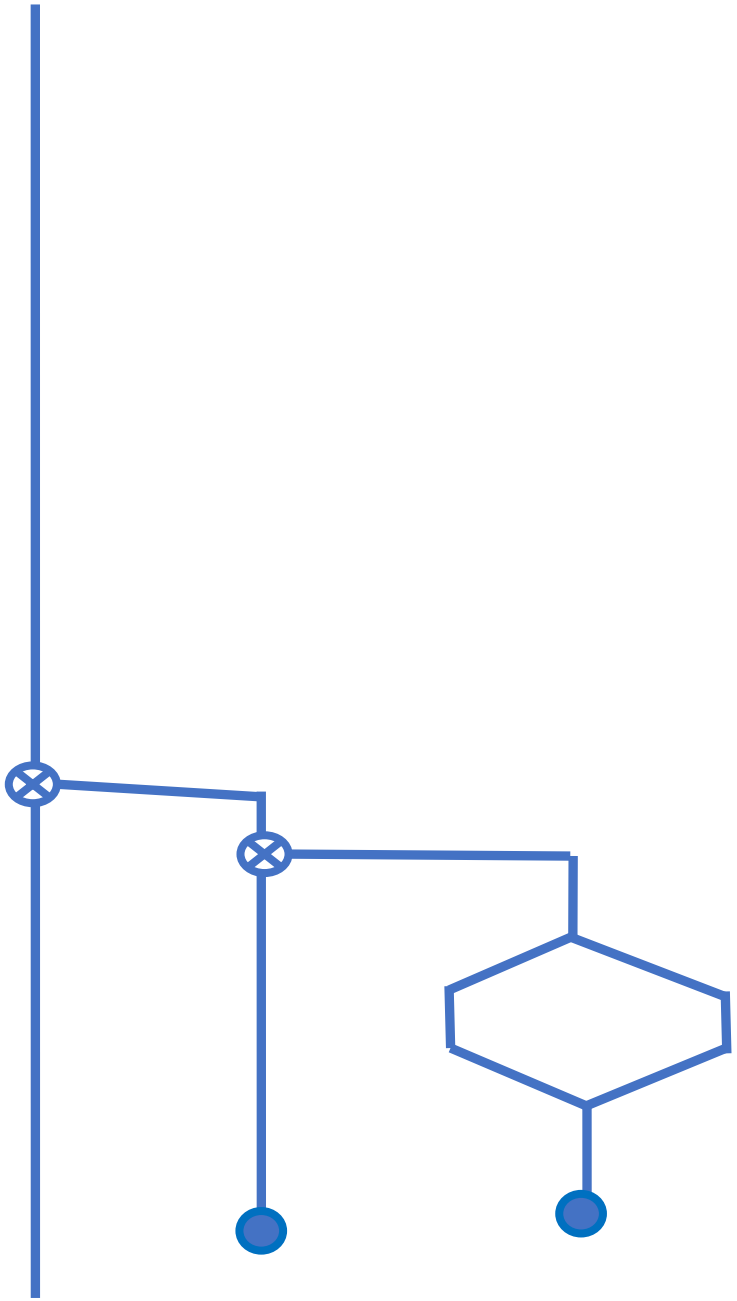


Messing With Floats

SVFIG
April 26, 2025
Bill Ragsdale

The Problem

The financial community often uses a securities file format using a very old MicroSoft Basic floating-point format from the 1980s.

	Date	High	Low	Close	Volume
2044	2/28/2025	6,043.40	5,837.66	5,954.50	130
2045	3/7/2025	5,985.99	5,666.29	5,770.20	134
2046	3/14/2025	5,705.37	5,504.65	5,638.94	137
2047	3/21/2025	5,715.33	5,597.76	5,667.56	137
2048	3/28/2025	5,786.90	5,572.42	5,580.94	139
2049	4/4/2025	5,695.31	5,069.89	5,074.08	153
2050	4/11/2025	5,481.34	4,835.27	5,363.36	145
2051	4/17/2025	5,459.46	5,220.79	5,282.70	147
2052					

MS Floating Point

It is 32 bit. An 8-bit exponent, 1 bit sign and 23 bit significand.

At that time, the significand was referred to as mantissa, a holdover from logarithms.

Details

- The significand is left justified with a ‘one’ leftmost in what would be bit 23.
- When left justified that ‘bit 23’ is omitted and implied. The sign appears in that bit position.
- ‘Zero’ has no implied bit, therefore zero is defined as a zero exponent, sign and significand. Zero is zero.

Testing Word, binary print

```
: B. ( n -- ) \ A diagnostic binary print.
  cr ."         3  2    2    2    1    1    0    0    0 "
  cr ."         1  8    4 x  0    6    2    8    4    0 “
base @ binary    cr 6 spaces
swap s>d  <#  8 0 do  # # # # bl hold loop #> type
base !   ;

      3  2    2    2    1    1    0    0    0
      1  8    4 x  0    6    2    8    4    0
      1000 0011 1110 0000 0000 0000 0000 0000
```

The General Representation

(sign) significand * $2^{(\text{exponent}+\text{offset})}$

```
Bit
3       2     2  2      1     1        0     0        0
1       4     3  2      6     5        8     7        0
EEEEEEEE      S  MMMMMMMM     MMMMMMMMM     MMMMMMMMM
                 ^Hidden bit to left of bit 22
E is 8 bit exponent with offset.
S is sign bit
M is 23 bit significand.
```

An Example, MSFLOAT

```
-7 S>MS  B.   ( convert and display )

  3  2    2    2    1    1    0    0    0
  1  8    4 x  0    6    2    8    4    0
 1000 0011 1110 0000 0000 0000 0000 0000
```

Conversion, single to MS

```
: S>MS  ( n1 --- n2 )
   dup 0= if exit then                          \ zero yields zero
   dup 0< if abs 0x00800000 else 0 then >r      \ strip sign
   0x98000000      \ proto-exponent with offset
   swap            \ proto-significand
   23 0 do dup  0x00800000  and                 \ test most sig bit
         if leave
           else 1 lshift                        \ for the significand
                swap -0x01000000 + swap         \ decrement exponent
          then loop
    0x007fffff  and                             \ mask away hidden bit
    or  r> or ;                                 \ or the three components
```

An Example, MSFLOAT

```
-7 S>MS  B.   ( convert and display )

  3  2    2    2    1    1    0    0    0
  1  8    4 x  0    6    2    8    4    0
 1000 0011 1110 0000 0000 0000 0000 0000 0000
```

IEEE>MS

```
: IEEE>MS ( f: f -- ) ( -- ms_float ) \ IEEE to 32 bit MS
   fdup f0= if fdrop 0 exit then      \ force a zero
   FS>DS  9 rshift                    \ transfer double to stack
   swap  dup 0x7FF00000 and           \ get 11 bit exponent
   20 rshift 1023 - 129 +             \ translate the exponent
   24 lshift                        \ position the exponent
   over 0x80000000 and              \ extract sign bit
   8 rshift or                      \ position prototype add sign
   swap 0x000FFFFF and              \ select low bits
   3 lshift  or or     ;            \ combine components
```

Debug Example IEEE > MS

IEEE to MS, Lower 32 bits

```
        3   2     2     2     1     1     0     0     0
        1   8     4 x   0     6     2     8     4     0
        0000  0000  0000  0000  0000  0000  0000  0000
```

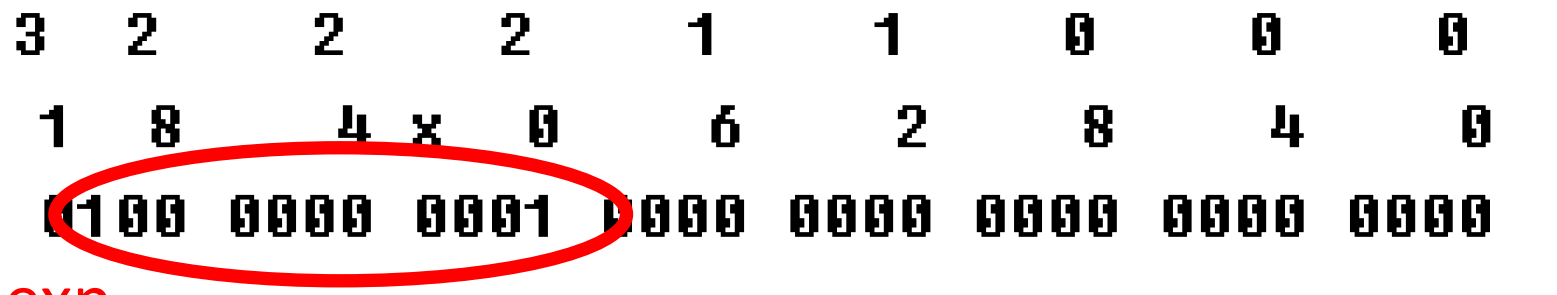
Now shifted

```
        3   2     2     2     1     1     0     0     0
        1   8     4 x   0     6     2     8     4     0
        0000  0000  0000  0000  0000  0000  0000  0000
```

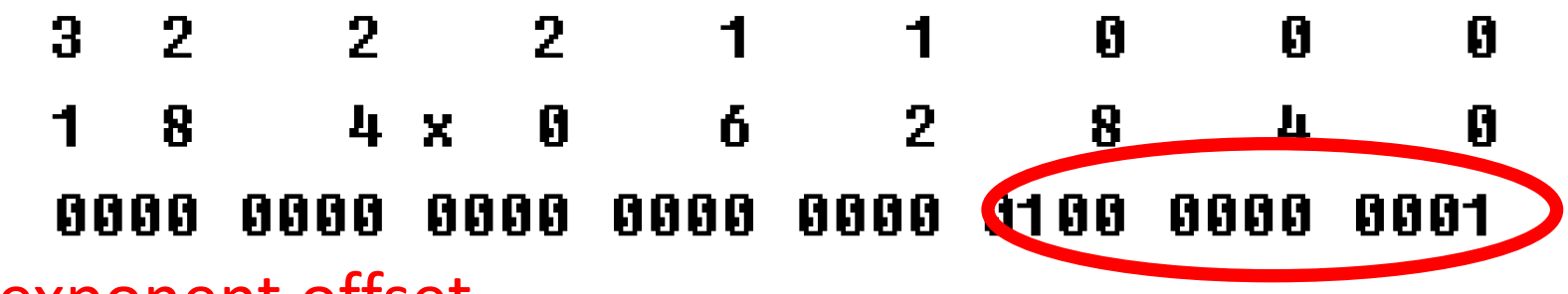
Raw high 32 bits

```
        3   2     2     2     1     1     0     0     0
        1   8     4 x   0     6     2     8     4     0
        1100  0000  0001  1100  0000  0000  0000  0000
```

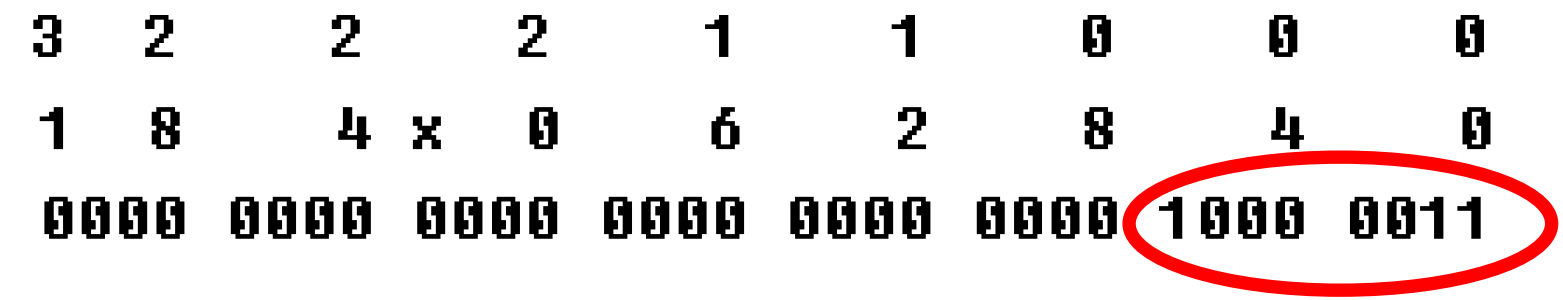
Exponent



Shifted exp



Adjust exponent offset



Exponent placed

```
3  2     2     2     1     1     0     0     0
1  8     4  x  0     6     2     8     4     0
1000  0011  0000  0000  0000  0000  0000  0000
```

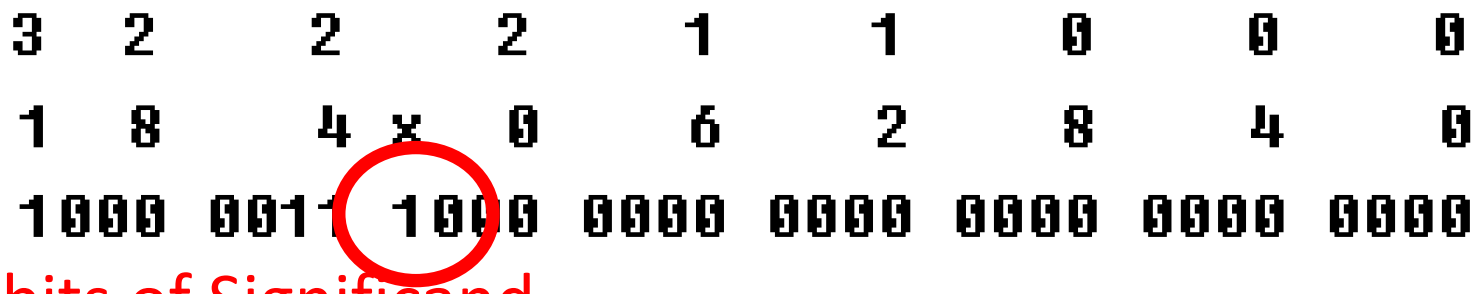
Want exponent

```
3  2     2     2     1     1     0     0     0
1  8     4  x  0     6     2     8     4     0
1000  0100  0111  0000  0000  0000  0000  0000
```

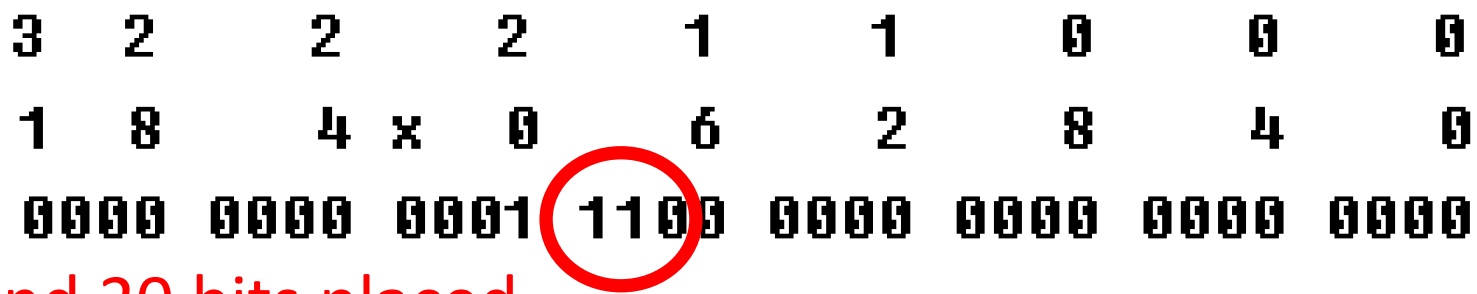
Sign bit

```
3  2     2     2     1     1     0     0     0
1  8     4  x  0     6     2     8     4     0
1000  0000  0000  0000  0000  0000  0000  0000
```

Sign positioned



High 20 bits of Significand



Significand 20 bits placed

Final MSFLOAT format

```
        3   2     2     2     1     1     0     0     0
        1   8     4 x   0     6     2     8     4     0
        1000 0011 1110 0000 0000 0000 0000 0000 0000
```

-7.000000 ok

MS to IEEE

Note: Lots of bit fiddling done in floating point!

```
: MS>IEEE  ( n -- ) ( fs: -- f )     \ MicroSoft to IEEE
  dup 0= if s>f exit then            \ zero yields zero
  dup
  0x007FFFFF and 0x00800000 or s>f    \ restore hidden bit
  8388608e f/                         \ 23 bit right shift
  dup 0xFF000000 and 24 rshift 129 -  \ remove exp. offset
  2e s>f f** f*                  \ scale by the float exponent.
  0x00800000 and if fnegate then  ;   \ adjust sign
```

Examples

```
      0e   IEEE>MS   MS>IEEE   F. .0000000
     -0e   IEEE>MS   MS>IEEE   F.  .0000000
       0   S>MS      MS>IEEE   F.  .0000000
-1234567   S>MS      MS>IEEE   F.  -1234567.
123.456e   IEEE>MS   MS>IEEE   F.  123.4560  ok
```

Some Support Words

```
: S.D>MS  ( n1 n2 --- n3 )
\ scale n1 by decimals n2 to MSFloat n3
  10e S>F f** s>f f* IEEE>MS ;

123456  3  S.D>MS ms. 123456000.
123456 -3  S.D>MS ms. 123.45600  ok
```

Summary

Some conversions done on the data stack, others done on the floating-point stack. Be flexible.

I had to find an obscure Q-Basic book from the 1990s to find the MicroSoft format.

This format from the 1980s will live forever due to the sunk-cost in data bases.

I have no reason to do math on the MS format. Always done as Win32Forth IEEE 64-bit floats.