Proposed changes to the FITS Standard to support 64-bit integers

Revision date: 7 November 2005

Change 1: Add a line to table 5.2 as shown:

Value	Data Represented
8	Character or unsigned binary integer
16	16-bit twos-complement binary integer
32	32-bit twos-complement binary integer
64	64-bit twos-complement binary integer
-32	IEEE single precision floating point
-64	IEEE double precision floating point

Change 2: Add new section 6.2.4

> 6.2.4 Sixty-four-bit
>
> Sixty-four-bit integers shall be twos-complement signed binary integers, contained in eight bytes.

Change 3: Renumber existing section 6.2.4 to 6.2.5 and add the following sentence at the end of the paragraph:

> Unsigned sixty-four-bit integers can be represented in FITS files by applying an offset of 9223372036854775808 (2^{63}) to the data values.

All references to section 6.2.4 elsewhere in the document will be changed to 6.2.5.

Change 4: Add 2 lines to table 8.5 as shown:

TFORMn value	Description	8-bit Bytes
L	Logical	1
X	Bit	*
B	Unsigned byte	1
I	16-bit integer	2
J	32-bit integer	4
K	64-bit integer	8
A	Character	1
E	Single precision floating point	4
D	Double precision floating point	8
C	Single precision complex	8
M	Double precision complex	16
P	Array Descriptor (32-bit)	8
Q	Array Descriptor (64-bit)	16

Change 5: In section 8.3.1, in the description of the TFORMn keyword modify the following sentence as shown:

> For fields of type P or Q, the only permitted repeat counts are 0 and 1.

Change 6: In section 8.3.2, in the description of the TNULLn, TSCALn, TZEROn, TDISPn, and TDIMn keywords, there are 5 instances where the 'P' TFORMn type is specifically mentioned. Modify these 5 sentences so that their meaning applies to the 'Q' TFORMn type as well as the 'P' TFORMn type.

Change 7: Insert a new paragraph in section 8.3.3.1, following the paragraph that begins with "32-Bit Integer"

> 64-Bit Integer If the value of the TFORMn keyword specifies data type K, the data in field n shall consist of twos-complement signed 64-bit integers, contained in eight bytes. The most significant byte shall be first, and subsequent bytes shall be in order of decreasing significance. Within each byte, the most significant bit shall be first, and subsequent bits shall be in order of decreasing significance. Null values are given by the value of the associated TNULLn keyword. Unsigned integers can be represented using the convention described in Section 6.2.5.

Change 8: Insert a sentence into the last paragraph of section 8.3.3.1, as shown:

> Array Descriptor If the value of the TFORMn keyword specifies data type P, the data in field n shall consist of not more than one pair of 32-bit integers. If the value of the TFORMn keyword specifies data type Q, the data in field n shall consist of not more than one pair of 64-bit integers. The meaning of these integers is defined in section 8.3.5.

Change 9: In section 8.3.5, modify the 2nd sentence in the 3rd paragraph as follows:

> Since a field of any datatype can be a static array, a field of any datatype can also be a variable-length array (excluding the type P or type Q ~~the~~ variable-length array descriptor itself, which is not a datatype so much as a storage class specifier).

Change 10: In section 8.3.5, modify the 4th paragraph as follows:

> A variable-length array is declared in the table header with a special field datatype specifier of the form
>
> `rPt(`e_{max}`)  or  rQt(`e_{max}`)`
>
> where the "P" or "Q" indicates the presence of an array descriptor (described below) ~~amount of space occupied by the array descriptor in the data record (64 bits)~~, the element count r should be 0, 1, or absent, t is a character denoting the datatype of the array data (L, X, B, I, J, K, etc., but not P or Q), and e_{max} is a quantity guaranteed to be equal to or greater than the maximum number of elements of type t actually stored in a table record. There is no built-in upper limit on the size of a stored array (other than the fundamental limit imposed by the range of the ~~32-bit~~ array descriptor, defined below);

Change 11: In section 8.3.5, modify the 6th paragraph as follows:

> The data for the variable-length arrays in a table are not stored in the actual data records; they are stored in a special data area, the heap, following the last fixed size data record. What is stored in the data record is an *array descriptor*. This consists of two 32-bit signed integer values in the case of "P" array descriptors, or two 64-bit signed integer values in the case of "Q" array descriptors: the number of elements (array length) of the stored array, followed by the zero-indexed byte offset of the first element of the array, measured from the start of the heap area. The meaning of a negative value for either of these integers is not defined by this standard.