Floating Point Routines for the 6502

by Roy Rankin, Department of Mechanical Engineering,
Stanford University, Stanford, CA 94305
(415) 497-1822

and

Steve Wozniak, Apple Computer Company
770 Welch Road, Suite 154
Palo Alto, CA 94304
(415) 326-4248

Editor's Note: Although these routines are for the 6502, it would appear that one could generate equivalent routines for most of the "traditional" microprocessors, relatively easily, by following the flow of the algorithms given in the excellent comments included in the program listing. This is particularly true of the transcendental functions which were directly modeled after well-known and proven algorithms, and for which, the comments are relatively machine-independent.

These floating point routines allow 6502 users to perform most of the more popular and desired floating point and transcendental functions, namely:

Natural Log - LOG
Common Log - LOG10
Exponential - EXP
Floating Add - FADD
Floating Subtract - FSUB
Floating Multiply - FMUL
Floating Divide - FDIV
Convert Floating to Fixed - FIX
Convert Fixed to Floating - FLOAT

They presume a four-byte floating point operand consisting of a one-byte exponent ranging from -218 through +127, and a 24-bit two's complement mantissa between 1.0 and 2.0.

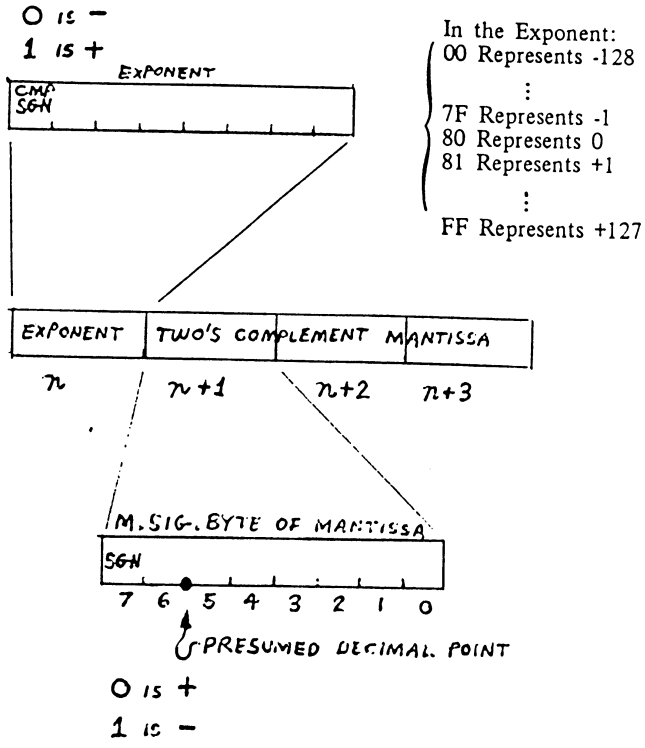
The floating point routines were done by Steve Wozniak, one of the principals in Apple Computer Company. The transcendental functions were patterned after those offered by Hewlett-Packard for their HP2100 minicomputer (with some modifications), and were done by Roy Rankin, a Ph.D. student at Stanford University.

There are three error traps; two for overflow, and one for prohibited logarithm argument. ERROR (1DO6) is the error exit used in event of a non-positive log argument. OVFLW (1E3B) is the error exit for overflow occuring during calculation of e to some power. OVFL (1FE4) is the error exit for overflow in all of the floating point routines. There is no trap for underflow; in such cases, the result is set to 0.0.

All routines are called and exited in a uniform manner: The argument(s) are placed in the specified floating point storage locations (for specifics, see documentation preceeding each routine in the listing), then a JSR is used to enter the desired routine. Upon normal completion, the called routine is exited via a subroutine return instruction (RTS).

Note: The preceeding documentation was written by the Editor, based on phone conversations with Roy and studying the listing. There is a high probability that it is correct. However, since it was not written nor reviewed by the authors of these routines, the preceeding documentation may contain errors in concept or in detail.

– JCW, Jr.




```
*       JULY 5, 1976
*   BASIC FLOATING POINT ROUTINES
*     FOR 6502 MICROPROCESSOR
*     BY R. RANKIN AND S. WOZNIAK
*
*   CONSISTING OF:
*      NATURAL LOG
*      COMMON LOG
*      EXPONENTIAL (E**X)
*      FLOAT      FIX
*      FADD       FSUB
*      FMUL       FDIV
*
*
*   FLOATING POINT REPRESENTATION (4-BYTES)
*                EXPONENT BYTE 1
*                MANTISSA BYTES 2-4
*
*   MANTISSA:    TWO'S COMPLIMENT REPRESENTATION WITH SIGN IN
*     MSB OF HIGH-ORDER BYTE.  MANTISSA IS NORMALIZED WITH AN
*     ASSUMED DECIMAL POINT BETWEEN BITS 5 AND 6 OF THE HIGH-ORDER
*     BYTE.  THUS THE MANTISSA IS IN THE RANGE 1. TO 2. EXCEPT
*     WHEN THE NUMBER IS LESS THAN 2**(-128).
*
*   EXPONENT:    THE EXPONENT REPRESENTS POWERS OF TWO.  THE
*     REPRESENTATION IS 2'S COMPLIMENT EXCEPT THAT THE SIGN
*     BIT (BIT 7) IS COMPLIMENTED.  THIS ALLOWS DIRECT COMPARISION
*     OF EXPONENTS FOR SIZE SINCE THEY ARE STORED IN INCREASING
*     NUMERICAL SEQUENCE RANGING FROM $00 (-128) TO $FF (+127)
*     ($ MEANS NUMBER IS HEXADECIMAL).
*
*   REPRESENTATION OF DECIMAL NUMBERS:    THE PRESENT FLOATING
*     POINT REPRESENTATION ALLOWS DECIMAL NUMBERS IN THE APPROXIMATE
*     RANGE OF 10**(-38) THROUGH 10**(38) WITH 6 TO 7 SIGNIFICANT
*     DIGITS.
*
*

0003                    ORG 3          SET BASE PAGE ADRESSES
0003 EA          SIGN   NOP
0004 EA          X2     NOP            EXPONENT 2
0005 00 00 00    M2     BSS 3          MANTISSA 2
0008 EA          X1     NOP            EXPONENT 1
0009 00 00 00    M1     BSS 3          MANTISSA 1
000C             E      BSS 4          SCRATCH
0010             S      BSS 4
0014             T      BSS 4
```

```
0018            SEXP   BSS 4
001C  00        INT    BSS 1
                *
1D00                   ORG $1D00     STARTING LOCATION FOR LOG
                *
                *      NATURAL LOG OF MANT/EXP1 WITH RESULT IN MANT/EXP1
                *
1D00  A5 09     LOG    LDA M1
1D02  F0 02            BEQ ERROR
1D04  10 01            BPL CONT      IF ARG>0 OK
1D06  00        ERROR  BRK           ERROR ARG<=0
                *
1D07  20 1C 1F  CONT   JSR SWAP      MOVE ARG TO EXP/MANT2
1D0A  A5 04            LDA X2        HOLD EXPONENT
1D0C  A0 80            LDY #$80
1D0E  84 04            STY X2        SET EXPONENT 2 TO 0 ($80)
1D10  49 80            EOR #$80      COMPLIMENT SIGN BIT OF ORIGINAL EXPONENT
1D12  85 0A            STA M1+1      SET EXPONENT INTO MANTISSA 1 FOR FLOAT
1D14  A9 00            LDA #0
1D16  85 09            STA M1        CLEAR MSB OF MANTISSA 1
1D18  20 2C 1F         JSR FLOAT     CONVERT TO FLOATING POINT
1D1B  A2 03            LDX #3        4 BYTE TRANSFERS
1D1D  B5 04     SEXP1  LDA X2,X
1D1F  95 10            STA Z,X       COPY MANTISSA TO Z
1D21  B5 08            LDA X1,X
1D23  95 18            STA SEXP,X    SAVE EXPONENT IN SEXP
1D25  BD D1 1D         LDA R22,X     LOAD EXP/MANT1 WITH SQRT(2)
1D28  95 08            STA X1,X
1D2A  CA               DEX
1D2B  10 F0            BPL SEXP1
1D2D  20 4A 1F         JSR FSUB      Z-SQRT(2)
1D30  A2 03            LDX #3         4 BYTE TRANSFER
1D32  B5 08     SAVET  LDA X1,X      SAVE EXP/MANT1 AS T
1D34  95 14            STA T,X
1D36  B5 10            LDA Z,X       LOAD EXP/MANT1 WITH Z
1D38  95 08            STA X1,X
1D3A  BD D1 1D         LDA R22,X     LOAD EXP/MANT2 WITH SQRT(2)
1D3D  95 04            STA X2,X
1D3F  CA               DEX
1D40  10 F0            BPL SAVET
1D42  20 50 1F         JSR FADD      Z+SQRT(2)
1D45  A2 03            LDX #3        4 BYTE TRANSFER
1D47  B5 14     TM2    LDA T,X
1D49  95 04            STA X2,X      LOAD T INTO EXP/MANT2
1D4B  CA               DEX
1D4C  10 F9            BPL TM2
1D4E  20 9D 1F         JSR FDIV      T=(Z-SQRT(2))/(Z+SQRT(2))
1D51  A2 03            LDX #3        4 BYTE TRANSFER
1D53  B5 08     M1T    LDA X1,X
1D55  95 14            STA T,X       COPY EXP/MANT1 TO T AND
1D57  95 04            STA X2,X      LOAD EXP/MANT2 WITH T
1D59  CA               DEX
1D5A  10 F7            BPL M1T
1D5C  20 77 1F         JSR FMUL   T*T
1D5F  20 1C 1F         JSR SWAP      MOVE T*T TO EXP/MANT2
1D62  A2 03            LDX #3        4 BYTE TRANSFER
1D64  BD E1 1D  M1C    LDA C,X
1D67  95 08            STA X1,X      LOAD EXP/MANT1 WITH C
1D69  CA               DEX
1D6A  10 F8            BPL M1C
1D6C  20 4A 1F         JSR FSUB      T*T-C
1D6F  A2 03            LDX #3        4 BYTE TRANSFER
1D71  BD DD 1D  M2MB   LDA MB,X
1D74  95 04            STA X2,X      LOAD EXP/MANT2 WITH MB
1D76  CA               DEX
1D77  10 F8            BPL M2MB
1D79  20 9D 1F         JSR FDIV      MB/(T*T-C)
1D7C  A2 03            LDX #3        4 BYTE TRANSFER
1D7E  BD D9 1D  M2A1   LDA A1,X
1D81  95 04            STA X2,X      LOAD EXP/MANT2 WITH A1
1D83  CA               DEX
1D84  10 F8            BPL M2A1
1D86  20 50 1F         JSR FADD      MB/(T*T-C)+A1
1D89  A2 03            LDX #3        4 BYTE TRANSFER
1D8B  B5 14     M2T    LDA T,X
1D8D  95 04            STA X2,X      LOAD EXP/MANT2 WITH T
1D8F  CA               DEX
1D90  10 F9            BPL M2T
1D92  20 77 1F         JSR FMUL      (MB/(T*T-C)+A1)*T
1D95  A2 03            LDX #3        4 BYTE TRANSFER
1D97  BD E5 1D  M2MHL  LDA MHLF,X
1D9A  95 04            STA X2,X      LOAD EXP/MANT2 WITH MHLF (.5)
1D9C  CA               DEX
1D9D  10 F8            BPL M2MHL
1D9F  20 50 1F         JSR FADD      +.5
1DA2  A2 03            LDX #3        4 BYTE TRANSFER
1DA4  B5 18     LDEXP  LDA SEXP,X
1DA6  95 04            STA X2,X      LOAD EXP/MANT2 WITH ORIGINAL EXPONENT
1DA8  CA               DEX
1DA9  10 F9            BPL LDEXP
1DAB  20 50 1F         JSR FADD      +EXPN
1DAE  A2 03            LDX #3        4 BYTE TRANSFER
1DB0  BD D5 1D  MLE2   LDA LE2,X
1DB3  95 04            STA X2,X      LOAD EXP/MANT2 WITH LN(2)
1DB5  CA               DEX
1DB6  10 F8            BPL MLE2
1DB8  20 77 1F         JSR FMUL   *LN(2)
1DBB  60               RTS           RETURN RESULT IN MANT/EXP1
                *
                *      COMMON LOG  OF MANT/EXP1 RESULT IN MANT/EXP1
                *
1DBC  20 00 1D  LOG10  JSR LOG       COMPUTE NATURAL LOG
1DBF  A2 03            LDX #3
1DC1  BD CD 1D  L10    LDA LN10,X
1DC4  95 04            STA X2,X      LOAD EXP/MANT2 WITH 1/LN(10)
1DC6  CA               DEX
1DC7  10 F8            BPL L10
1DC9  20 77 1F         JSR FMUL      LOG10(X)=LN(X)/LN(10)
1DCC  60               RTS
                *
1DCD  7E 6F     LN10   DCM 0.4342945
      2D ED
1DD1  80 5A     R22    DCM   1.4142136   SQRT(2)
      82 7A
1DD5  7F 58     LE2    DCM   0.69314718  LOG BASE E OF 2
      B9 0C
1DD9  80 52     A1     DCM   1.2920074
      B8 40
1DDD  81 A8     MB     DCM   -2.6398577
      86 49
1DE1  80 6A     C      DCM   1.6567626
      08 66
1DE5  7F 40     MHLF   DCM   0.5
      00 00
                *
1E00                   ORG $1E00     STARTING LOCATION FOR EXP
                *
                *      EXP OF MANT/EXP1 RESULT IN MANT/EXP1
                *
1E00  A2 03     EXP    LDX #3        4 BYTE TRANSFER
1E02  BD D8 1E         LDA L2E,X
1E05  95 04            STA X2,X      LOAD EXP/MANT2 WITH LOG BASE 2 OF E
1E07  CA               DEX
1E08  10 F8            BPL EXP+2
1E0A  20 77 1F         JSR FMUL      LOG2(E)*X
1E0D  A2 03            LDX #3        4 BYTE TRANSFER
1E0F  B5 08     FSA    LDA X1,X
1E11  95 10            STA Z,X       STORE EXP/MANT1 IN Z
1E13  CA               DEX
1E14  10 F9            BPL FSA       SAVE Z=LN(2)*X
1E16  20 E8 1F         JSR FIX       CONVERT CONTENTS OF EXP/MANT1 TO AN INTEGER
1E19  A5 0A            LDA M1+1
1E1B  85 1C            STA INT       SAVE RESULT AS INT
1E1D  38               SEC           SET CARRY FOR SUBTRACTION
1E1E  E9 7C            SBC #124      INT-124
1E20  A5 09            LDA M1
1E22  E9 00            SBC #0
1E24  10 15            BPL OVFLW     OVERFLOW INT>=124
1E26  18               CLC           CLEAR CARRY FOR ADD
1E27  A5 0A            LDA M1+1
1E29  69 78            ADC #120      ADD 120 TO INT
1E2B  A5 09            LDA M1
1E2D  69 00            ADC #0
1E2F  10 08            BPL CONTIN    IF RESULT POSITIVE CONTINUE
1E31  A9 00            LDA #0        INT<-120 SET RESULT TO ZERO AND RETURN
1E33  A2 03            LDX #3        4 BYTE MOVE
1E35  95 08     ZERO   STA X1,X      SET EXP/MANT1 TO ZERO
1E37  CA               DEX
1E38  10 F8            BPL ZERO
1E3A  60               RTS           RETURN
                *
1E3B  00        OVFLW  BRK           OVERFLOW
                *
1E3C  20 2C 1F  CONTIN JSR FLOAT     FLOAT INT
1E3F  A2 03            LDX #3
1E41  B5 10     ENTD   LDA Z,X
1E43  95 04            STA X2,X      LOAD EXP/MANT2 WITH Z
1E45  CA               DEX
1E46  10 F9            BPL ENTD
1E48  20 4A 1F         JSR FSUB      Z=Z-FLOAT(INT)
1E4B  A2 03            LDX #3        4 BYTE MOVE
1E4D  B5 08     ZSAV   LDA X1,X
1E4F  95 10            STA Z,X       SAVE EXP/MANT1 IN Z
1E51  95 04            STA X2,X      COPY EXP/MANT1 TO EXP/MANT2
1E53  CA               DEX
1E54  10 F7            BPL ZSAV
1E56  20 77 1F         JSR FMUL      Z*Z
1E59  A2 03            LDX #3        4 BYTE MOVE
1E5B  BD DC 1E  LA2    LDA A2,X
1E5E  95 04            STA X2,X      LOAD EXP/MANT2 WITH A2
1E60  B5 08            LDA X1,X
1E62  95 18            STA SEXP,X    SAVE EXP/MANT1 AS SEXP
1E64  CA               DEX
1E65  10 F4            BPL LA2
1E67  20 50 1F         JSR FADD      Z*Z+A2
1E6A  A2 03            LDX #3        4 BYTE MOVE
1E6C  BD E0 1E  LB2    LDA B2,X
1E6F  95 04            STA X2,X      LOAD EXP/MANT2 WITH B2
1E71  CA               DEX
1E72  10 F8            BPL LB2
1E74  20 9D 1F         JSR FDIV      T=B2/(Z*Z+A2)
1E77  A2 03            LDX #3        4 BYTE MOVE
1E79  B5 08     DLOAD  LDA X1,X
1E7B  95 14            STA T,X       SAVE EXP/MANT1 AS T
1E7D  BD E4 1E         LDA C2,X
1E80  95 08            STA X1,X      LOAD EXP/MANT1 WITH C2
1E82  B5 18            LDA SEXP,X
1E84  95 04            STA X2,X      LOAD EXP/MANT2 WITH SEXP
1E86  CA               DEX
1E87  10 F0            BPL DLOAD
1E89  20 77 1F         JSR FMUL      Z*Z*C2
1E8C  20 1C 1F         JSR SWAP      MOVE EXP/MANT1 TO EXP/MANT2
1E8F  A2 03            LDX #3        4 BYTE TRANSFER
1E91  B5 14     LTMP   LDA T,X
1E93  95 08            STA X1,X      LOAD EXP/MANT1 WITH T
1E95  CA               DEX
```

```
1E96  10 F9           BPL LTMP
1E98  20 4A 1F        JSR FSUB      C2*Z*Z-B2/(Z*Z+A2)
1E9B  A2 03           LDX =3        4 BYTE TRANSFER
1E9D  B5 E8 1E  LDD   LDA D.X
1EA0  95 04           STA X2.X      LOAD EXP/MANT2 WITH D
1EA2  CA              DEX
1EA3  10 F8           BPL LDD
1EA5  20 50 1F        JSR FADD      D+C2*Z*Z-B2/(Z*Z+A2)
1EA8  20 1C 1F        JSR SWAP      MOVE EXP/MANT1 TO EXP/MANT2
1EAB  A2 03           LDX =3        4 BYTE TRANSFER
1EAD  B5 10     LFA   LDA Z.X
1EAF  95 08           STA X1.X      LOAD EXP/MANT1 WITH Z
1EB1  CA              DEX
1EB2  10 F9           BPL LFA
1EB4  20 4A 1F        JSR FSUB      -Z+D+C2*Z*Z-B2/(Z*Z+A2)
1EB7  A2 03           LDX =3        4 BYTE TRANSFER
1EB9  B5 10     LF3   LDA Z.X
1EBB  95 04           STA X2.X      LOAD EXP/MANT2 WITH Z
1EBD  CA              DEX
1EBE  10 F9           BPL LF3
1EC0  20 9D 1F        JSR FDIV      Z/(**** )
1EC3  A2 03           LDX =3        4 BYTE TRANSFER
1EC5  BD E5 1D  LD12  LDA MHLF.X
1EC8  95 04           STA X2.X      LOAD EXP/MANT2 WITH .5
1ECA  CA              DEX
1ECB  10 F8           BPL LD12
1ECD  20 50 1F        JSR FADD      +Z/(***)+.5
1ED0  38              SEC           ADD INT TO EXPONENT WITH CARRY SET
1ED1  A5 1C           LDA INT       TO MULTIPLY BY
1ED3  65 08           ADC X1        2**(INT+1)
1ED5  85 08           STA X1        RETURN RESULT TO EXPONENT
1ED7  60              RTS           RETURN ANS*(.5+Z/(-Z+D+C2*Z*Z-B2/(Z*
1ED8  80 5C     L2E   DCM 1.4426950409   LOG BASE 2 OF E
      55 1E
1EDC  86 57     A2    DCM 87.417497202
      6A E1
1EE0  89 4D     B2    DCM 617.9722695                    Z+A2))*2**(INT+1)
      3F 1D
1EE4  7B 46     C2    DCM .03465735903
      FA 70
1EE8  83 4F     D     DCM 9.9545957821
      A3 03
                *
                *
                *       BASIC FLOATING POINT ROUTINES
                *
1F00                  ORG $1F00     START OF BASIC FLOATING POINT ROUTINES
1F00  18        ADD   CLC           CLEAR CARRY
1F01  A2 02           LDX =$02      INDEX FOR 3-BYTE ADD
1F03  B5 09     ADD1  LDA M1.X
1F05  75 05           ADC M2.X      ADD A BYTE OF MANT2 TO MANT1
1F07  95 09           STA M1.X
1F09  CA              DEX           ADVANCE INDEX TO NEXT MORE SIGNIF.BYTE
1F0A  10 F7           BPL ADD1      LOOP UNTIL DONE
1F0C  60              RTS           RETURN
1F0D  06 03     MD1   ASL SIGN      CLEAR LSB OF SIGN
1F0F  20 12 1F        JSR ABSWAP    ABS VAL OF MANT1. THEN SWAP MANT2
1F12  24 09     ABSWAP BIT M1       MANT1 NEG?
1F14  10 05           BPL ABSWP1    NO.SWAP WITH MANT2 AND RETURN
1F16  20 8F 1F        JSR FCOMPL    YES, COMPLIMENT IT.
1F19  E6 03           INC SIGN      INCR SIGN, COMPLEMENTING LSB
1F1B  38        ABSWP1 SEC          SET CARRY FOR RETURN TO MUL/DIV
                *
                *       SWAP EXP/MANT1 WITH EXP/MANT2
                *
1F1C  A2 04     SWAP  LDX =$04      INDEX FOR 4-BYTE SWAP
1F1E  94 08     SWAP1 STY E-1.X
1F20  B5 07           LDA X1-1.X    SWAP A BYTE OF EXP/MANT1 WITH
1F22  B4 03           LDY X2-1.X    EXP/MANT2 AND LEAVEA COPY OF
1F24  94 07           STY X1-1.X    MANT1 IN E(3BYTES). E+3 USED.
1F26  95 03           STA X2-1.X
1F28  CA              DEX           ADVANCE INDEX TO NEXT BYTE
1F29  D0 F3           BNE SWAP1     LOOP UNTIL DONE.
1F2B  60              RTS
                *
                *
                *
                *     CONVERT 16 BIT INTEGER IN M1(HIGH) AND M1+1(LOW) TO
                *     RESULT IN EXP/MANT1. EXP/MANT2 UNEFFECTED      F.P.
                *
                *
1F2C  A9 8E     FLOAT LDA =$8E
1F2E  85 08           STA X1        SET EXPN TO 14 DEC
1F30  A9 00           LDA =0        CLEAR LOW ORDER BYTE
1F32  85 0B           STA M1+2
1F34  F0 08           BEQ NORM      NORMALIZE RESULT
1F36  C6 0B     NORM1 DEC X1        DECREMENT EXP1
1F38  06 0B           ASL M1+2
1F3A  26 0A           ROL M1+1      SHIFT MANT1 (3 BYTES) LEFT
1F3C  26 09           ROL M1
1F3E  A5 09     NORM  LDA M1        HIGH ORDER MANT1 BYTE
1F40  0A              ASL A         UPPER TWO BITS UNEQUAL?
1F41  45 09           EOR M1
1F43  30 04           BMI RTS1      YES,RETURN WITH MANT1 NORMALIZED
1F45  A5 08           LDA X1        EXP1 ZERO?
1F47  D0 ED           BNE NORM1     NO, CONTINUE NORMALIZING
1F49  60        RTS1  RTS           RETURN
                *
                *
                *     EXP/MANT2-EXP/MAN1 RESULT IN EXP/MANT1
                *
1F4A  20 8F 1F  FSUB  JSR FCOMPL    COMPL MANT1 CLEARS CARRY UNLESS ZERO
1F4D  20 5D 1F  SUPALG JSR ALGNSW   RIGHT SHIFT MANT1 OR SWAP WITH
                *                                        MANT2 ON CARRY
1F50  A5 04           LDA X2
1F52  C5 08           CMP X1        COMPARE EXP1 WITH EXP2
1F54  D0 F7           BNE SUPALG    IF UNEQUAL, SWAP ADDENDS OR ALIGN MANTISSAS
1F56  20 00 1F        JSR ADD       ADD ALIGNED MANTISSAS
1F59  50 E3     ADDEND BVC NORM     NO OVERFLOW, NORMALIZE RESULTS
1F5B  70 05           BVS RTLOG     OV: SHIFT MANT1 RIGHT. NOTE CARRY IS CORRECT SIGN
1F5D  90 BD     ALGNSW BCC SWAP     SWAP IF CARRY CLEAR, ELSE SHIFT RIGHT ARITH.
1F5F  A5 09     RTAR  LDA M1        SIGN OF MANT1 INTO CARRY FOR
1F61  0A              ASL A         RIGHT ARITH SHIFT
1F62  E6 08     RTLOG INC X1        INCR EXP1 TO COMPENSATE FOR RT SHIFT
1F64  F0 7E           BEQ OVFL      EXP1 OUT OF RANGE
1F66  A2 FA     RTLOG1 LDX =$FA     INDEX FOR 6 BYTE RIGHT SHIFT
1F68  A9 80     ROR1  LDA =$80
1F6A  B0 01           BCS ROR2
1F6C  0A              ASL A
1F6D  56 0F     ROR2  LSR E+3.X     SIMULATE ROR E+3.X
1F6F  15 0F           ORA E+3.X
1F71  95 0F           STA E+3.X
1F73  E8              INX           NEXT BYTE OF SHIFT
1F74  D0 F2           BNE ROR1      LOOP UNTIL DONE
1F76  60              RTS           RETURN
                *
                *
                *     EXP/MANT1 X EXP/MANT2 RESULT IN EXP/MANT1
                *
1F77  20 0D 1F  FMUL  JSR MD1       ABS. VAL OF THE MANT1, MANT2
1F7A  65 08           ADC X1        ADD EXP1 TO EXP2 FOR PRODUCT EXPONENT
1F7C  20 CD 1F        JSR MD2       CHECK PRODUCT EXP AND PREPARE FOR MUL
1F7F  18              CLC           CLEAR CARRY
1F80  20 66 1F  MUL1  JSR RTLOG1    MANT1 AND E RIGHT.(PRODUCT AND MPLIER)
1F83  90 03           BCC MUL2      IF CARRY CLEAR, SKIP PARTIAL PRODUCT
1F85  20 00 1F        JSR ADD       ADD MULTIPLICAN TO PRODUCT
1F88  88        MUL2  DEY           NEXT MUL ITERATION
1F89  10 F5           BPL MUL1      LOOP UNTIL DONE
1F8B  46 03     MDEND LSR SIGN      TEST SIGN (EVEN/ODD)
1F8D  90 AF     NORMX BCC NORM      IF EXEN. NORMALIZE PRODUCT. ELSE COMPLEMENT
1F8F  38        FCOMPL SEC          SET CARRY FOR SUBTRACT
1F90  A2 03           LDX =$03      INDEX FOR 3-BYTE SUBTRACTION
1F92  A9 00     COMPL1 LDA =$00     CLEAR A
1F94  F5 08           SBC X1.X      SUBTRACT BYTE OF EXP1
1F96  95 08           STA X1.X      RESTORE IT
1F98  CA              DEX           NEXT MORE SIGNFICANT BYTE
1F99  D0 F7           BNE COMPL1    LOOP UNTIL DONE
1F9B  F0 BC           BEQ ADDEND    NORMALIZE (OR SHIFT RIGHT IF OVERFLOW)
                *
                *
                *     EXP/MANT2 / EXP/MANT1 RESULT IN EXP/MANT1
                *
1F9D  20 0D 1F  FDIV  JSR MD1       TAKE ABS VAL OF MANT1, MANT2
1FA0  E5 08           SBC X1        SUBTRACT EXP1 FROM EXP2
1FA2  20 CD 1F        JSR MD2       SAVE AS QUOTIENT EXP
1FA5  38        DIV1  SEC           SET CARRY FOR SUBTRACT
1FA6  A2 02           LDX =$02      INDEX FOR 3-BYTE INSTRUCTION
1FA8  B5 05     DIV2  LDA M2.X
1FAA  F5 0C           SBC E.X       SUBTRACT A BYTE OF E FROM MANT2
1FAC  48              PHA           SAVE ON STACK
1FAD  CA              DEX           NEXT MORE SIGNIF BYTE
1FAE  10 F8           BPL DIV2      LOOP UNTIL DONE
1FB0  A2 FD           LDX =$FD      INDEX FOR 3-BYTE CONDITIONAL MOVE
1FB2  68        DIV3  PLA           PULL A BYTE OF DIFFERENCE OFF STACK
1FB3  90 02           BCC DIV4      IF MANT2<E THEN DONT RESTURE MANT2
1FB5  95 08           STA M2+3.X
1FB7  E8        DIV4  INX           NEXT LESS SIGNIF BYTE
1FB8  D0 F8           BNE DIV3      LOOP UNTIL DONE
1FBA  26 0B           ROL M1+2
1FBC  26 0A           ROL M1+1      ROLL QUOTIENT LEFT. CARRY INTO LSB
1FBE  26 09           ROL M1
1FC0  06 07           ASL M2+2
1FC2  26 06           ROL M2+1      SHIFT DIVIDEND LEFT
1FC4  26 05           ROL M2
1FC6  B0 1C           BCS OVFL      OVERFLOW IS DUE TO UNNORMALIZED DIVISOR
1FC8  88              DEY           NEXT DIVIDE ITERATION
1FC9  D0 DA           BNE DIV1      LOOP UNTIL DONE 23 ITERATIONS
1FCB  F0 BE           BEQ MDEND     NORMALIZE QUOTIENT AND CORRECT SIGN
1FCD  86 0B     MD2   STX M1+2
1FCF  86 0A           STX M1+1      CLR MANT1 (3 BYTES) FOR MUL/DIV
1FD1  86 09           STX M1
1FD3  B0 0D           BCS OVCHK     IF EXP CALC SET CARRY, CHECK FOR OVFL
1FD5  30 04           BMI MD3       IF NEG NO UNDERFLOW
1FD7  68              PLA           POP ONE
1FD8  68              PLA           RETURN LEVEL
1FD9  90 B2           BCC NORMX     CLEAR X1 AND RETURN
1FDB  49 80     MD3   EOR =$80      COMPLIMENT SIGN BIT OF EXP
1FDD  85 08           STA X1        STORE IT
1FDF  A0 17           LDY =$17      COUNT FOR 24 MUL OR 23 DIV ITERATIONS
1FE1  60              RTS           RETURN
1FE2  10 F7     OVCHK BPL MD3       IF POS EXP THEN NO OVERFLOW
1FE4  00        OVFL  BRK
                *
                *
                *     CONVERT EXP/MANT1 TO INTEGER IN M1 (HIGH) AND M1+1(LOW)
                *      EXP/MANT2 UNEFFECTED
                *
1FE5  20 5F 1F        JSR RTAR      SHIFT MANT1 RT AND INCREMENT EXPNT
1FE8  A5 00     FIX   LDA X1        CHECK EXPONENT
1FEA  C9 8E           CMP =$8E      IS EXPONENT 14?
1FEC  D0 F7           BNE FIX-3     NO.SHIFT
1FEE  60        RTRN  RTS           RETURN
                      END
```