Floating Point Agreement for FITS

Don Wells, Preben Grosbøl

22 December 1989

1 The Agreement

The Basic FITS, Random Groups and Generalized Extensions Agreements are revised to add IEEE-754 32- and 64-bit floating point numbers to the original set of FITS data types. `BITPIX=-32` and `BITPIX=-64` signify 32- and 64-bit IEEE floating point numbers; the absolute value of `BITPIX` is used for computing the sizes of data structures. The full IEEE set of number forms are allowed for FITS interchange, including all special values (e.g., the 'not-a-number' cases). The order of the bytes will be sign and exponent first, followed by the mantissa bytes in order of decreasing significance. The `BLANK` keyword will be ignored by FITS readers when `BITPIX=-32` or `-64`.

2 Discussion

The dynamic range of astronomical data has increased steadily in the years since the original FITS Agreement (March 1979) which defined 8-, 16- and 32-bit integers as the only data formats. All integer formats are limited by an absolute error, whereas floating point formats can span a much wider numerical range with only a relative error. Floating point formats are desirable for the applications that need increased dynamic range. Many astronomical data systems now use floating point in data processing, and conversion to and from FITS integer formats sacrifice accuracy and dynamic range and consume significant computer time. Almost all of the new computers designed since about 1981 use the IEEE format (see references below). We are proposing that FITS be revised to allow transmission of 32- and 64-bit floating point data within the FITS format using the IEEE standard. This Floating Point Agreement will also apply to the Random Groups Extension and to those Generalized Extensions for which `BITPIX` is not explicitly restricted (e.g., `BITPIX=8` for `XTENSION='TABLE'`).

In order to transmit floating point data in the main data matrix we must add some new convention to the main FITS header. We propose to add two new allowed values for the basic keyword `BITPIX`. The set of values specified in the original FITS Agreement was 8, 16 and 32; we propose to add -32 and -64 to indicate IEEE single- and double-precision floating point data. Thus, the number of bits per pixel would be the absolute value of `BITPIX`. This implies that the number of 2880-byte logical records used by the main data matrix must be computed using the absolute value of `BITPIX` (this rule was specified in the Generalized Extensions Agreement (Grosbøl etal. 1988)).

The IEEE standard specifies a variety of special exponent and mantissa values in order to support the concepts of plus and minus infinity, plus and minus zero, 'denormalized' numbers and 'not-a-number' (NaN). We propose that all of these special cases should be fully accepted for FITS interchange.

We propose that the order of the bytes should be sign and exponent first, followed by the mantissa bytes in order of decreasing significance (i.e., the standard non-byte-swapped order).

The `BLANK` keyword of the original FITS Agreement should be ignored by FITS readers when BITPIX=-32 or -64 (the NaNs of the IEEE format will act as the blank). FITS writers should not write the `BLANK` keyword if `BITPIX=-32` or `-64`.

The `BSCALE` and `BZERO` values should be applied by FITS readers if they differ from 1.0 and 0.0. However, such usage will be regarded as bad practice and will be strongly discouraged because of the risk of generating overflows and underflows in FITS readers.

3 Bibliography

COMPUTER Magazine, Vol. 13, No. 1, p.68, January 1980.

COMPUTER Magazine, Vol. 14, No. 3, p. 51, March 1981.

ANSI/IEEE 754-1985, "IEEE Standard for Binary Floating Point Arithmetic".

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

Greisen, E.W., Harten, R.H.: 1981, Astron. Astrophys. Suppl. Ser. 44, 371, "An Extension of FITS for Groups of Small Arrays of Dat".

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

Harten, R.H., Grosbøl, P., Greisen, E.W., Wells, D.C.: 1988, Astron. Astrophys. Suppl. Ser. 73, 365, "The FITS Tables Extension".

4 Implementation Notes

Note-1: Many data analysis systems today use floating point internally and so conversion costs for FITS input and output will be reduced by this proposal. For machines with IEEE hardware the conversion cost will be nearly zero, especially when `BZERO=0` and `BSCALE=1` and the reader has two DO-loops (one with and one without scaling).

Note-2: The interpretation of the IEEE sign, exponent and mantissa bit fields is precisely specified by the five rules of Section 3.2.1 of ANSI/IEEE Standard 754-1985. Algorithms for transformation between IEEE and non-IEEE formats can be derived from these rules. The FITS Committees will provide a central archive for conversion software for various architectures.

FITS readers on non-IEEE systems should test the IEEE exponents for the special values 0 and 255 / 2047 (for 32 / 64-bit numbers). Exponent zero should be mapped to the local value for floating zero (exponent zero is used by IEEE for 'denormalized' numbers as well as for both plus and minus zero); exponent 255 / 2047 should be mapped to the local value used for blank pixels (all IEEE infinity and NaN cases have exponent 255 / 2047). The IEEE exponent field can be selected from the 32 / 64-bit values by masking with hexadecimal byte string `7F,80,00,00 / 7F,F0,00,00,00,00,00,00` (i.e., the constant `7F800000 / 7FF0000000000000` hexadecimal on computers with the normal word ordering; for VAXes use constant `00007F80` / `0000000000007FF0` instead). The two tests are made by comparing the result against zero and against the mask.

FITS writers executing on non-IEEE systems should map their local value for blank pixels to an IEEE NaN value; we suggest that all bits set (minus one in 32- and 64-bit two-s complement integers) is a good choice. The FITS writer should also map local floating point absolute values which exceed the largest value of IEEE (approximately 10^{38} for 32-bit, 10^{304} for 64-bit) to IEEE-Infinity. We recommend that absolute values smaller than the smallest normalized IEEE value (approximately 10^{-38} or 10^{-304}) be mapped to zero. These two tests are particularly important on machines which use 'hexadecimal' normalization, with an exponent range of approximately 10^{75} in single-precision.

The DEC VAX architecture is an important special case. VAX systems use the DEC-proprietary 'F' format for 32-bit floating data and their byte order is 'swapped'. Once the swap has been corrected the F-format differs from the IEEE 32-bit format only in that the exponent bias differs by one and the (hidden) binary point is shifted by one bit. The two differences add, and the total effect is that the two number systems differ by a factor of 4.0 ($VAXF = 4.0 \times IEEE32$, $IEEE32 = 0.25 \times VAXF$). This floating point multiplication operation is little, if any, more expensive than the conversion operation between integer and floating point which is required by the original FITS Agreement. VAX systems should test the exponent field on input to detect the infinity and NaN cases and map them to the local blank code using the test specified previously. The minus zero and denormalized cases may also need to be trapped. Initial prototype implementations of this Agreement have disclosed that some IEEE hardware produces -0.0 results and that reserved operand errors occur in VAXes when these values are used as operands. The 32-bit operand should be compared against 32768 to detect this case, and the result 0.0 be given rather than multiplication by 4.0.

The DEC VAX architecture supports two different double precision formats, 'D' and 'G'. The 'G' format has the same relationship to IEEE 64-bit format that the DEC 'F' format has to IEEE 32-bit ($VAXG = 4.0 \times IEEE64$). The 'D' format, however, requires a complicated transformation to convert to/from IEEE 64-bit; programmers should consult archives of FITS code to obtain suitable algorithms and implementations for this case.