the observer and the selected standard of rest in the direction of the celestial reference coordinate *may* be provided, and if so *shall* be given by the VELOSYS*a* keyword. The frame of rest defined with respect to the emitting source may be represented in *FITS*; for this reference frame it is necessary to define the velocity with respect to some other frame of rest. The keywords SPECSYS*a* and ZSOURCE*a* are used to document the choice of reference frame and the value of the systemic velocity of the source, respectively.

SSYSSRC*a* – [character; default: none]. Reference frame for the value expressed in the ZSOURCE*a* keyword to document the

[12] The specification of location for an instrument on a spacecraft in flight requires an ephemeris; keywords that might be required in this circumstance are not defined here.

Table 25: Reserved spectral coordinate type codes.

Code[1]	Type	Symbol	Assoc. variable	Default units
FREQ	Frequency	ν	ν	Hz
ENER	Energy	E	ν	J
WAVN	Wavenumber	κ	ν	m^{-1}
VRAD	Radio velocity[2]	V	ν	m s^{-1}
WAVE	Vacuum wavelength	λ	λ	m
VOPT	Optical velocity[2]	Z	λ	m s^{-1}
ZOPT	Redshift	z	λ	...
AWAV	Air wavelength	λ_a	λ_a	m
VELO	Apparent radial velocity	v	v	m s^{-1}
BETA	Beta factor (v/c)	β	v	...

(1) Characters 1 through 4 of the value of the keyword CTYPE*ia*. (2) By convention, the 'radio' velocity is given by $c(\nu_0 - \nu)/\nu_0$ and the 'optical' velocity is given by $c(\lambda - \lambda_0)/\lambda_0$.

Table 27: Spectral reference systems.

Value	Definition
TOPOCENT	Topocentric
GEOCENTR	Geocentric
BARYCENT	Barycentric
HELIOCEN	Heliocentric
LSRK	Local standard of rest (kinematic)
LSRD	Local standard of rest (dynamic)
GALACTOC	Galactocentric
LOCALGRP	Local Group
CMBDIPOL	Cosmic microwave background dipole
SOURCE	Source rest frame

Notes. These are the allowed values of the SPECSYS*a*, SSYSOBS*a*, and SSYSSRC*a* keywords.

systemic velocity of the observed source. Value *must* be one of those given in Table 27 *except* for SOURCE.

VELOSY*a* – [floating point; default: none]. Relative radial velocity between the observer and the selected standard of rest in the direction of the celestial reference coordinate. Units *must* be m s^{-1}. The CUNIT*ia* keyword is not used for this purpose since the WCS version *a* might not be expressed in velocity units.

ZSOURCE*a* – [floating point; default: none]. Radial velocity with respect to an alternative frame of rest, expressed as a unitless redshift (i.e., velocity as a fraction of the speed of light in vacuum). Used in conjunction with SSYSSRC*a* to document the systemic velocity of the observed source.

VELANGL*a* – [floating point; default:+90.]. In the case of relativistic velocities (e.g., a beamed astrophysical jet) the transverse velocity component is important. This keyword may be used to express the orientation of the space velocity vector with respect to the plane of the sky. See Appendix A of reference Greisen et al. (2006) for further details.

8.5. Conventional coordinate types

The first *FITS* paper (Wells et al. 1981) listed a number of 'suggested values' for the CTYPE*i* keyword. Two of these have the attribute the associated world coordinates can assume only integer values and that the meaning of these integers is only defined by convention. The first 'conventional' coordinate is CTYPE*ia* = 'COMPLEX' to specify that complex values (i.e., pairs of real and imaginary components) are stored in the data array (along with an optional weight factor). Thus, the complex axis of the data array will contain two values (or three if the weight is specified). By convention, the real component has a coordinate value of 1, the imaginary component has a coordinate value of 2, and the weight, if any, has a coordinate value of 3. Table 28 illustrates the required keywords for an array of 100 complex values (without weights).

Table 28: Example keywords for a 100 element array of complex values.

Keyword		
SIMPLE	=	T
BITPIX	=	-32
NAXIS	=	2
NAXIS1	=	2
NAXIS2	=	100
CTYPE1	=	'COMPLEX'
CRVAL1	=	0.
CRPIX1	=	0.
CDELT1	=	1.
END		

Table 29: Conventional Stokes values.

Value	Symbol	Polarization
1	I	Standard Stokes unpolarized
2	Q	Standard Stokes linear
3	U	Standard Stokes linear
4	V	Standard Stokes circular
−1	RR	Right-right circular
−2	LL	Left-left circular
−3	RL	Right-left cross-circular
−4	LR	Left-right cross-circular
−5	XX	X parallel linear
−6	YY	Y parallel linear
−7	XY	XY cross linear
−8	YX	YX cross linear

The second conventional coordinate is CTYPE*ia* = 'STOKES' to specify the polarization of the data. Conventional values, their symbols, and polarizations are given in Table 29.

9. Representations of time coordinates

Time as a dimension in astronomical data presents challenges for its representation in FITS files. This section formulates the representation of the time axis, or possibly multiple time axes,

into the World Coordinate System (WCS) described in Sect. 8. Much of the basic structure is employed, while extensions are developed to cope with the differences between time and spatial dimensions; notable amongst these differences is the huge dynamic range, covering the highest resolution timing relative to the age of the Universe.

The precision with which any time stamp conforms to any conventional time scale is highly dependent on the characteristics of the acquiring system. The definitions of many conventional time scales vary over their history along with the precision that can be attributed to any time stamp. The meaning of any time stamp may be ambiguous if a time scale is used for dates prior to its definition by a recognized authority, or for dates after that definition is abandoned. However, common sense should prevail: the precision in the description of the time coordinate should be appropriate to the accuracy of the temporal information in the data.

9.1. Time values

The three most common ways to specify time are: ISO-8601 (ISO 2004b), Julian Date (JD), or Modified Julian Date (MJD = JD − 2,400,000.5; see IAU 1997). Julian Dates are counted from Julian proleptic calendar date 1 January 4713 BCE at noon, or Gregorian proleptic calendar date 24 November 4714 BCE, also at noon. For an explanation of the calendars, see Rots et al. (2015). Even though it is common to think of certain representations of time as absolute, time values in FITS files *shall* all be considered relative: elapsed time since a particular reference point in time. It may help to view the "absolute" values as merely relative to a globally accepted zero point. For a discussion of the precision required to represent time values in floating-point numbers, see Rots et al. (2015).

9.1.1. ISO-8601 *datetime* strings

FITS datetime strings conform to a subset of ISO-8601 (which in itself does not imply a particular time scale) for several time-related keywords (Bunclark & Rots 1997), such as `DATE-xxxx`. Here *datetime* will be used as a pseudo data type to indicate its use, although its values *must* be written as a character string in `A` format. The full specification for the format of the *datetime* string has been:

`CCYY-MM-DD[Thh:mm:ss[.s...]]`

All of the time part *may* be omitted (just leaving the date) or the decimal seconds *may* be omitted. Leading zeroes *must not* be omitted and timezone designators are *not allowed*. This definition is extended to allow five-digit years with a *mandatory* sign, in accordance with ISO-8601. That is, one *shall* use either the *unsigned* four-digit year format or the *signed* five-digit year format:

`[±C]CCYY-MM-DD[Thh:mm:ss[.s...]]`

Note the following:

– In counting years, ISO-8601 follows the convention of including year zero. Consequently, for negative year numbers there is an offset of one from BCE dates which do not recognize a year zero. Thus year 1 corresponds to 1 CE, year 0 to 1 BCE, year −1 to 2 BCE, and so on.
– The earliest date that may be represented in the four-digit year format is `0000-01-01T00:00:00` (in the year 1 BCE); the latest date is `9999-12-31T23:59:59`. This representation of time is tied to the Gregorian calendar. In conformance with the present ISO-8601:2004(E) standard (ISO 2004b) dates prior to 1582 *must* be interpreted according to the proleptic application of the rules of Gregorius XIII. For dates not covered by that range the use of Modified Julian Date (MJD) or Julian Date (JD) numbers or the use of the signed five-digit year format is *recommended*.
– In the five-digit year format the earliest and latest dates are `-99999-01-01T00:00:00` (i.e., −100 000 BCE) and `+99999-12-31T23:59:59`.
– The origin of JD can be written as: `-04713-11-24T12:00:00`.
– In time scale UTC the integer part of the seconds field runs from 00 to 60 (in order to accommodate leap seconds); in all other time scales the range is 00 to 59.
– The ISO-8601 *datetime* data type is *not allowed* in image axis descriptions since `CRVAL` is required to be a floating point value.
– ISO-8601 *datetime* does not imply the use of any particular time scale (see Section 9.2.1).
– As specified by Bunclark & Rots (1997), time zones are explicitly not supported in FITS and, consequently, appending the letter `'Z'` to a FITS ISO-8601 string is *not allowed*. The rationale for this rule is that its role in the ISO standard is that of a time zone indicator, not a time scale indicator. As the concept of a time zone is not supported in FITS, the use of time zone indicator is inappropriate.

9.1.2. Julian and Besselian epochs

In a variety of contexts *epochs* are provided with astronomical data. Until 1976 these were commonly based on the Besselian year (see Sect. 9.3), with standard epochs B1900.0 and B1950.0. After 1976 the transition was made to Julian epochs based on the Julian year of 365.25 days, with the standard epoch J2000.0. They are tied to time scales ET and TDB, respectively. Note that the Besselian epochs are scaled by the variable length of the Besselian year (see Sect. 9.3 and its cautionary note, which also applies to this context). The Julian epochs are easier to calculate, as long as one keeps track of leap days.

9.2. Time coordinate frame

9.2.1. Time scale

The *time scale* defines the temporal reference frame, and is specified in the header in one of a few ways, depending upon the context. When recorded as a global keyword, the time scale *shall* be specified by:

`TIMESYS` – [character; default: `'UTC'`]. The value field of this keyword *shall* contain a character string code for the time scale of the time-related keywords. The *recommended* values for this keyword in Table 30 have well defined meanings, but other values *may* be used. If this keyword is absent, `'UTC'` *must* be assumed.

In relevant contexts (e.g., time axes in image arrays, table columns, or random groups) TIMESYS *may* be overridden by a time scale recorded in CTYPE*ia*, its binary table equivalents, or PTYPE*i* (see Table 22).

The keywords TIMESYS, CTYPE*ia*, TCTYP*n*, and TCTY*na* or binary table equivalent *may* assume the values listed in Table 30. In addition, for backward compatibility, all except TIMESYS and PTYPE*i* *may* also assume the value TIME (case-insensitive), whereupon the time scale *shall* be that recorded in TIMESYS or, in its absence, its default value, UTC. As noted above, local time scales other than those listed in Table 30 *may* be used, but their use *should* be restricted to alternate coordinates in order that the primary coordinates will always refer to a properly recognized time scale.

See Rots et al. (2015), Appendix A, for a detailed discussion of the various time scales. In cases where high-precision timing is important one may append a specific realization, in parentheses, to the values in the table; e.g., TT(TAI), TT(BIPM08), UTC(NIST). Note that linearity is not preserved across all time scales. Specifically, if the reference position remains unchanged (see Section 9.2.3), the first ten, with the exception of UT1, are linear transformations of each other (excepting leap seconds), as are TDB and TCB. On average TCB runs faster than TCG by approximately 1.6×10^{-8}, but the transformation from TT or TCG (which are linearly related) is to be achieved through a time ephemeris as provided by Irwin & Fukushima (1999).

The relations between coordinate time scales and their dynamical equivalents have been defined as:

$$T(\mathrm{TCG}) = T(\mathrm{TT}) + L_G \times 86400 \times (JD(\mathrm{TT}) - JD_0)$$
$$T(\mathrm{TDB}) = T(\mathrm{TCB}) - L_B \times 86400 \times (JD(\mathrm{TCB}) - JD_0) + TDB_0$$

where:

T is in seconds
$L_G = 6.969290134 \times 10^{-10}$
$L_B = 1.550519768 \times 10^{-8}$
$JD_0 = 2443144.5003725$
$TDB_0 = -6.55 \times 10^{-5}$ s

Linearity is virtually guaranteed since images and individual table columns do not allow more than one reference position to be associated with them, and since there is no overlap between reference positions that are meaningful for the first nine time scales on the one hand, and for the barycentric ones on the other. All use of the time scale GMT in FITS files *shall* be taken to have its zero point at midnight, conformant with UT, including dates prior to 1925. For high-precision timing prior to 1972, see Rots et al. (2015), Appendix A.

Some time scales in use are not listed in Table 30 because they are intrinsically unreliable or ill-defined. When used, they *should* be tied to one of the existing scales with appropriate specification of the uncertainties; the same is true for free-running clocks. However, a local time scale such as MET (Mission Elapsed Time) or OET (Observation Elapsed Time) *may* be defined for practical reasons. In those cases the time reference value (see Section 9.2.2) *shall not* be applied to the values, and it is strongly *recommended* that such timescales be provided as alternate time scales, with a defined conversion to a recognized time scale.

It useful to note that while UT1 is, in essence, an angle (of the Earth's rotation – *i.e.*, a *clock*), the others are SI-second counters (*chronometers*); UTC, by employing leap seconds, serves as a bridge between the two types of time scales.

Table 30: Recognized Time Scale Values

Value	Meaning
TAI	(International Atomic Time): atomic time standard maintained on the rotating geoid
TT	(Terrestrial Time; IAU standard): defined on the rotating geoid, usually derived as TAI + 32.184 s
TDT	(Terrestrial Dynamical Time): synonym for TT (deprecated)
ET	(Ephemeris Time): continuous with TT; should not be used for data taken after 1984-01-01
IAT	synonym for TAI (deprecated)
UT1	(Universal Time): Earth rotation time
UTC	(Universal Time, Coordinated; default): runs synchronously with TAI, except for the occasional insertion of leap seconds intended to keep UTC within 0.9 s of UT1; as of 2012-07-01 UTC = TAI − 35 s
GMT	(Greenwich Mean Time): continuous with UTC; its use is deprecated for dates after 1972-01-01
UT()[1]	(Universal Time, with qualifier): for high-precision use of radio signal distributions between 1955 and 1972; see Rots et al. (2015), Appendix A
GPS	(Global Positioning System): runs (approximately) synchronously with TAI; GPS ≈ TAI − 19 s
TCG	(Geocentric Coordinate Time): TT reduced to the geocenter, corrected for the relativistic effects of the Earth's rotation and gravitational potential; TCG runs faster than TT at a constant rate
TCB	(Barycentric Coordinate Time): derived from TCG by a 4-dimensional transformation, taking into account the relativistic effects of the gravitational potential at the barycenter (relative to that on the rotating geoid) as well as velocity time dilation variations due to the eccentricity of the Earth's orbit, thus ensuring consistency with fundamental physical constants; Irwin & Fukushima (1999) provide a time ephemeris
TDB	(Barycentric Dynamical Time): runs slower than TCB at a constant rate so as to remain approximately in step with TT; runs therefore quasi-synchronously with TT, except for the relativistic effects introduced by variations in the Earth's velocity relative to the barycenter. When referring to celestial observations, a pathlength correction to the barycenter may be needed which requires the Time Reference Direction used in calculating the pathlength correction.
LOCAL	for simulation data and for free-running clocks.

[1] Specific realization codes may be appended to these values, in parentheses; see text. For a more detailed discussion of time scales, see Rots et al. (2015), Appendix A.

9.2.2. Time reference value

The time reference value is *not required* to be present in an HDU. However, if the time reference point is specified explicitly it *must* be expressed in one of ISO-8601, JD, or MJD. These reference values *must* only be applied to time values associated with one of the recognized time scales listed in Table 30, and that time scale *must* be specified.

The reference point in time, to which all times in the HDU are relative, *shall* be specified through one of three keywords:

MJDREF – [floating-point]; default: 0.0] The value field of this keyword *shall* contain the value of the reference time in MJD.

JDREF – [floating-point; default: None] The value field of this keyword *shall* contain the value of the reference time in JD.

DATEREF – [datetime; default: None] The value field of this keyword *shall* contain a character string representation of the reference time in ISO-8601 format.

MJDREF and JDREF *may* , for clarity or precision reasons, be split into two keywords holding the integer and fractional parts separately:

MJDREFI – [integer; default: 0] The value field of this keyword *shall* contain the integer part of reference time in MJD.

MJDREFF – [floating-point; default: 0.0] The value field of this keyword *shall* contain the fractional part of reference time in MJD.

JDREFI – [integer; default: None] The value field of this keyword *shall* contain the integer part of reference time in JD.

JDREFF – [floating-point; default: None] The value field of this keyword *shall* contain the fractional part of reference time in JD.

If [M]JDREF and both [M]JDREFI and [M]JDREFF are present, the integer and fractional values shall have precedence over the single value. If the single value is present with one of the two parts, the single value shall have precedence. In the following, MJDREF and JDREF refer to their literal meaning or the combination of their integer and fractional parts. If a header contains more than one of these keywords, JDREF *shall* have precedence over DATEREF and MJDREF *shall* have precedence over both the others. If none of the three keywords is present, there is no problem as long as all times in the HDU are expressed in ISO-8601; otherwise MJDREF = 0.0 *must* be assumed. If TREFPOS = 'CUSTOM' (Section 9.2.3) it is legitimate for none of the reference time keywords to be present, as one may assume the data are from a simulation. Note that the *value* of the reference time has global validity for all time values, but it does not have a particular time scale associated with it.

9.2.3. Time reference position

An observation is an event in space-time. The reference position specifies the spatial location at which the time is valid, either where the observation was made or the point in space for which light-time corrections have been applied. When recorded as a global keyword, the time reference position *shall* be specified by:

TREFPOS – [character; default: TOPOCENTER]. The value field of this keyword *shall* contain a character string code for the spatial location at which the observation time is valid. The value *should* be one of those given in Table 31. This keyword *shall* apply to time coordinate axes in images as well.

In binary tables different columns *may* represent completely different Time Coordinate Frames. However, each column can have only one time reference position, thus guaranteeing linearity (see Section 9.2.1).

TRPOS*n* – [character; default: TOPOCENTER] The value field of this keyword *shall* contain a character string code for the spatial location at which the observation time is valid. This table keyword *shall* override TREFPOS.

The reference position value *may* be a standard location (such as GEOCENTER or TOPOCENTER) or a point in space defined by specific coordinates. In the latter case one should be aware that a (3-D) spatial coordinate frame needs to be defined that is likely to be different from the frame(s) that the data are associated with. Note that TOPOCENTER is only moderately informative if no observatory location is provided or indicated. The commonly allowed standard values are shown in Table 31. Note that for the gaseous planets the barycenters of their planetary systems, including satellites, are used for obvious reasons. While it is preferable to spell the location names out in full, in order to be consistent with the practice of Greisen et al. (2006) the values are allowed to be truncated to eight characters. Furthermore, in order to allow for alternative spellings, only the first three characters of all these values *shall* be considered significant. The value of the keyword *shall* be case-sensitive.

Table 31: Standard Time Reference Position Values

Value[1]	Meaning
TOPOCENTER	Topocenter: the location from where the observation was made (default)
GEOCENTER	Geocenter
BARYCENTER	Barycenter of the Solar System
RELOCATABLE	Relocatable: to be used for simulation data only
CUSTOM	A position specified by coordinates that is not the observatory location
Less common, but allowed standard values are:	
HELIOCENTER	Heliocenter
GALACTIC	Galactic center
EMBARYCENTER	Earth-Moon barycenter
MERCURY	Center of Mercury
VENUS	Center of Venus
MARS	Center of Mars
JUPITER	Barycenter of the Jupiter system
SATURN	Barycenter of the Saturn system
URANUS	Barycenter of the Uranus system
NEPTUNE	Barycenter of the Neptune system

Notes. [(1)]Recognized values for TREFPOS, TRPOS*n*; only the first three characters of the values are significant and solar system locations are as specified in the JPL Ephemerides.

The reader is cautioned that time scales and reference positions cannot be combined arbitrarily if one wants a clock that runs linearly at TREFPOS. Table 32 provides a summary of compatible combinations. BARYCENTER *should* only be used in conjunction with time scales TDB and TCB and *should* be the only reference position used with these time scales. With proper care GEOCENTER, TOPOCENTER, and EMBARYCENTER are appropriate for the first ten time scales in Table 30. However, relativistic effects introduce a (generally linear) scaling in certain combinations; highly eccentric spacecraft orbits are the exceptions. Problems will arise when using a reference position on another solar system body (including HELIOCENTER). Therefore it is *recommended* to synchronize the local clock with one of the time scales defined on the Earth's surface, TT, TAI, GPS, or UTC (in the last case: beware of leap seconds). This is common practice for spacecraft clocks. Locally, such a clock will not appear to run at a constant rate, because of variations in the gravitational potential and in motions with respect to Earth, but the effects

Table 32: Compatibility of Time Scales and Reference Positions

Reference Position	Time scale[1]				
	TT, TDT TAI, IAT GPS UTC, GMT	TCG	TDB	TCB	LOCAL
TOPOCENTER	t	ls			
GEOCENTER	ls	c			
BARYCENTER			ls	c	
RELOCATABLE					c
Other[2]	re	re			

Notes. (1)Legend (combination is not recommended if no entry); **c**: correct match; reference position coincides with the spatial origin of the space-time coordinates; **t**: correct match on Earth's surface, otherwise usually linear scaling; **ls**: linear relativistic scaling; **re**: non-linear relativistic scaling. (2)All other locations in the solar system.

can be calculated and are probably small compared with errors introduced by the alternative: establishing a local time standard.

In order to provide a complete description, TOPOCENTER requires the observatory's coordinates to be specified. There are three options: *(a)* the ITRS Cartesian coordinates defined in Sect. 8.4.1 (OBSGEO-X, OBSGEO-Y, OBSGEO-Z), which are *strongly preferred*; *(b)* a geodetic latitude/longitude/elevation triplet (defined below); or *(c)* a reference to an orbit ephemeris file. A set of geodetic coordinates is recognized:

OBSGEO-B – [floating-point] The value field of this keyword *shall* contain the latitude of the observation in deg, with North positive.

OBSGEO-L – [floating-point] The value field of this keyword *shall* contain the longitude of the observation in deg, with East positive.

OBSGEO-H – [floating-point] The value field of this keyword *shall* contain the altitude of the observation in m.

An orbital ephemeris file can instead be specified:

OBSORBIT – [character] The value field of this keyword *shall* contain the character-string URI, URL, or the name of an orbit ephemeris file.

Beware that only one set of coordinates is allowed in a given HDU. Cartesian ITRS coordinates are the preferred coordinate system; however, when using these in an environment requiring nanosecond accuracy, one should take care to distinguish between meters consistent with TCG or with TT. If one uses geodetic coordinates, the geodetic altitude OBSGEO-H is measured with respect to the IAU 1976 ellipsoid which is defined as having a semi-major axis of 6 378 140 m and an inverse flattening of 298.2577.

A non-standard location indicated by CUSTOM *must* be specified in a manner similar to the specification of the observatory location (indicated by TOPOCENTER). One should be careful with the use of the CUSTOM value and not confuse it with TOPOCENTER, as use of the latter imparts additional information on the provenance of the data.

ITRS coordinates (X,Y,Z) may be derived from geodetic coordinates (L,B,H) through:

$$X = (N(B) + H)\cos(L)\cos(B)$$

$$Y = (N(B) + H)\sin(L)\cos(B)$$

$$Z = (N(B)(1 - e^2) + H)\sin(B)$$

where:

$$N(B) = \frac{a}{\sqrt{1 - e^2\sin^2(B)}}$$

$$e^2 = 2f - f^2$$

a is the semi-major axis, and f is the inverse of the inverse flattening. Nanosecond precision in timing requires that OBSGEO-[BLH] be expressed in a geodetic reference frame defined after 1984 in order to be sufficiently accurate.

9.2.4. Time reference direction

If any pathlength corrections have been applied to the time stamps (i.e., if the reference position is not TOPOCENTER for observational data), the reference direction that is used in calculating the pathlength delay *should* be provided in order to maintain a proper analysis trail of the data. However, this is useful only if there is also information available on the location from where the observation was made (the observatory location). The direction will usually be provided in a spatial coordinate frame that is already being used for the spatial metadata, although it is conceivable that multiple spatial frames are involved, e.g., spherical ICRS coordinates for celestial positions, and Cartesian FK5 for spacecraft ephemeris. The time reference direction does not by itself provide sufficient information to perform a fully correct transformation; however, within the context of a specific analysis environment it should suffice.

The uncertainty in the reference direction affects the errors in the time stamps. A typical example is provided by barycentric corrections where the time error is related to the position error:

$$t_{err}(\mathrm{ms}) \leq 2.4 pos_{err}(\mathrm{arcsec})$$

The reference direction is indicated through a reference to specific keywords. These keywords *may* hold the reference direction explicitly or (for data in BINTABLEs) indicate columns holding the coordinates. In event lists where the individual photons are tagged with a spatial position, those coordinates *may* have been used for the reference direction and the reference will point to the columns containing these coordinate values. The time reference direction *shall* be specified by the keyword:

TREFDIR – [character] The value field of this keyword *shall* contain a character string composed of: the name of the keyword containing the longitudinal coordinate, followed by a comma, followed by the name of the keyword containing the latitudinal coordinate. This reference direction *shall* apply to time coordinate axes in images as well.

In binary tables different columns *may* represent completely different Time Coordinate Frames. However, also in that situation the condition holds that each column can have only one Time Reference Direction. Hence, the following keyword may override TREFDIR:

TRDIR*n* – [character] The value field of this keyword *shall* contain a character string consisting of the name of the keyword or column containing the longitudinal coordinate, followed by a comma, followed by the name of the keyword or column containing the latitudinal coordinate. This reference direction *shall* apply to time coordinate axes in images as well.

9.2.5. Solar System Ephemeris

If applicable, the Solar System ephemeris used for calculating pathlength delays *should* be identified. This is particularly pertinent when the time scale is TCB or TDB. The ephemerides that are currently most often used are those from JPL (2014a,b).

The Solar System ephemeris used for the data (if required) *shall* be indicated by the following keyword:

PLEPHEM – [character; default: 'DE405'] The value field of this keyword *shall* contain a character string code for the name of the Solar System ephemeris, the only permitted values for which are given in Table 33.

Table 33: Valid solar system ephemerides

Value	Reference
'DE200'	Standish (1990); considered obsolete, but still in use
'DE405'	Standish (1998); default
'DE421'	Folkner, et al. (2009)
'DE430'	Folkner, et al. (2014)
'DE431'	Folkner, et al. (2014)
'DE432'	Folkner, et al. (2014)

Future ephemerides in this series *shall* be accepted and recognized as they are released.

9.3. Time unit

When recorded as a global keyword, the unit used to express time *shall* be specified by:

TIMEUNIT – [character; default: 's'] The value field of this keyword *shall* contain a character string that specifies the time unit; the value *should* be one of those given in Table 34. This time unit *shall* apply to all time instances and durations that do not have an implied time unit (such as is the case for JD, MJD, ISO-8601, J and B epochs). If this keyword is absent, 's' *shall* be assumed.

In an appropriate context, e.g., when an image has a time axis, TIMEUNIT *may* be overridden by the CUNIT*ia* keywords and their binary table equivalents (see Table 22).

The specification of the time unit allows the values defined in Greisen & Calabretta (2002), shown in Table 34, with the addition of the century. See also Sect. 4.3 for generalities about units.

The use of ta and Ba is not encouraged, but there are data and applications that require the use of tropical years or Besselian epochs (see Section 9.1.2). The length of the tropical year, ta, in days is:

$$1\,\mathrm{ta} = 365.24219040211236 - 0.00000615251349\,T - 6.0921\times10^{-10}\,T^2 + 2.6525\times10^{-10}\,T^3 \ \text{(d)}$$

Table 34: Recommended time units

Value	Definition
's'	second (default)
'd'	day (= 86,400 s)
'a'	(Julian) year (= 365.25 d)
'cy'	(Julian) century (= 100 a)
The following values are also acceptable:	
'min'	minute (= 60 s)
'h'	day (= 86,400 s)
'yr'	(Julian) year (= a = 365.25 d)
'ta'	tropical year
'Ba'	Besselian year

where T is in Julian centuries since J2000, using time scale TDB. The length of the Besselian year in days is:

$$1\mathrm{Ba} = 365.2421987817 - 0.00000785423\,T \ \text{(d)}$$

where T is in Julian centuries since J1900, using time scale ET, although for these purposes the difference with TDB is negligible.

Readers are cautioned that the subject of tropical and Besselian years presents a particular quandary for the specification of standards. The expressions presented here are the most accurate available, but are applicable for use when creating data files (which is strongly discouraged), rather than for interpreting existing data that are based upon these units. But there is no guarantee that the authors of the data applied these particular definitions. Users are therefore advised to pay close attention and attempt to ascertain what the authors of the data really used.

9.4. Time offset, binning, and errors

9.4.1. Time offset

A uniform clock correction may be applied in bulk with the following single keyword.

TIMEOFFS – [floating-point; default: 0.0] The value field of this keyword *shall* contain the value of the offset in time that *shall* be added to the reference time, given by one of: MJDREF, JDREF, or DATEREF.

The time offset may serve to set a zero-point offset to a relative time series, allowing zero-relative times, or just higher precision, in the time stamps. Its default value is zero. The value of this keyword affects the values of TSTART, and TSTOP, as well as any time pixel values in a binary table. However, this construct *may* only be used in tables and *must not* be used in images.

9.4.2. Time resolution and binning

The resolution of the time stamps (the width of the time sampling function) *shall* be specified by:

TIMEDEL – [floating-point] The value field of this keyword *shall* contain the value of the time resolution in the units of TIMEUNIT. This construct, when present, *shall only* be used in tables and *must not* be used in images.

In tables this may, for instance, be the size of the bins for time series data or the bit precision of the time stamp values.

When data are binned in time bins (or, as a special case, events are tagged with a time stamp of finite precision) it is important to know to the position within the bin (or pixel) to which the time stamp refers. Coordinate values normally correspond to the center of all pixels (see Sect. 8.2); yet clock readings are effectively truncations, not rounded values, and therefore correspond to the lower bound of the pixel.

TIMEPIXR – [floating-point; default: 0.5] The value field of this keyword *shall* contain the value of the position within the pixel, from 0.0 to 1.0, to which the time-stamp refers. This construct, when present, *shall only* be used in tables and *must not* be used in images.

A value of 0.0 may be more common in certain contexts, e.g. when truncated clock readings are recorded, as is the case for almost all event lists.

9.4.3. Time errors

The absolute time error is the equivalent of a systematic error, *shall* be given by the following keyword:

TIMSYER – [floating-point; default: 0.] The value field of this keyword *shall* contain the value of the absolute time error, in units of TIMESYS.

This keyword *may* be overridden, in appropriate context (e.g., time axes in image arrays or table columns; by the CSYER*ia* keywords and their binary table equivalents (see Table 22).

The relative time error specifies accuracy of the time stamps relative to each other. This error will usually be much smaller than the absolute time error. This error is equivalent to a random error, and *shall* be given by the following keyword:

TIMRDER – [floating-point; default: 0.] The value field of this keyword *shall* contain the value of the relative time error, i.e. the random error between time stamps, in units of TIMESYS.

This keyword *may* be overridden, in appropriate context (e.g., time axes in image arrays or table columns; by the CRDER*ia* keywords and their binary table equivalents (see Table 22).

9.5. Global time keywords

The time keywords in Table 35 are likely to occur in headers even when there are no time axes in the data. Except for DATE, they provide the top-level temporal bounds of the data in the HDU. As noted before, they may also be implemented as table columns. Keywords not previously described are defined below; all are included in the summary Table 22.

DATE-BEG – [datetime] The value field of this keyword *shall* contain a character string in ISO-8601 format that specifies the start time of data acquisition in the time system specified by the TIMESYS keyword.

DATE-END – [datetime] The value field of this keyword *shall* contain a character string in ISO-8601 format that specifies the stop time of data acquisition in the time system specified by the TIMESYS keyword.

MJD-BEG – [floating-point] The value field of this keyword *shall* contain the value of the MJD start time of data acquisition in the time system specified by the TIMESYS keyword.

Table 35: Keywords for global time values

Keyword	Notes
DATE	Defined in Sect. 4.4.2
DATE-OBS	Defined in Sect. 4.4.2. Keyword value was not restricted to mean the start time of an observation, and has historically also been used to indicate some form of mean observing date and time. To avoid ambiguity use DATE-BEG instead.
DATE-BEG	Defined in this section.
DATE-AVG	Defined in Sect. 8.4.1. The method by which average times should be calculated is not defined by this Standard.
DATE-END	Defined in this section.
MJD-OBS	Defined in Sect. 8.3
MJD-BEG	Defined in this section.
MJD-AVG	Defined in Sect. 8.4.1. The method by which average times should be calculated is not defined by this Standard.
MJD-END	Defined in this section.
TSTART	Defined in this section.
TSTOP	Defined in this section.

MJD-END – [floating-point] The value field of this keyword *shall* contain the value of the MJD stop time of data acquisition in the time system specified by the TIMESYS keyword.

TSTART – [floating-point] The value field of this keyword *shall* contain the value of the start time of data acquisition in units of TIMEUNIT, relative to MJDREF, JDREF, or DATEREF and TIMEOFFS, in the time system specified by the TIMESYS keyword.

TSTOP – [floating-point] The value field of this keyword *shall* contain the value of the stop time of data acquisition in units of TIMEUNIT, relative to MJDREF, JDREF, or DATEREF and TIMEOFFS, in the time system specified by the TIMESYS keyword.

The alternate-axis equivalent keywords for BINTABLEs, DOBS*n*, MJDOB*n*, DAVG*n*, and MJDA*n*, as defined in Table 22, are also allowed. Note that of the above only TSTART and TSTOP are relative to the time reference value. As in the case of the time reference value (see Section 9.2.2), the JD values supersede DATE values, and MJD values supersede both, in cases where conflicting values are present.

It should be noted that, although they do not represent global time values within an HDU, the CRVAL*ia* and CDELT*ia* keywords, and their binary table equivalents (see Table 22), also represent (binary) time values. They should be handled with the same care regarding precision when combining them with the time reference value as any other time value.

Finally, Julian and Besselian epochs (see Sections 9.1.2 and 9.3) *may* be expressed by these two keywords – to be used with great caution, as their definitions are more complicated and hence their use more prone to confusion:

JEPOCH – [floating-point] The value field of this keyword *shall* contain the value of the Julian epoch, with an implied time scale of 'TDB'.

BEPOCH – [floating-point] The value field of this keyword *shall* contain the value of the Besselian epoch, with an implied time scale of 'ET'.

When these epochs are used as time stamps in a table column their interpretation will be clear from the context. When the key-

words appear in the header without obvious context, they *must* be regarded as equivalents of DATE-OBS and MJD-OBS, i.e., with no fixed definition as to what part of the dataset they refer to.

9.6. Other time coordinate axes

There are a few coordinate axes that are related to time and that are accommodated in this standard: (temporal) *phase*, *timelag*, and *frequency*. Phase results from folding a time series on a given period, and can appear in parallel with *time* as an alternate description of the same axis. Timelag is the coordinate of cross- and auto-correlation spectra. The temporal *frequency* is the Fourier transform equivalent of time and, particularly, the coordinate axis of power spectra; spectra where the dependent variable is the electromagnetic field are excluded here, but see Greisen et al. (2006). These coordinate axes *shall* be specified by giving CTYPE*i* and its binary table equivalents one of the values: PHASE, TIMELAG, or FREQUENCY.

Timelag units are the regular time units, and the basic unit for frequency is Hz. Neither of these two coordinates is a linear or scaled transformation of time, and therefore cannot appear in parallel with time as an alternate description. That is, a given vector of values for an observable can be paired with a coordinate vector of time, or timelag, or frequency, but not with more than one of these; the three coordinates are orthogonal.

Phase can appear in parallel with time as an alternate description of the same axis. Phase *shall* be recorded in the following keywords:

CZPHS*ia* – [floating-point] The value field of this keyword *shall* contain the value of the time at the zero point of a phase axis. Its units *may* be deg, rad, or turn.

CPER*ia* – [floating-point] The value field of this keyword, if present *shall* contain the value of the period of a phase axis. This keyword can be used only if the period is a constant; if that is not the case, this keyword *should* either be absent or set to zero.

CZPHS*ia* may instead appear in binary table forms TCZPH*n*, TCZP*na*, *i*CZPH*n*, and *i*CZP*na*. CPER*ia* may instead appear in binary table forms TCPER*n*, TCPR*na*, *i*CPER*n*, and *i*CPR*na*. The Phase, period and zero point *shall* be expressed in the globally valid time reference frame and unit as defined by the global keywords (or their defaults) in the header.

9.7. Durations

There is an extensive collection of header keywords that indicate time durations, such as exposure times, but there are many pitfalls and subtleties that make this seemingly simple concept treacherous. Because of their crucial role and common use, keywords are defined below to record exposure and elapsed time.

XPOSURE – [floating-point] The value field of this keyword *shall* contain the value for the effective exposure duration for the data, corrected for dead time and lost time in the units of TIMEUNIT. If the HDU contains multiple time slices, this value *shall* be the total accumulated exposure time over all slices.

TELAPSE – [floating-point] The value field of this keyword *shall* contain the value for the amount of time elapsed, in the units of TIMEUNIT, between the start and the end of the observation or data stream.

Durations *must not* be expressed in ISO-8601 format, but only as actual durations (i.e., numerical values) in the units of the specified time unit.

Good-Time-Interval (GTI) tables are common for exposures with gaps in them, particularly photon-event files, as they make it possible to distinguish time intervals with "no signal detected" from "no data taken." GTI tables in BINTABLE extensions *must* contain two mandatory columns, START and STOP, and *may* contain one optional column, WEIGHT. The first two define the interval, the third, with a value between 0 and 1, the quality of the interval; *i.e.*, a weight of 0 indicates a *Bad*-Time-Interval. WEIGHT has a default value of 1. Any time interval not covered in the table shall be considered to have a weight of zero.

9.8. Recommended best practices

The following guidelines should be helpful in creating data products with a complete and correct time representation.

- The presence of the informational DATE keyword is *strongly recommended* in all HDUs.
- One or more of the informational keywords DATE-xxxx and/or MJD-xxxx *should* be present in all HDUs whenever a meaningful value can be determined. This also applies, e.g., to catalogs derived from data collected over a well-defined time range.
- The global keyword TIMESYS is *strongly recommended*.
- The global keywords MJDREF or JDREF or DATEREF are *recommended*.
- The remaining informational and global keywords *should* be present whenever applicable.
- All context-specific keywords *shall* be present as needed and required by the context of the data.

9.8.1. Global keywords and overrides

For reference to the keywords that are discussed here, see Table 22. The globally applicable keywords listed in section B of the table serve as default values for the corresponding C* and TC* keywords in that same section, but only when axis and column specifications (including alternate coordinate definitions) use a time scale listed in Table 30 or when the corresponding CTYPE or TTYPE keywords are set to the value 'TIME'. Any alternate coordinate specified in a non-recognized time scale assumes the value of the axis pixels or the column cells, optionally modified by applicable scaling and/or reference value keywords; see also Section 9.2.1.

9.8.2. Restrictions on alternate descriptions

An image will have at most one time axis as identified by having the CTYPE*i* value of TIME or one of the values listed in Table 30. Consequently, as long as the axis is identified through CTYPE*i*, there is no need to have axis number identification on the global time-related keywords. It is expressly prohibited to specify more than one time reference position on this axis for alternate time coordinate frames, since this would give rise to complicated model-dependent non-linear relations between

these frames. Hence, time scales TDB and TCB (or ET, to its precision) may be specified in the same image, but cannot be combined with any of the first nine time scales in Table 30; those first nine can be expressed as linear transformations of each other, too, provided the reference position remains unchanged. Time scale LOCAL is by itself, intended for simulations, and should not be mixed with any of the others.

9.8.3. Image time axes

Section 8.2 requires keywords CRVAL*ia* to be numeric and they cannot be expressed in ISO-8601 format. Therefore it is *required* that CRVAL*ia* contain the elapsed time in units of TIMEUNIT or CUNIT*ia*, even if the zero point of time is specified by DATEREF. If the image does not use a matrix for scaling, rotation and shear (Greisen & Calabretta 2002), CDELT*ia* provides the numeric value for the time interval. If the PC form of scaling, rotation and shear (Greisen & Calabretta 2002) is used, CDELT*ia* provides the numeric value for the time interval, and PC*i_j*, where $i = j$ = the index of the time axis (in the typical case of an image cube with axis 3 being time, $i = j = 3$) would take the exact value 1, the default (Greisen & Calabretta 2002). When the CD*i_j* form of mapping is used, CD*i_j* provides the numeric value for the time interval. If one of the axes is time and the matrix form is used, then the treatment of the PC*i_ja* (or CD*i_ja*) matrices involves at least a Minkowsky metric and Lorentz transformations (as contrasted with Euclidean and Galilean).

digit :=
‘0’–‘9’

floating_value :=
decimal_number [exponent]
decimal_number :=
[sign] [integer_part] [‘.’ [fraction_part]]
{Constraint: At least one of the integer_part and fraction_part *must* be present.}

integer_part :=
digit | [digit...]

fraction_part :=
digit | [digit...]

exponent :=
exponent_letter [sign] digit [digit...]

exponent_letter :=
‘E’ | ‘D’

complex_integer_value :=
‘(’ [space...] real_integer_part [space...] ‘,’ [space...]
imaginary_integer_part [space...] ‘)’

real_integer_part :=
integer_value

imaginary_integer_part :=
integer_value

complex_floating_value :=
‘(’ [space...] real_floating_part [space...] ‘,’ [space...]
imaginary_floating_part [space...] ‘)’

real_floating_part :=
floating_value

imaginary_floating_part :=
floating_value

Appendix B: Suggested time scale specification

The content of this Appendix has been superseded by Section 9 of the formal Standard, which derives from Rots et al. (2015).

References

Note: Many of these *FITS* references are available electronically from the NASA Astrophysics Data System (ADS) and/or the *FITS* Support Office web sites at
`http://adswww.harvard.edu` and
`http://fits.gsfc.nasa.gov/fits_documentation.html`.

Allen, S. & Wells, D. 2005, IETF RFC 4047, `http://www.ietf.org/rfc/rfc4047.txt`

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

Bradner, S. 1997, IETF RFC 2119, `http://www.ietf.org/rfc/rfc2119.txt`

Bunclark, P. & Rots, A. 1997, *Precise re-definition of* `DATE-OBS` *Keyword encompassing the millennium*, `http://fits.gsfc.nasa.gov/year2000.html`

Calabretta, M. R. & Greisen, E. W. 2002, A&A, 395, 1077

Calabretta, M. R. & Roukema, B. F. 2007, MNRAS, 381, 865

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

Cotton, W. D., et al. 1990, *Going AIPS: A Programmer's Guide to the NRAO Astronomical Image Processing System*, Charlottesville: NRAO

Folkner, W. M., Williams, J. G., & Boggs, D. H. 2009, Interplanetary Network Progress Report 42-178, available online: `http://tmo.jpl.nasa.gov/progress_report/42-178/178C.pdf`

Folkner, W. M. et al. 2014, Interplanetary Network Progress Report 42-196, available online: `http://ipnpr.jpl.nasa.gov/progress_report/42-196/196C.pdf`

Greisen, E. W. & Calabretta, M. R. 2002, A&A, 395, 1061

Greisen, E. W., Calabretta, M. R., Valdes, F. G., & Allen, S. L. 2006, A&A, 446, 747

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

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

Grosbøl, P. & Wells, D. C. 1994, *Blocking of Fixed-block Sequential Media and Bitstream Devices*, `http://fits.gsfc.nasa.gov/blocking94.html`

Hanisch, R., et al. 2001, A&A, 376, 359

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

IAU 1983, *Transactions of the IAU*, XVIIIB, 45

IAU 1988, *Transactions of the IAU*, XXB, 51

IAU 1997, Resolution B1 of the XXIIIrd General Assembly – Transactions of the IAU Vol. XXIII B, Ed. J. Andersen, (Dordrecht: Kluwer). Available online: http://www.iau.org/static/resolutions/IAU1997_French.pdf

IEEE 1985, *American National Standard – IEEE Standard for Binary Floating Point Arithmetic*, ANSI/IEEE 754–1985, New York: American National Standards Institute, Inc.

Irwin, A. W. & Fukushima, T. A. 1999, A&A348, 642

ISO 2004, *Information technology – Programming languages – Fortran*, ISO/IEC 1539-1:2004, Geneva: International Organization for Standardization

ISO 2004b, International Standard ISO 8601:2004(E), *Data elements and interchange formats – Information interchange – Representation of dates and times*

NASA/JPL Planetary Ephemerides 2014a, available online: `http://ssd.jpl.nasa.gov/?ephemerides`

NASA/JPL Solar and Planetary Ephemerides 2014b, available online: `http://ssd.jpl.nasa.gov/?planet_eph_export`

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

Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. 2010, A&A, 524, A42

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

Rots, A. H., Bunclark, P. S., Calabretta, M. R., Allen, S. L., Manchester, R. N. & Thompson, W. T. 2015, A&A, 574, A36

Schmitz, M., et al. 1995, *Information & On-line data in Astronomy*, eds. D. Egret & M. A. Albrecht (Kluwer Academic Pub.), 259

Standish, E. M. 1990, A&A, 233, 252

Standish, E. M. 1998, JPL Memo IOM 312.F-98-048

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

Wells, D. C. & Grosbøl, P. 1990, *Floating Point Agreement for* FITS, `http://fits.gsfc.nasa.gov/fp89.txt`

Index