IEEE floating point format

IEEE 754-1990 standard for binary floating point arithmetic defined what is commonly referred to as “IEEE floating point”. MIMOSA utilizes the 32-bit IEEE floating point format:

N = 1.F* 2^(E-127) (N = floating point number, F = fractional part in binary notation, E = exponent in bias 127 representation)

In 32 bit IEEE format, 1 bit is allocated as the sign bit, the next 8 bits are allocated as the exponent field, and the last 23 bits are the fractional parts of the normalized number.

Sign	**Exponent**	**Fraction**
0	00000000	00000000000000000000000
Bit 31	**[30 -- 23]**	**[22 -- 0]**

A sign bit of 0 indicates a positive number, and a 1 is negative. The exponent field is represented by "excess 127 notation". The 23 fraction bits actually represent 24 bits of precision, as a leading 1 in front of the decimal point is implied.

There are some exceptions.

E = 255; F = 0;
+/- infinity

E = 255; F != 0;
NaN, Not a number. Overflow, error.....

E = 0; F = 0;
0

E = 0; F != 0;
denormalized, tiny number, smaller than smallest allowed.

With exponent field 00000000 and 11111111 now reserved, the range is restricted to 2 ** -126 to 2 ** 127.

IEEE floating point format example

Suppose that we want to convert 9 97/128 into IEEE 32 bit format.
Here is what we do.

1) Convert to base 2:
   1001.1100001

2) Shift number to the form of 1.FFFFFF * 2**E:
   1.0011100001 * 2**3

3) Add 127 (excess 127 code) to exponent field, convert to binary:
   3+127 = 130 = 10000010

4) Determine the sign bit. If a negative number, set to 1. Otherwise set to 0.

4) Now put the numbers together, using only the fractional part of the number represented by step 2 above (remove the "1." preceding the fractional part):
   0 10000010 00111000010000000000000
   in hex representation, this is
   0100 0001 0001 1100 0010 0000 0000 0000
   or
   411C2000 in Hex format

5) Finally, MIMOSA asks for the low-order byte first:
   00201C41