A Floating Point Package for the HPC

National Semiconductor
Application Note 486
Ashok Krishnamurthy
April 1987

INTRODUCTION

This report describes the implementation of a Single Precision Floating Point Arithmetic package for the National Semiconductor HPC microcontroller. The package is based upon the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754–1985). However, the package is not a conforming implementation of the standard. The differences between the HPC implementation and the standard are described later in this report.

The following single precision (SP) operations have been implemented in the package.

(1) FADD. Addition of two SP floating point (FLP) numbers.

(2) FSUB. Subtraction of two SP FLP numbers.

(3) FMULT. Multiplication of two SP FLP numbers.

(4) FDIV. Division of two SP FLP numbers.

(5) ATOF. Convert an ASCII string representing a decimal FLP number to a binary SP FLP number.

(6) FTOA. Convert a binary SP FLP number to a decimal FLP number and output the decimal FLP number as an ASCII string.

The report is organized as follows. The next section discusses the representation of FLP numbers. Then, the differences between the HPC implementation and the IEEE/ANSI standard are described. This is followed by a description of the algorithms used in the computations. Appendix A is a User's Manual for the package, Appendix B describes the test data for the package and Appendix C is a listing of the code.

Note that this report assumes that the reader is familiar with the IEEE/ANSI Binary Floating-Point Standard. Please refer to this document for an explanation of the terms used here.

REPRESENTATION OF FLOATING POINT NUMBERS

The specification of a binary floating point number involves two parts: a mantissa and an exponent. The mantissa is a signed fixed point number and the exponent is a signed integer. The IEEE/ANSI standard specifies that a SP FLP number shall be represented in 32 bits as shown in *Figure 1*.

1	8	23
S	E	F

FIGURE 1

The significance of each of these fields is as follows:

1. S—this 1-bit field is the sign of the mantissa. S = 0 means that the number is positive, while S = 1 means that it is negative.
2. E—this is the 8-bit exponent field. The exponent is represented as a biased value with a bias of 127-decimal.
3. F—this is the 23-bit mantissa field. For normalized FLP numbers (see below), a MSB of 1 is assumed and not represented. Thus, for normalized numbers, the value of the mantissa is 1.F. This provides an effective precision of 24 bits for the mantissa.

Normalized FLP number: A binary FLP number is said to be normalized if the value of the MSB of the mantissa is 1. Normalization is important and useful because it provides maximum precision in the representation of the number. If we deal with normalized numbers only (as the HPC implementation does) then since the MSB of the mantissa is always 1, it need not be explicitly represented. This is as specified in the IEEE/ANSI standard.

Given the values of S , E and F, the value of the SP FLP number is obtained as follows.

If 0 < E < 255, then the FLP number is (−1) ^S*1.F*2^(E−127).

If E = 0, then the value of the FLP number is 0.

If E = 255, then the FLP number is not a valid number (NAN).

The above format for binary SP FLP numbers provides for the representation of numbers in the range −3.4*10^38 to −1.75*10^−38, 0, and 1.75*10^−38 to 3.4*10^38. The accuracy is between 7 and 8 decimal digits.

DIFFERENCES BETWEEN THE IMPLEMENTATION AND THE IEEE/ANSI STANDARD

The IEEE/ANSI standard specifies a comprehensive list of operations and representations for FLP numbers. Since an implementation that fully conforms to this standard would lead to an excessive amount of overhead, a number of the features in the standard were dropped. This section describes the differences between the implemented package and the standard.

1. Omission of −0. The IEEE/ANSI standard requires that both + and − zero be represented, and arithmetic carried out using both. The implementation does not represent −0. Only +0 is represented and arithmetic is carried out with +0 only.
2. Omission of Infinity Arithmetic. The IEEE/ANSI standard provides for the representation of plus and minus Infinity, and requires that valid arithmetic operations be carried out on Infinity. The HPC implementation does not support this.
3. Omission of Quiet NaN. The IEEE/ANSI standard provides for both quiet and signalling NaNs. The HPC implementation provides for signalling NaNs only. A signalling NaN can be produced as the result of overflow during an arithmetic operation. If the NaN is passed as input to further floating point routines, then these routines will produce another NaN as output. The routines will also set the Invalid Operation flag, and call the user floating point error trap routine at address FPTRAP.
4. Omission of denormalized numbers. Denormalized numbers are FLP numbers with a biased exponent, E of zero and a non zero mantissa F. Such denormalized numbers are useful in providing gradual underflow to zero. Denormalized numbers are not represented or used in the HPC implementation. Instead, if the result of a computation cannot be represented as a normalized number within the allowable exponent range, then an underflow is signaled, the result is set to zero, and the user floating point error trap routine at address FPTRAP is called.
5. Omission of the Inexact Result exception. The IEEE/ANSI standard requires that an Inexact Result exception be signaled when the rounded result of an operation is not exact, or it overflows without an overflow trap. This feature is not provided in the HPC implementation.

6. Biased Rounding to Nearest. The IEEE/ANSI standard requires that rounding to nearest be provided as the default rounding mode. Further, the rounding is required to be unbiased. The HPC implementation provides biased rounding to nearest only. An example will help clarify this.

   Suppose the result of an operation is .b1b2b3XXX and needs to be rounded to 3 binary digits. Then if XXX is 0YY, the round to nearest result is .b1b2b3. If XXX is 1YY, with at least one of the Y's being 1, then the result is .b1b2b3 + 0.001. Finally if XXX is 100, it is a tie situation. In such a case, the IEEE/ANSI standard requires that the rounded result be such that its LSB is 0. The HPC implementation, on the other hand, will round the result in such a case to .b1b2b3 + 0.001.

DESCRIPTION OF ALGORITHMS

1. **General Considerations.** The HPC implementation of the SP floating point package consists of a series of subroutines. The subroutines have been designed to be compatible with the CCHPC C Cross Compiler. They have, however, not been tested with the CCHPC Cross Compiler.

   The Arithmetic subroutines that compute F1 op F2 (where op is +, −, * or /) expect that F1 and F2 are input in the IEEE format. Each of F1 and F2 consists of two 16-bit words organized as follows.

   Fn−HI: S EXP 7 MS bits of F

   Fn−LO: 16 LS bits of F

   In the above, S is the sign of the mantissa, EXP is the biased exponent, and F is the mantissa.

   On input it is assumed that F1−HI is in register K, F1−LO is in the accumulator A, and F2−HI and F2−LO are on the stack just below the return address i.e., F2−HI is at W(SP-4) and F2−LO is at W(SP-6). The result, C, is also returned in IEEE format with C−HI in register K and C−LO in the accumulator A.

   The two Format Conversion routines, ATOF and FTOA expect that on entry, register B contains the address of the start of the ASCII byte string representing the decimal FLP number. ATOF reads the byte string starting from this address. Note that the string must be terminated with a null byte. The binary floating point number is returned in registers K and A. FTOA, on the other hand, writes the decimal FLP string starting from the address in register B on entry. A terminating null byte is also output. Also, FTOA expects that the binary FLP number to be converted is in registers K and A on entry.

   Most of the storage required by the subroutines is obtained from the stack. Two additional words of storage in the base page are also used. The first is W(0), and is referenced in the subroutines as W(TMP1). The second word of storage can be anywhere in the base page and is used to store the sticky flags used to signal floating point exceptions. This is referenced in the subroutines as W(FPERWD). Thus any user program that uses the floating point package needs to have the symbols TMP1 and FPERWD defined appropriately.

2. **Exception Handling.** The following types of exception can occur during the course of a computation.

   (i) Invalid Operand. This exception occurs if one of the input operands is a NaN.

   (ii) Exponent Overflow. This occurs if the result of a computation is such that its exponent has a biased value of 255 or more.

   (iii) Exponent Underflow. This occurs if the result of a computation is such that its exponent is 0 or less.

   (iv) Divide-by-zero. This exception occurs if the FDIV routine is called with F2 being zero.

   The package signals exceptions in two ways. First a word at address FPERWD is maintained that records the history of these exception conditions. Bits 0−3 of this word are used for this purpose.

   Bit 0—Set on Exponent Overflow.

   Bit 1—Set on Exponent Underflow.

   Bit 2—Set on Illegal Operand.

   Bit 3—Set on Divide-by-zero.

   These bits are never cleared by the floating point package, and can be examined by the user software to determine the exception conditions that occurred during the course of a computation. It is the responsibility of the user software to initialize this word before calling any of the floating point routines.

   The second method that the package uses to signal exceptions is to call a user floating point exception handler subroutine whenever an exception occurs. The corresponding exception bit in FPERWD is set before calling the handler. The starting address of the handler should be defined by the symbol FPTRAP.

3. **Unpacked Floating Point Format.** The IEEE/ANSI standard floating point format described earlier is very cumbersome to deal with during computation. This is primarily because of the splitting of the mantissa between the two words. The subroutines in the package unpack the input FLP numbers into an internal representation, do the computations using this representation, and finally pack the result into the IEEE format before return to the calling program. The unpacking is done by the subroutine FUNPAK and the packing by the subroutine FPAK. The unpacked format consists of 3 words and is organized as follows.

Fn-EXP.Fn-SIGN	8 bits biased exponent	sign (extended to 8 bits)
Fn-HI	MS 16 bits of mantissa (implicit 1 is present as MSB)	
Fn-LO	LS 8 bits of mantissa	Eight Zeros

   Since all computations are carried out in this format, note that the result is actually known to 32 bits. This 32-bit mantissa is rounded to 24 bits before being packed to the IEEE format.

4. **Algorithm Description.** All the arithmetic algorithms first check for the easy cases when either F1 or F2 is zero or a NaN. The result in these cases is immediately available. The description of the algorithms below is for those cases when neither F1 nor F2 is zero or a NaN. Also, in order to keep the algorithm description simple, the check for underflow/overflow at the various stages is not shown. The documentation in the program, the descriptions given below, and the theory as described in the references should allow these programs to be easily maintained.

   (i) FADD.

   The processing steps are as follows:

   1. Compare F1-EXP and F2-EXP. Let the difference be D. Shift right the mantissa (Fn-HI.Fn-LO, n = 1 or 2) of the FLP number with the smaller exponent D times. Let the numbers after this step be F1-EXP.F1-SIGN, F1-HI, F1-LO and F2-EXP.F2-SIGN,

F2-HI and F2-LO. This step equalizes the two exponents.

2. Take the XOR of F1-SIGN and F2-SIGN. If this is 0, then go to step 4, else go to step 3.
3. Do a true subtract of F2-LO from F1-LO. (A true subtract is when the SUBC instruction is preceded by a SET C instruction.) Then do a 1's complement subtract of F2-HI from F1-HI. If the last subtract resulted in C = 1, then go to step 3.2, else go to step 3.1.

   3.1. Get here means that F2 is larger than F1, and the computed result is negative. Take the 2's complement of the result to make it positive. Set the sign of the result to be the sign of F2. Go to step 3.3.

   3.2. Get here means F1 is larger than F2, and the result of the mantissa subtract is positive. Set the sign of the result to be the sign of F1. Go to step 3.3.

   3.3. The result after a subtract need not be normalized. Shift left the result mantissa until its MSB is 1. Decrement the exponent of the result by 1 for each such left shift. Go to step 5.
4. Add F2-LO to F1-LO. Next add with any carry from the previous add, F2-HI to F1-HI. If this last add results in C = 1, then go to step 4.1, else go to step 5.

   4.1. Rotate Right with carry C-HI. Next load C-LO in and rotate it right with carry. Increase the exponent of the result, C by 1. Go to step 5.
5. Round the result. Go to step 6.
6. Pack the result and return.

(ii) FSUB.

The processing steps are as follows:

1. Copy F2 to the stack and change its sign. Go to step 2.
2. Call FADD.
3. Remove the copy of -F2 from the stack and return.

(iii) FMULT.

The processing steps are as follows.

1. Add F1-EXP and F2-EXP to get C1-EXP. Subtract from C1-EXP 127-decimal which is the IEEE bias, to get C-EXP. Go to step 2.
2. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 3.
3. Compute F1-HI*F2-HI. Let the upper half of the product be C1-HI and the lower half C1-LO. Go to step 4.
4. Compute F1-HI*F2-LO. Let the upper half of this product be C2-HI. Add C2-HI to C1-LO to give C11-LO. If this last add results in C = 1, then increment C1-HI. Go to step 5.
5. Compute F1-LO*F2-HI. Let the upper half of this product be C3-HI. Add C3-HI to C11-LO to get C12-LO. If this last add results in C = 1, then increment C1-HI. Go to step 6.
6. Mantissa normalization. If the MSB of C1-HI is 1, then increment C-EXP, else shift left C1-HI.C12-LO. Go to step 7.
7. Round C1-HI.C12-LO to get C-HI.C-LO. Go to step 8.
8. Pack C-EXP.C-SIGN, C-HI and C-LO and return as the answer.

(iv) FDIV.

The processing steps are as follows:

1. Compare F1-HI and F2-HI. If F2-HI is greater than F1-HI then go to Step 3, else go to step 2.
2. Shift right F1-HI.F1-LO. Increase F1-EXP by 1.
3. Subtract F2-EXP from F1-EXP. Add to the result 127-decimal to get C1-EXP. Go to step 4.
4. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 5.
5. Compute F1-HI*F2-LO. Let the result be M1-HI.M1-LO. Go to step 6.
6. Divide M1-HI.M1-LO by F2-HI. Let the quotient be M2-HI. Go to step 7.
7. Do a true subtract of M2-HI from F1-LO. Let the result be M3-LO. If C = 1 as a result of this subtract, then go to step 8, else decrement F1-HI and go to step 8.
8. Divide F1-HI.M3-LO by F2-HI. Let the quotient be C1-HI and the remainder R1. Go to step 9.
9. Divide R1 .0000 by F2-HI. Let the quotient be C1-LO. Go to step 10.
10. If the MSB of C1-HI is 1 then go to step 11, else shift left C1-HI.C1-LO, decrease C1-EXP by 1 and go to step 11.
11. Round C1-HI.C1-LO to get C-HI.C-LO. go to step 12.
12. Pack C1-EXP.C-SIGN, C-HI and C-LO and return as the result.

(v) ATOF.

The processing steps in this case are as follows.

1. Set M-SIGN, the mantissa sign to 0.

   Set M10-EXP, the implicit decimal exponent to 0.

   Set HI-INT to 0.

   Set LO-INT to 0.

   Go to step 2.
2. Get a character from the input string. Let the character be C.

   If C is a '+', then go to the start of step 2.

   If C is a '−', then set M-SIGN to FF and go to start of step 2.

   If C is a '.', then go to step 5.

   If C is none of the above, then go to step 3.
3. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 3.1, else go to step 3.2.

   3.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Go to step 4.

   3.2. Increase M10-EXP by 1 and go to step 4.
4. Get a character from the input string. Let the character be C.

   If C is a '.', then go to step 5.

   If C is a 'E', then go to step 7.

   If C is the space character, then go to the start of step 4.

   If C is none of the above, then go to step 3.

5. Get a character from the input string. Let the character be C.

   If C is a ‘E’, then go to step 7.

   If C is the space character, then go to the start of step 5.

   If C is none of the above, then go to step 6.

6. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 6.1, else go to step 5.

   6.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Decrement M10-EXP by 1. Go to step 5.

7. Set SEXP, the exponent sign to be 0. Go to step 8.

8. Get a character from the input string. Let the character be C.

   If C is a ‘+’, then go to start of step 8.

   If C is a ‘−’, then set SEXP to be FF and go to the start of step 8.

   If C is none of the above, then go to step 9.

9. Set M20-EXP, the explicit decimal exponent to 0. Go to step 10.

10. Subtract 30 from C to get its integer value. Let this be I. Multiply M20-EXP by 10 and add I to the product. Store this sum back in M20-EXP. Go to step 11.

11. Get a character from the input string. Let this be C. If C is the null character, then go to step 12, else go to step 10.

12. Add M10-EXP and M20-EXP (with the proper sign as determined by SEXP) to get the 10’s exponent M-EXP. Save in M-EXP the magnitude of the sum and in SEXP the sign of the sum. Go to step 13.

13. Check and see if HI-INT.LO-INT is 0. If it is, then set the resulting floating point number, C, to zero and return. If it is not then go to step 14.

14. Normalize HI-INT.LO-INT by left shifts such that the MSB is 1. Let the number of left shifts needed to do this be L. Set B1-EXP to 32-decimal − L. Go to step 15.

15. If SEXP is 0, then set P-HI.P-LO to the binary representation of 0.625, else set P-HI.P-LO to the binary representation of 0.8. Go to step 16.

16. Multiply HI-INT.LO-INT by P-HI.P-LO M-EXP times. After each multiplication, normalize the partial product if needed by left shifting. Accumulate the number of left shifts needed in B2-EXP. Let the final product be C-HI.C-LO. Go to step 17.

17. Subtract B2-EXP from B1-EXP. Let the result be B-EXP. Go to step 18.

18. If SEXP is 0, then multiply M-EXP by 4, else multiply M-EXP by −3. Let the result be B3-EXP. Go to step 19.

19. Add B-EXP and B3-EXP. Let the result be C1-EXP. Add 126 to C1-EXP to restore the IEEE bias, getting C-EXP. Go to step 20.

20. Round C-HI.C-LO. Go to step 21.

21. Pack C-EXP.M-SIGN, C-HI and C-LO and return.

(vi) FTOA.

The processing steps are as follows.

1. Unpack the input FLP number. Let the unpacked number be represented by C-EXP.C-SIGN, C-HI and C-LO. Go to step 2.

2. Subtract 126-decimal from C-EXP to remove the IEEE bias. Let the result be C1-EXP. Go to step 3.

3. Multiply C1-EXP by the binary representation of log(2). Let the product be U-HI.U-LO. Go to step 4.

4. Subtract 8 from U-HI.U-LO. Let the magitude of the integer part of the result be V and its sign VSIGN. Go to step 5.

5. If VSIGN is 0, then set P-HI.P-LO to the binary representation of 0.8, else set P-HI.P-LO to the binary representation of 0.625. Go to step 6.

6. Multiply C-HI.C-LO by P-HI.P-LO V times. Normalize the partial product after each multiplication, if needed, by left shifting. Accumulate any left shifts needed in B1-EXP. Let the final product be HI-INT.LO-INT. Go to step 7.

7. Subtract B1-EXP from C1-EXP. Let the result be B2-EXP. Go to step 8.

8. If VSIGN is 0, then multiply V by −3, else multiply it by 4. Let the result be B3-EXP. Go to step 9.

9. Add B2-EXP and B3-EXP. Let the result be B4-EXP. Go to step 10.

10. If B4-EXP is more than 32-decimal, then increase V and go to step 6, else go to step 11.

11. If B4-EXP is less than 28-decimal, then decrease V and go to step 6, else go to step 12.

12. Subtract B4-EXP from 32. Let the result be B5-EXP. Go to step 13.

13. Shift HI-INT.LO-INT right B5-EXP number of times. Go to step 14.

14. Add 16-decimal to the address of the start of the decimal string. Output a null byte there. Go to step 15.

15. Divide V by 10-decimal. Let the quotient be Q and the remainder R. Add 30 to R and output it to the decimal string. Next add 30 to Q and output it to the decimal string. Go to step 16.

16. If VSIGN is 0, then output ‘+’ to the output string, else output ‘−’ to the output string. Go to step 17.

17. Output ‘E’ to the output string. Output ‘.’ to the output string. Go to step 18.

18. Divide C-HI.C-LO by 10-decimal 10 times. Let the remainder in each division be R. Add 30 to each R and output it to the output string. Go to step 19.

19. If C-SIGN is 0, then output the space character to the output string, else output ‘−’ to the output string. Then return to the calling program.

REFERENCES

1. ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, Aug. 12, 1985.

2. J.T. Coonen, “An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic,” IEEE Computer, Jan. 1980, pp. 68–79.

3. K. Hwang, *Computer Arithmetic,* John-Wiley and Sons, 1979.

4. M. M. Mano, *Computer System Design,* Prentice-Hall, 1980.

APPENDIX A

A USER'S MANUAL FOR THE HPC FLOATING POINT PACKAGE

The Single Precision Floating Point Package for the HPC implements the following functions.

ARITHMETIC FUNCTIONS

1. FADD—Add two floating point numbers.
2. FSUB—Subtract two floating point numbers.
3. FMULT—Multiply two floating point numbers.
4. FDIV—Divide two floating point numbers.

FORMAT CONVERSION FUNCTIONS

5. ATOF—Convert an ASCII string representing a decimal floating point number to a single precision floating point number.
6. FTOA—Convert a single precision floating point number to an ASCII string that represents the decimal floating point value of the number.

The entire package is in the form of a collection of subroutines and is contained in the following files.

1. FERR.MAC
2. FNACHK.MAC
3. FZCHK.MAC
4. FUNPAK.MAC
5. FPAK.MAC
6. FPTRAP.MAC
7. ROUND.MAC
8. BFMUL.MAC
9. ISIOK.MAC
10. MUL10.MAC
11. ATOF.MAC
12. FTOA.MAC
13. FADD.MAC
14. FMULT.MAC
15. FDIV.MAC

The first 7 files are general utility routines that are used by all the Arithmetic and Format Conversion subroutines. The next 3 files, BFMUL.MAC, ISIOK.MAC and MUL10.MAC are used only by the Format Conversion subroutines, ATOF and FTOA. Depending on the functions being used in the user program, only the necessary files need be included.

INTERFACE WITH USER PROGRAMS

1. All the Arithmetic routines expect the input to be in the IEEE Single Precision format. This format requires 2 words for the storage of each floating point number. If the required arithmetic operation is FlopF2, where op is +, −, * or /, then the routines expect that F1 is available in registers K and A on entry, with the high half in K. Also, the two words of F2 are expected to be on the stack. If SP is the stack pointer on entry into one of the Arithmetic function subroutines, then the high word of F2 should be at W(SP-4) and the low word at W(SP-6). The result of the Arithmetic operation is returned in IEEE format in registers K and A, with the high word in K.

2. The Format Conversion subroutine ATOF expects that on entry, B contains the address of the ASCII string representing the decimal floating point number. This string must be of the form

   Siiiii.ffffffEsNND

   where

   S is an optional sign for the mantissa. Thus S can be '+', '−' or not present at all.

   iiiii is the optional integer part of the mantissa. If it is present, it can be of any length, must contain only the characters '0' through '9' and must not contain any embedded blanks.

   . is the optional decimal point. It need not be present if the number has no fractional part.

   ffffff is the optional fractional part of the mantissa. ffffff, if it is present must consist of a sequence of digits '0' through '9'. It can be of any length. Note that either iiiii, the integer part or .ffffff the fractional part must be present.

   E is the required exponent start symbol.

   s is the optional sign of the exponent. If it is present, it must be '+' or '−'.

   NN is the exponent and consists of at most two decimal digits. It is required to be present.

   D is the null byte <00> and must be present to terminate the string.

   The floating point number represented by the above string is returned by ATOF in IEEE format in registers K and A.

3. The format conversion routine FTOA expects the floating point number input to be in registers K and A in the IEEE format. Register B is expected to contain the starting address of a 17 byte portion of memory where the output string will be stored.

4. Three global symbols need to be defined in the user program before assembling the user program and any included floating point package files. These symbols are:

   (i) TMP1 which must be set to 0. The package uses W(TMP1) for temporary storage.

   (ii) FPERWD which must be set to an address in the base page. The package signals floating point exceptions using W(FPERWD). This is described below.

   (iii) FPTRAP which must be set to the address of the start of a user floating point exception handler. Again this is described below.

FLOATING POINT EXCEPTS

The package maintains a history of floating point exceptions in the 4 least significant bits of the word W(FPERWD). The value of the symbol FPERWD should be defined by the user program, and should be an address in the base page. This word should also be cleared by the user program before calling any floating point routine. The word is never cleared by the floating point package, and the user program can examine this word to determine the type of exceptions that may have occurred during the course of a computation.

The following 4 types of error can occur in the course of a floating point computation.

1. Invalid Operand. This happens if one of the input numbers for an Arithmetic routine or the input for FTOA is not a valid floating point number. An invalid floating point number (or NaN) can be created either by an overflow in a previous computation step, or if the ASCII decimal floating point number input to ATOF is too large to be represented in the IEEE format. The result, if one of the inputs is a NaN is always set to a NaN.
2. Overflow. This happens if the result of a computation is too large to be represented within the exponent range available. Overflow can occur in any of the arithmetic routines or ATOF. On overflow, the result is set to a representation called NaN. An NaN is considered an illegal operand in all successive steps.
3. Underflow. This occurs if the result of a computation is too small to be represented with the precision and exponent range available. On underflow, the result is set to zero.
4. Divide-by-zero. This error occurs if F2 is zero when computing F1/F2. The result is set to an NaN.

Each of the above errors results in a bit being set in W(FPERWD). This is done as follows:

Bit 0—Set on Overflow.

Bit 1—Set on Underflow.

Bit 2—Set on Illegal Operand.

Bit 3—Set on Divide-by-zero.

One further action is taken when a floating point exception occurs. After the result has been set to the appropriate value, and the corresponding bit in W(FPERWD) set, the package does a subroutine call to address FPTRAP. The user can provide any exception handler at this address. The file FPTRAP.MAC contains the simplest possible user exception handler. It does nothing, but merely returns back to the calling program.

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP

                                     .TITLE FLP
                         LISTER:
         0071                        .LIST 071
         F000                        . = 0F000
         0002                        FPERWD = W(2)
         0000                        TMP1 = W(0)

NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     2
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
THE FLP ROUTINES

                                     .FORM 'THE FLP ROUTINES'
```

The code listed in this App Note is available on Dial-A-Helper.

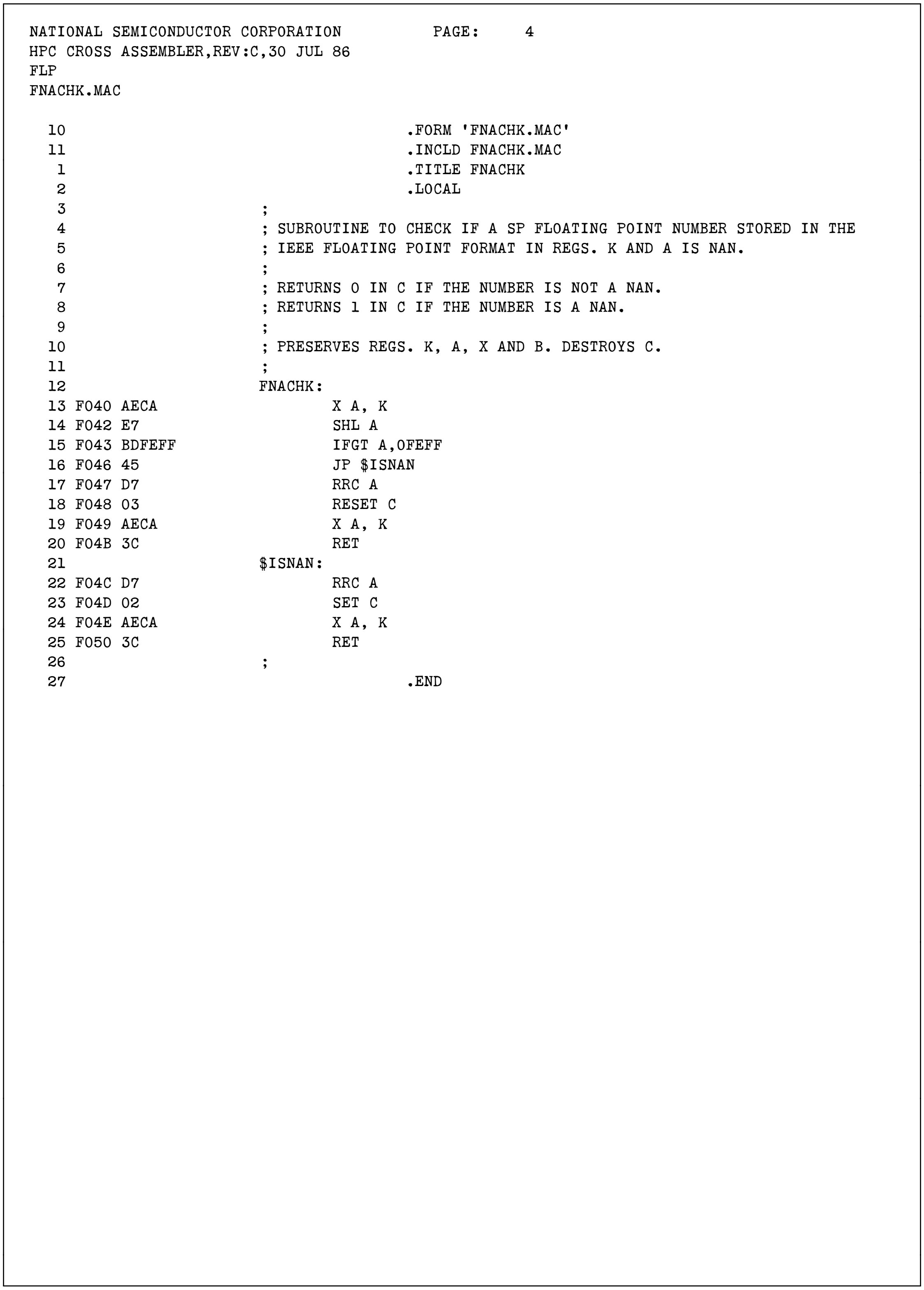
Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FERR.MAC

                                    .FORM 'FERR.MAC'
                                    .INCLD FERR.MAC
                    ; EXCEPTION HANDLING.
                    ; DIVIDE BY ZERO.
                    DIVBY0:
F000 820802FA               OR FPERWD, 08   ; SET THE DIVIDE BY 0 BIT.
F004 00                     CLR A
F005 B17F80                 LD K, 07F80
F008 3093                   JSR FPTRAP
F00R 3FCC                   POP B
F00C 3FCE                   POP X
F00E 3C                     RET
                    ; ILLEGAL OPERAND - ONE OF F1 OR F2 IS A NAN.
                    FNAN:
F00F 820402FA               OR FPERWD, 04   ; SET THE ILLEGAL OPERAND BIT.
F013 00                     CLR A
F014 B17F80                 LD K, 07F80     ; RETURN NAN IN K AND A.
F017 3084                   JSR FPTRAP      ; GO TO USER TRAP ROUTINE.
F019 3FCC                   POP B
F01B 3FCE                   POP X
F01D 3C                     RET
                    ; EXPONENT UNDERFLOW.
                    UNDFL:
F01E 820202FA               OR FPERWD, 02   ; SET THE EXPONENT UNDERFLOW BIT.
F022 00                     CLR A
F023 ACC8CA                 LD K, A
F026 3075                   JSR FPTRAP
F028 3FC4                   POP SP
F02A 3FCC                   POP B
F02C 3FCE                   POP X
F02E 3C                     RET
                    ; EXPONENT OVERFLOW.
                    OVRFL:
F02F 820102FA               OR FPERWD, 01   ; SET THE EXPONENT OVERFLOW BIT.
F033 00                     CLR A
F034 B17F80                 LD K, 07F80
F037 3064                   JSR FPTRAP
F039 3FC4                   POP SP
F03B 3FCC                   POP B
F03D 3FCE                   POP X
F03F 3C                     RET
                    ;
                            .END
```



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FNACHK.MAC

  10                                    .FORM 'FNACHK.MAC'
  11                                    .INCLD FNACHK.MAC
   1                                    .TITLE FNACHK
   2                                    .LOCAL
   3                     ;
   4                     ; SUBROUTINE TO CHECK IF A SP FLOATING POINT NUMBER STORED IN THE
   5                     ; IEEE FLOATING POINT FORMAT IN REGS. K AND A IS NAN.
   6                     ;
   7                     ; RETURNS 0 IN C IF THE NUMBER IS NOT A NAN.
   8                     ; RETURNS 1 IN C IF THE NUMBER IS A NAN.
   9                     ;
  10                     ; PRESERVES REGS. K, A, X AND B. DESTROYS C.
  11                     ;
  12                     FNACHK:
  13 F040 AECA                   X A, K
  14 F042 E7                     SHL A
  15 F043 BDFEFF                 IFGT A,0FEFF
  16 F046 45                     JP $ISNAN
  17 F047 D7                     RRC A
  18 F048 03                     RESET C
  19 F049 AECA                   X A, K
  20 F04B 3C                     RET
  21                     $ISNAN:
  22 F04C D7                     RRC A
  23 F04D 02                     SET C
  24 F04E AECA                   X A, K
  25 F050 3C                     RET
  26                     ;
  27                                    .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FNACHK
FZCHK.MAC

                                        .FORM 'FZCHK.MAC'
                                        .INCLD FZCHK.MAC
                                        .TITLE FZCHK
                                        .LOCAL
                         ;
                         ; SUBROUTINE THAT CHECKS IF A SP FLOATING POINT NUMBER STORED
                         ; IN THE IEEE FORMAT IN REGS K AND A IS ZERO.
                         ;
                         ; RETURNS 0 IN C IF THE NUMBER IS NOT ZERO.
                         ; RETURNS 1 IN C IF THE NUMBER IS ZERO.
                         ; SAVES REGS. K, A, X, AND B BUT DESTROYS C.
                         ;
                         FZCHK:
 F051 AECA                       X A, K
 F053 E7                         SHL A
 F054 9DFF                       IFGT A,0FF
 F056 45                         JP $ANOTO
 F057 D7                         RRC A
 F058 02                         SET C
 F059 AECA                       X A, K
 F05B 3C                         RET
                         $ANOTO:
 F05C D7                         RRC A
 F05D 03                         RESET C
 F05E AECA                       X A, K
 F060 3C                         RET
                         ;
                                        .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FZCHK
FUNPAK.MAC

                                         .FORM 'FUNPAK.MAC'
                                         .INCLD FUNPAK.MAC
                                         .TITLE FUNPAK
                                         .LOCAL
                         ;
                         ; SUBROUTINE TO UNPACK A SP FLOATING POINT NUMBER STORED IN THE
                         ; IEEE FORMAT IN REGS. K AND A. THE UNPACKED FORMAT OCCUPIES 3
                         ; WORDS AND IS ORGANIZED AS FOLLOWS:
                         ;                       |                |
                         ; increasing addrs |    |                |             <- X on exit
                         ;                       |EEEEEEEESSSSSSSS| FEXP-FSIGN
                         ;                       |MMMMMMMMMMMMMMMM| FHI
                         ;                       |MMMMMMMM00000000| FLO          <- X on entry
                         ;                       |                |
                         ;
                         ; EEEEEEEE - 8 BIT EXPONENT IN EXCESS-127 FORMAT
                         ; SSSSSSSS - SIGN BIT < 00 -> +, FF -> ->
                         ; M ... M - 24 BITS OF MANTISSA. NOTE THAT IMPLIED 1 IS PRESENT HERE.
                         ;
                         ; ON ENTRY TO THE SUBROUTINE X SHOULD POINT TO FLO. ON EXIT, X POINTS
                         ; TO THE WORD AFTER FSIGN.
                         ; REGS. K, A AND B ARE DESTROYED BY THIS SUBROUTINE.
                         ;
                         FUNPAK:
     F061 ABCC                   ST A,B           ; SAVE A IN B.
     F063 00                     CLR A
     F064 D1                     X A, M(X+)       ; ZERO LOW BYTE OF FLO.
     F065 88CC                   LD A, L(B)
     F067 D1                     X A, M(X+)       ; MOVE LOW BYTE OF F-R0 INTO HIGH BYTE OF FLO.
     F068 88CD                   LD A, H(B)
     F06A D1                     X A, M(X+)       ; MOVE MID BYTE OF MANT INTO LOW BYTE OF FHI.
     F06B A8CA                   LD A, K
     F06D 96C80F                 SET A.7          ; SET IMPLIED 1 IN MANTISSA
     F070 D1                     X A, M(X+)       ; MOVE HIGH BYTE OF MANT INTO HIGH BYTE OF FHI.
     F071 A8CA                   LD A, K
     F073 E7                     SHL A            ; SIGN BIT TO CARRY.
     F074 B9FF00                 AND A, 0FF00     ; ZERO SIGN.
     F077 07                     IF C
     F078 9AFF                   OR A, 0FF        ; PUT SIGN BACK IF -.
     F07A F1                     X A, W(X+)       ; SAVE FEXP-FSIGN.
     F07B 3C                     RET
                         ;
                                         .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FUNPAK
FPAK.MAC

  16                                        .FORM 'FPAK.MAC'
  17                                        .INCLD FPAK.MAC
   1                                        .TITLE FPAK
   2                                        .LOCAL
   3                         ;
   4                         ; SUBROUTINE TO PACK A SP FLOATING POINT NUMBER STORED IN THE
   5                         ; 3 WORD FEXP-FSIGN/FHI/FLO FORMAT INTO THE IEEE FORMAT IN REGS.
   6                         ; K AND A.
   7                         ;
   8                         ; ON ENTRY TO THE SUBROUTINE, X POINTS TO FLO. ON EXIT, X POINTS
   9                         ; TO THE WORD AFTER FSIGN.
  10                         ;
  11                         ; REGS. K, A AND B ARE DESTROYED.
  12                         ;
  13                         FPAK:
  14 F07C D1                         X A, M(X+)       ; GET RID OF ZERO LOW BYTE OF FLO.
  15 F07D D1                         X A, M(X+)       ; GET HIGH BYTE OF FLO.
  16 F07E ABCA                       ST A, K          ; STORE IT IN K.
  17 F080 D1                         X A, M(X+)       ; GET LOW BYTE OF FHI.
  18 F081 3B                         SWAP A
  19 F082 3B                         SWAP A
  20 F083 B9FF00                     AND A, 0FF00     ; SHIFT LEFT 8 TIMES.
  21 F086 A0C8CAFA                   OR K, A          ; LOW WORD OF RESULT IS NOW IN K.
  22                         ;
  23 F08A D1                         X A, M(X+)       ; GET HIGH BYTE OF FHI.
  24 F08B 96C81F                     RESET A.7        ; ZERO IMPLIED MSB 1 IN MANT.
  25 F08E ABCC                       ST A,B           ; SAVE IN REG. B.
  26 F090 D4                         LD A, M(X)       ; GET SIGN BYTE FROM ASIGN.
  27 F091 C7                         SHR A            ; MOVE 1 SIGN BIT INTO CARRY.
  28 F092 F0                         LD A, W(X+)      ; GET FEXP-FSIGN.
  29 F093 B9FF00                     AND A, 0FF00     ; ZERO SIGN.
  30 F096 D7                         RRC A            ; MOVE RIGHT 1 BIT. SIGN BIT FROM C
  31                                                  ; ENTERS INTO THE MSB.
  32 F097 96CCFA                     OR A, B          ; GET MANT BITS IN FROM B.
  33 F09A AECA                       X A, K           ; SWAP A AND K
  34 F09C 3C                         RET
  35                         ;
  36                                        .END


NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     8
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPAK
FPTRAP.MAC

  18                                        .FORM 'FPTRAP.MAC'
  19                                        .INCLD FPTRAP.MAC
   1                                        .TITLE FPTRAP
   2                         ; USER SUPPLIED FP TRAP ROUTINE.
   3                         FPTRAP:
   4 F09D 3C                         RET
   5                         ;
   6                                        .END
```

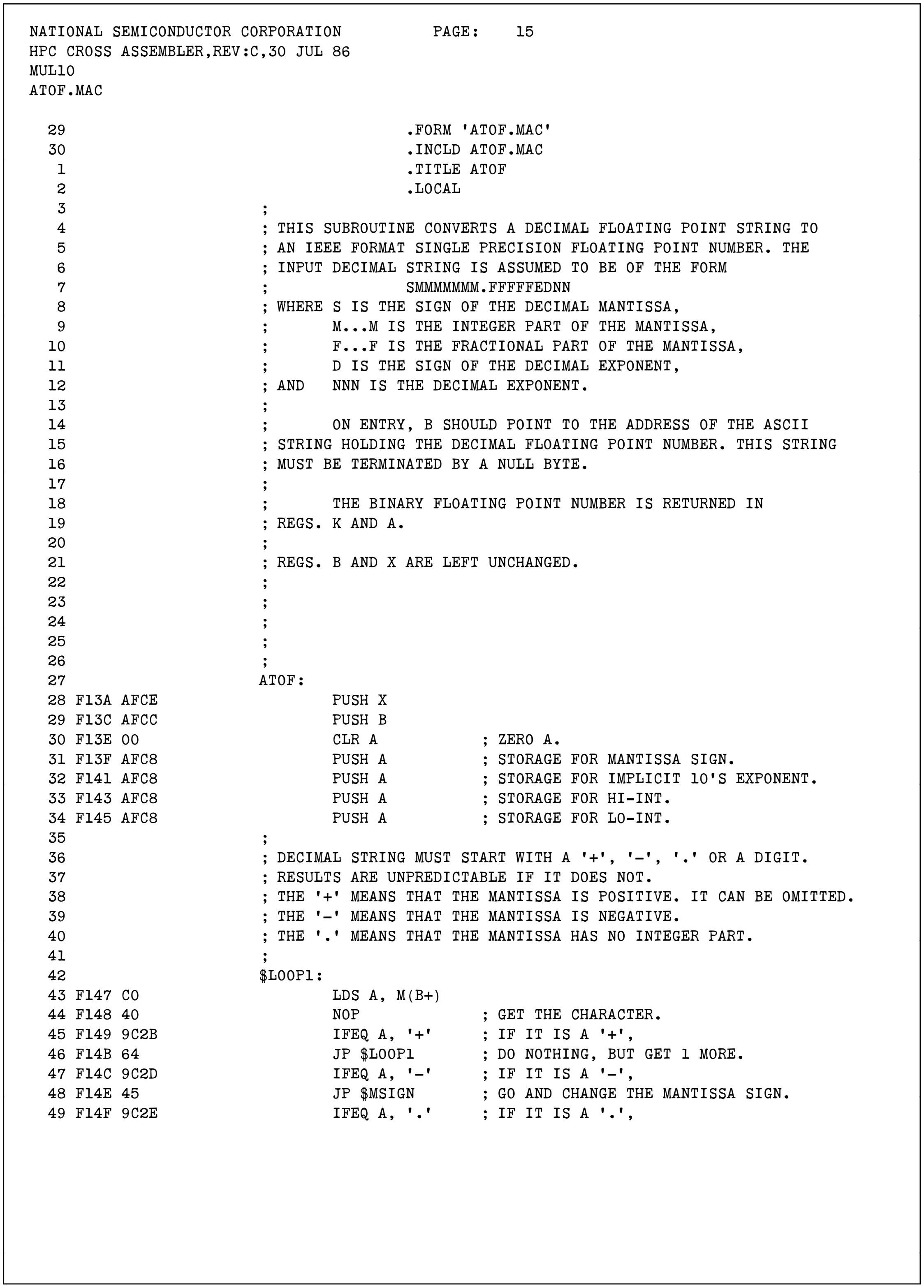
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     9
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPTRAP
ROUND.MAC

  20                                    .FORM 'ROUND.MAC'
  21                                    .INCLD ROUND.MAC
   1                                    .TITLE SROUND
   2                                    .LOCAL
   3                        ;
   4                        ; THIS SUBROUTINE IS USED TO ROUND THE 32 BIT MANTISSA OBTAINED
   5                        ; IN THE FLOATING POINT CALCULATIONS TO 24 BITS.
   6                        ;
   7                        ; THE UNPACKED FLOATING POINT NUMBER SHOULD BE STORED IN
   8                        ; CONSECUTIVE WORDS OF MEMORY. ON ENTRY, X SHOULD CONTAIN
   9                        ; THE ADDRESS OF C-HI. C-EXP.C-SIGN IS AT W(X+2) AND
  10                        ; C-LO IS AT W(X-2).
  11                        ;
  12                        ; ON EXIT X HAS THE ADDRESS OF C-EXP.C-SIGN.
  13                        SROUND:
  14 F09E F2                        LD A, W(X-)      ; REMEMBER X POINTS TO C-HI.
  15 F09F F4                        LDA, W(X)        ; LOAD C-LO.
  16 F0A0 96C817                    IF A.7           ; IF BIT 25 OF MANTISSA IS 1,
  17 F0A3 43                        JP $RNDUP        ; THEN NEED TO INCREASE MANTISSA.
  18 F0A4 F0                        LD A, W(X+)
  19 F0A5 F0                        LD A, W(X+)      ; X NOW POINTS TO C-EXP.C-SIGN.
  20 F0A6 5F                        JP $EXIT         ; DONE, SO GET OUT.
  21                        ; INCREASE MANTISSA.
  22                        $RNDUP:
  23 F0A7 B80100                    ADD A, 0100
  24 F0AA F1                        X A, W(X+)       ; INCREASE LOW BYTE BY 1.
  25 F0AB 07                        IF C             ; IF THERE IS A CARRY,
  26 F0AC 42                        JP $HIUP         ; THEN NEED TO INCREASE C-HI.
  27 F0AD F0                        LD A, W(X+)      ; X NOW POINTS TO C-EXP.C-SIGN.
  28 F0AE 57                        JP $EXIT         ; DONE, SO GET OUT.
  29                        ; MANTISSA INCREASE PROPAGATING TO HIGH WORD.
  30                        $HIUP:
  31 F0AF F4                        LD A, W(X)
  32 F0B0 B8                        .BYTE 0B8,00,01  ; DO ADD A, 01 BUT WITH WORD CARRY!!
     F0B1 00
     F0B2 01
  33 F0B3 07                        IF C             ; IF THERE IS A CARRY,
  34 F0B4 42                        JP $EXIN2        ; THEN NEED TO INCREASE EXPONENT.
  35 F0B5 F1                        X A,W(X+)
  36 F0B6 4F                        JP $EXIT         ; GET OUT.
  37                        ; ROUND UP LEADS TO EXPONENT INCREASE.
  38                        $EXIN2:
  39 F0B7 D7                        RRC A            ; CARRY->MSB, LSB->CARRY.
  40 F0B8 F3                        X A,W(X-)
  41 F0B9 F4                        LD A,W(X)        ; LOW WORD IS NOW IN A.
  42 F0BA D7                        RRC A
  43 F0BB F1                        X A, W(X+)
  44 F0BC F0                        LD A, W(X+)      ; X NOW POINTS TO C-EXP.CSIGN.
  45 F0BD F4                        LD A, W(X)
  46 F0BE B80100                    ADD A, 0100
  47 F0C1 07                        IF C
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    10
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
ROUND.MAC

F0C2 BAFF00                 OR A, 0FF00     ; MAKE IT A NAN.
F0C5 F6                     ST A, W(X)
                    ;
                    $EXIT:
F0C6 3C                     RET
                    ;
                                    .END
```

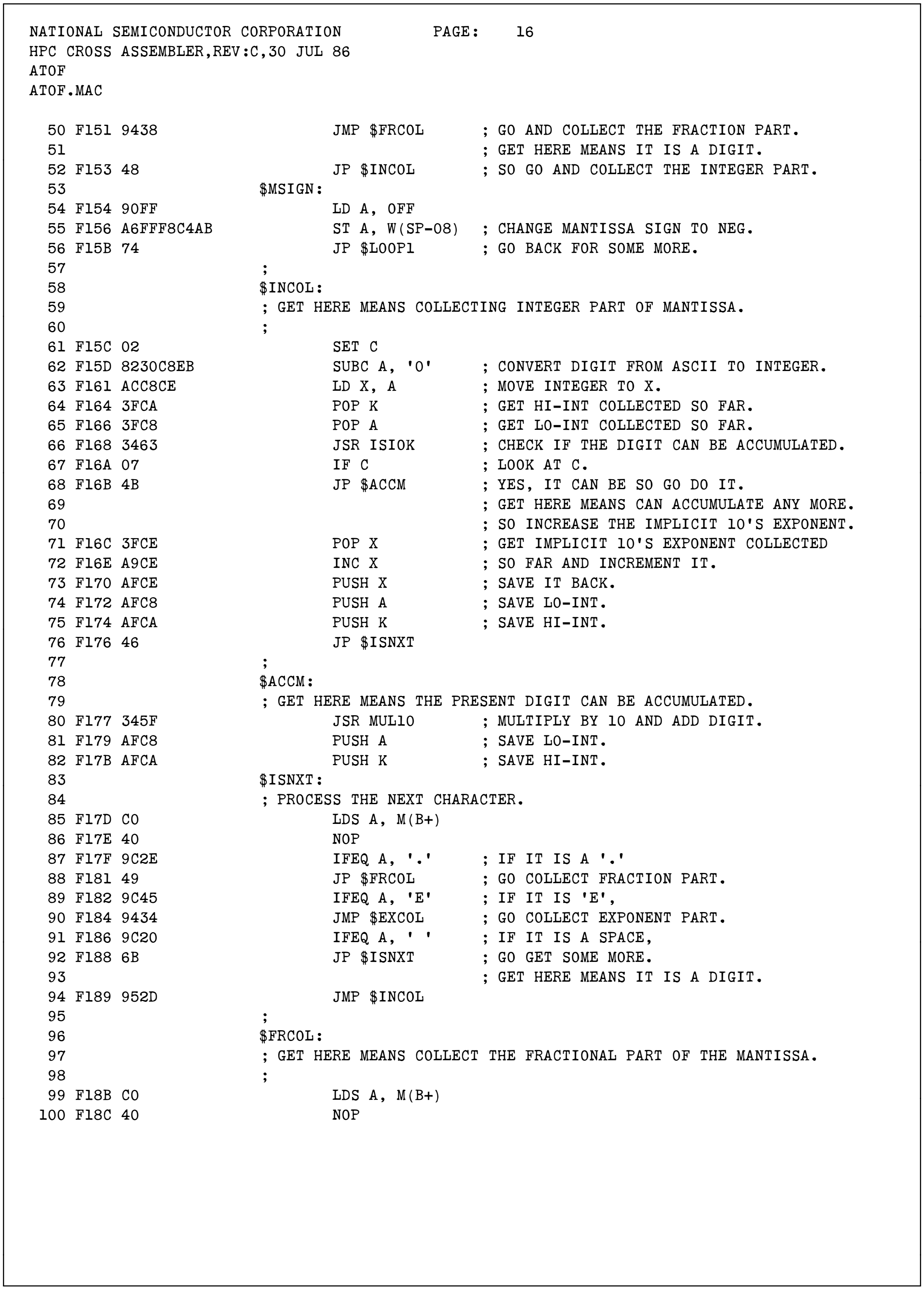
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    11
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
BFMUL.MAC

  22                                    .FORM 'BFMUL.MAC'
  23                                    .INCLD BFMUL.MAC
   1                                    .TITLE BFMUL
   2                        ;
   3                        ; THIS SUBROUTINE IS USED TO MULTIPLY TWO 32 BIT FIXED POINT FRACTIONS.
   4                        ; THE ASSUMED BINARY POINT IS TO THE IMMEDIATE LEFT OF THE MSB.
   5                        ;
   6                        ; THE FIRST FRACTION IS STORED IN REGS K AND A, WITH THE MORE
   7                        ; SIGNIFICANT WORD BEING IN K.
   8                        ;
   9                        ; THE SECOND FRACTION IS STORED ON THE STACK. THE MORE SIGNIFICANT
  10                        ; WORD IS AT W(SP-4) AND THE LOWER SIGNIFICANT WORD
  11                        ; IS IN THE WORD BELOW IT.
  12                        ;
  13                        ; THE 32 BIT PRODUCT IS LEFT IN REGS. K AND A, WITH THE MORE
  14                        ; SIGNIFICANT WORD BEING IN K.
  15                        ;
  16                        ; IMPORTANT NOTE : THE FRACTIONS ARE ASSUMED TO BE UNSIGNED.
  17                        ;
  18                        ; REGS. B AND X ARE UNCHANGED.
  19                        ;
  20                        BFMUL:
  21 F0C7 AFCE                      PUSH X          ; SAVE X.
  22 F0C9 AFC8                      PUSH A          ; SAVE F1-LO
  23 F0CB AFCA                      PUSH K          ; SAVE F1-HI.
  24 F0CD ABCA                      LD A, K         ; MOVE F1-HI TO A.
  25 F0CF A6FFF6C4FE                MULT A, W(SP-0A); MULTIPLY F1-HI BY F2-HI.
  26 F0D4 3FCA                      POP K           ; GET FI-HI.
  27 F0D6 AFCE                      PUSH X          ; SAVE PR-HI.
  28 F0D8 AFC8                      PUSH A          ; SAVE PR-LO.
  29 F0DA A8CA                      LD A, K         ; MOVE F1-HI TO A.
  30 F0DC A6FFF2C4FE                MULT A, W(SP-0E); MULTIPLY F1-HI BY F2-LO.
  31 F0E1 3FC8                      POP A           ; GET PR-LO SAVED. NOTE THAT THE
  32                                                ; LO WORD OF THIS PRODUCT IS DISCARDED.
  33 F0E3 3FCA                      POP K           ; GET PR-HI SAVED.
  34 F0E5 96CEF8                    ADD A, X        ; ADD TO PR-LO THE HI WORD OF THIS PRODUCT.
  35 F0E8 07                        IF C            ; ON CARRY,
  36 F0E9 A9CA                      INC K           ; PROPAGATE THRU TO PR-HI.
  37 F0EB 3FCE                      POP X           ; GET F1-LO.
  38 F0ED AFCA                      PUSH K          ; SAVE PR-HI.
  39 F0EF AFC8                      PUSH A          ; SAVE PR-LO.
  40 F0F1 A8CE                      LD A, X         ; MOVE F1-LO TO A.
  41 F0F3 A6FFF6C4FE                MULT A, W(SP-0A); MULTIPLY BY F2-HI.
  42 F0F8 3FC8                      POP A           ; GET PR-LO SAVED.
  43 F0FA 3FCA                      POP K           ; GET PR-HI SAVED.
  44 F0FC 96CEF8                    ADD A, X        ; ADD TO PR-LO THE HI-WORD OF THIS PRODUCT.
  45 F0FF 07                        IF C
  46 F100 A9CA                      INC K           ; PROPAGATE ANY CARRY TO PR-HI.
  47 F102 3FCE                      POP X           ; RESTORE X.
  48 F104 3C                        RET
  49                        ;
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    12
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
BFMUL.MAC

  50                                     .END



NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    13
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
ISIOK.MAC

  24                                     .FORM 'ISIOK.MAC'
  25                                     .INCLD ISIOK.MAC
   1                                     .TITLE ISIOK
   2                                     .LOCAL
   3                         ;
   4                         ; THIS SUBROUTINE IS USED TO DETERMINE IF ANOTHER DECIMAL DIGIT CAN
   5                         ; BE ACCUMULATED IN THE 32 BIT INTEGER STORED IN REGS. K AND A.
   6                         ; THE MORE SIGNIFICANT WORD IS IN K.
   7                         ; SETS THE CARRY TO 1 IF IT CAN BE ACCUMULATED; RESETS THE CARRY
   8                         ; OTHERWISE. PRESERVES ALL REGS.
   9                         ;
  10                         ISIOK:
  11 F105 02                         SET C
  12 F106 861999CAFC                 IFEQ K, 01999
  13 F10B 47                         JP $CHKOT
  14 F10C 861999CAFD                 IFGT K, 01999
  15 F111 03                         RESET C
  16 F112 3C                         RET
  17 F113 BD9998             $CHKOT: IFGT A, 09998
  18 F116 03                         RESET C
  19 F117 3C                         RET
  20                         ;
  21                                     .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    14
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ISIOK
MUL10.MAC

                                    .FORM 'MUL10.MAC'
                                    .INCLD MUL10.MAC
                                    .TITLE MUL10
                                    .LOCAL
                    ;
                    ; THIS SUBROUTINE MULTIPLIES THE 32 BIT INTEGER STORED IN REGS K AND A
                    ; BY 10-DECIMAL AND ADDS TO IT THE INTEGER STORED IN X.
                    ; THE RESULT IS RETURNED IN K AND A.
                    ; REGS. B AND X ARE NOT CHANGED.
                    ;
                    MUL10:
F118 AFCE                   PUSH X          ; SAVE INTEGER.
F11A AFC8                   PUSH A          ; SAVE LONG INT-LO.
F11C A8CA                   LD A, K
F11E 9E0A                   MULT A, 0A      ; MULT LONG INT-HI BY 10.
F120 AFC8                   PUSH A          ; SAVE LOW WORD OF PRODUCT.
F122 A6FFFCC4A8             LD A, W(SP-4)   ; GET LONG INT-LO.
F127 9E0A                   MULT A, 0A
F129 3FCA                   POP K           ; GET LO WORD OF LAST PRODUCT.
F12B A0CECAF8               ADD K, X        ; ADD TO IT HI WORD OF THIS PRODUCT.
F12F 3FCE                   POP X           ; GET RID OF GARBAGE.
F131 3FCE                   POP X           ; GET INTEGER TO BE ADDED.
F133 96CEF8                 ADD A, X
F136 07                     IF C
F137 A9CA                   INC K
F139 3C                     RET
                    ;
                                    .END
                    ;
```



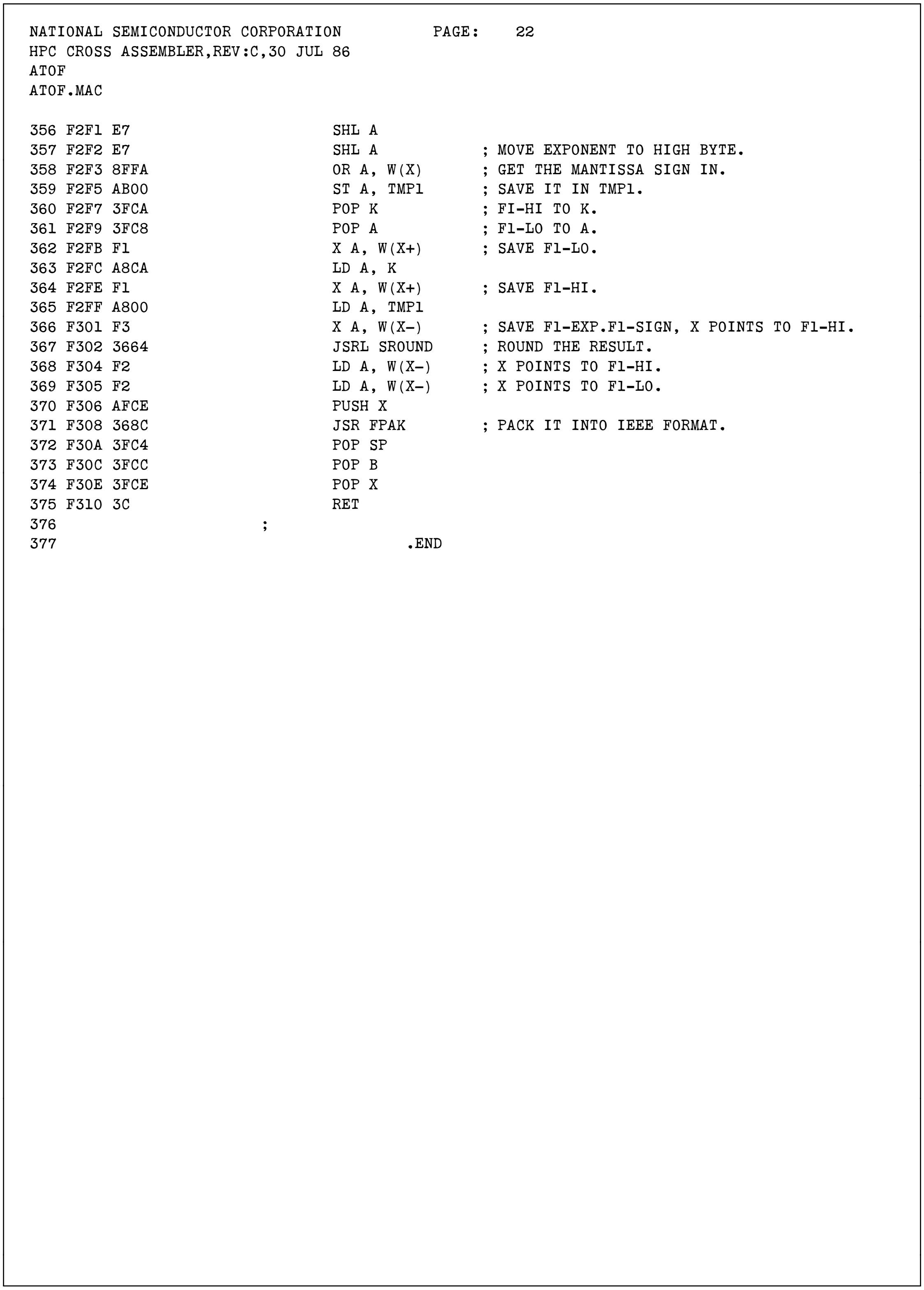
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    15
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
MUL10
ATOF.MAC

                                      .FORM 'ATOF.MAC'
                                      .INCLD ATOF.MAC
                                      .TITLE ATOF
                                      .LOCAL
                         ;
                         ; THIS SUBROUTINE CONVERTS A DECIMAL FLOATING POINT STRING TO
                         ; AN IEEE FORMAT SINGLE PRECISION FLOATING POINT NUMBER. THE
                         ; INPUT DECIMAL STRING IS ASSUMED TO BE OF THE FORM
                         ;              SMMMMMMM.FFFFFEDNN
                         ; WHERE S IS THE SIGN OF THE DECIMAL MANTISSA,
                         ;       M...M IS THE INTEGER PART OF THE MANTISSA,
                         ;       F...F IS THE FRACTIONAL PART OF THE MANTISSA,
                         ;       D IS THE SIGN OF THE DECIMAL EXPONENT,
                         ; AND   NNN IS THE DECIMAL EXPONENT.
                         ;
                         ;       ON ENTRY, B SHOULD POINT TO THE ADDRESS OF THE ASCII
                         ; STRING HOLDING THE DECIMAL FLOATING POINT NUMBER. THIS STRING
                         ; MUST BE TERMINATED BY A NULL BYTE.
                         ;
                         ;       THE BINARY FLOATING POINT NUMBER IS RETURNED IN
                         ; REGS. K AND A.
                         ;
                         ; REGS. B AND X ARE LEFT UNCHANGED.
                         ;
                         ;
                         ;
                         ;
                         ;
                         ATOF:
   F13A AFCE                     PUSH X
   F13C AFCC                     PUSH B
   F13E 00                       CLR A          ; ZERO A.
   F13F AFC8                     PUSH A         ; STORAGE FOR MANTISSA SIGN.
   F141 AFC8                     PUSH A         ; STORAGE FOR IMPLICIT 10'S EXPONENT.
   F143 AFC8                     PUSH A         ; STORAGE FOR HI-INT.
   F145 AFC8                     PUSH A         ; STORAGE FOR LO-INT.
                         ;
                         ; DECIMAL STRING MUST START WITH A '+', '-', '.' OR A DIGIT.
                         ; RESULTS ARE UNPREDICTABLE IF IT DOES NOT.
                         ; THE '+' MEANS THAT THE MANTISSA IS POSITIVE. IT CAN BE OMITTED.
                         ; THE '-' MEANS THAT THE MANTISSA IS NEGATIVE.
                         ; THE '.' MEANS THAT THE MANTISSA HAS NO INTEGER PART.
                         ;
                         $LOOP1:
   F147 C0                       LDS A, M(B+)
   F148 40                       NOP            ; GET THE CHARACTER.
   F149 9C2B                     IFEQ A, '+'    ; IF IT IS A '+',
   F14B 64                       JP $LOOP1      ; DO NOTHING, BUT GET 1 MORE.
   F14C 9C2D                     IFEQ A, '-'    ; IF IT IS A '-',
   F14E 45                       JP $MSIGN      ; GO AND CHANGE THE MANTISSA SIGN.
   F14F 9C2E                     IFEQ A, '.'    ; IF IT IS A '.',
```



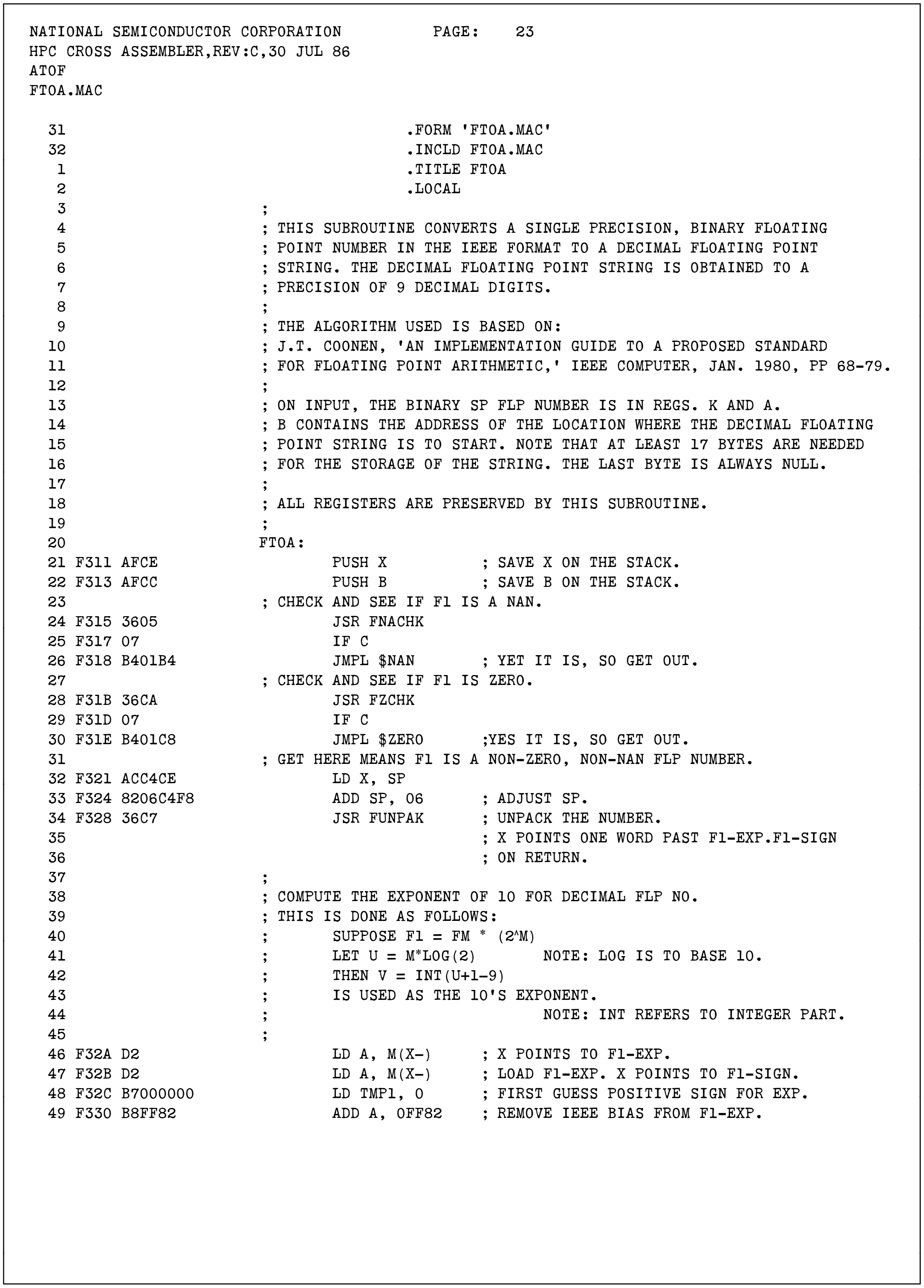
```
F151 9438                   JMP $FRCOL      ; GO AND COLLECT THE FRACTION PART.
                                            ; GET HERE MEANS IT IS A DIGIT.
F153 48                     JP $INCOL       ; SO GO AND COLLECT THE INTEGER PART.
                    $MSIGN:
F154 90FF                   LD A, OFF
F156 A6FFF8C4AB             ST A, W(SP-08)  ; CHANGE MANTISSA SIGN TO NEG.
F15B 74                     JP $LOOP1       ; GO BACK FOR SOME MORE.
                    ;
                    $INCOL:
                    ; GET HERE MEANS COLLECTING INTEGER PART OF MANTISSA.
                    ;
F15C 02                     SET C
F15D 8230C8EB               SUBC A, '0'     ; CONVERT DIGIT FROM ASCII TO INTEGER.
F161 ACC8CE                 LD X, A         ; MOVE INTEGER TO X.
F164 3FCA                   POP K           ; GET HI-INT COLLECTED SO FAR.
F166 3FC8                   POP A           ; GET LO-INT COLLECTED SO FAR.
F168 3463                   JSR ISIOK       ; CHECK IF THE DIGIT CAN BE ACCUMULATED.
F16A 07                     IF C            ; LOOK AT C.
F16B 4B                     JP $ACCM        ; YES, IT CAN BE SO GO DO IT.
                                            ; GET HERE MEANS CAN ACCUMULATE ANY MORE.
                                            ; SO INCREASE THE IMPLICIT 10'S EXPONENT.
F16C 3FCE                   POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED
F16E A9CE                   INC X           ; SO FAR AND INCREMENT IT.
F170 AFCE                   PUSH X          ; SAVE IT BACK.
F172 AFC8                   PUSH A          ; SAVE LO-INT.
F174 AFCA                   PUSH K          ; SAVE HI-INT.
F176 46                     JP $ISNXT
                    ;
                    $ACCM:
                    ; GET HERE MEANS THE PRESENT DIGIT CAN BE ACCUMULATED.
F177 345F                   JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
F179 AFC8                   PUSH A          ; SAVE LO-INT.
F17B AFCA                   PUSH K          ; SAVE HI-INT.
                    $ISNXT:
                    ; PROCESS THE NEXT CHARACTER.
F17D C0                     LDS A, M(B+)
F17E 40                     NOP
F17F 9C2E                   IFEQ A, '.'     ; IF IT IS A '.'
F181 49                     JP $FRCOL       ; GO COLLECT FRACTION PART.
F182 9C45                   IFEQ A, 'E'     ; IF IT IS 'E',
F184 9434                   JMP $EXCOL      ; GO COLLECT EXPONENT PART.
F186 9C20                   IFEQ A, ' '     ; IF IT IS A SPACE,
F188 6B                     JP $ISNXT       ; GO GET SOME MORE.
                                            ; GET HERE MEANS IT IS A DIGIT.
F189 952D                   JMP $INCOL
                    ;
                    $FRCOL:
                    ; GET HERE MEANS COLLECT THE FRACTIONAL PART OF THE MANTISSA.
                    ;
F18B C0                     LDS A, M(B+)
F18C 40                     NOP
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    17
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

101 F18D 9C45                   IFEQ A, 'E'     ; IF IT IS A 'E',
102 F18F 9429                   JMP $EXCOL      ; GO COLLECT EXPONENT.
103 F191 9C20                   IFEQ A, ' '     ; IF IT IS SPACE,
104 F193 68                     JP $FRCOL       ; GO GET SOME MORE.
105                                             ; GET HERE MEANS IT IS A DIGIT.
106 F194 D2                     SET C
107 F195 8230C8EB               SUBC A, '0'     ; GET INTEGER FROM DIGIT.
108 F199 ACC8CE                 LD X, A         ; SAVE IT IN A.
109 F19C 3FCA                   POP K           ; GET HI-INT.
110 F19E EFC8                   POP A           ; GET LO-INT.
111 F1A0 349B                   JSR ISIOK       ; CHECK IF IT CAN BE ACCUMULATED.
112 F1A2 07                     IF C
113 F1A3 45                     JP $ACCF        ; YES, SO GO DO IT.
114                                             ; GET HERE MEANS CAN'T COLLECT MORE DIGITS.
115 F1A4 AFC8                   PUSH A
116 F1A6 AFCA                   PUSH K
117 F1A8 7D                     JP $FRCOL       ; SO JUST IGNORE IT.
118                     ;
119                     $ACCF:
120                     ; ACCUMULATE THE FRACTIONAL DIGIT.
121 F1A9 3491                   JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
122 F1AB 3FCE                   POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED SO FAR,
123 F1AD 86FFFFCEF8             ADD X, 0FFFF    ; AND DECREMENT IT BY 1.
124 F1B2 AFCE                   PUSH X          ; SAVE IT BACK.
125 F1B4 AFC8                   PUSH A          ; SAVE LO-INT.
126 F1B6 AFCA                   PUSH K          ; SAVE HI-INT.
127 F1B8 952D                   JMP $FRCOL      ; GO GET SOME MORE.
128                     ;
128                     $EXCOL:
130                     ; GET HERE MEANS THE EXPLICIT 10'S EXPONENT NEEDS TO BE
131                     ; COLLECTED FROM THE STRING.
132 F1BA 03                     RESET C         ; MAKE EXPONENT SIGN POST.
133                     $EXCHR:
134 F1BB C0                     LDS A, M(B+)
135 F1BC 40                     MOP
136 F1BD 9C2B                   IFEQ A, '+'     ; IF IT IS A '+',
137 F1BF 64                     JP $EXCHR       ; GET SOME MORE.
138 F1C0 9C2D                   IFEQ A, '-',    ; IF IT IS A '-',
139 F1C2 44                     JP $ESIGN       ; GO FIX EXPONENT SIGN.
140 F1C3 9C20                   IFEQ A, ' '     ; IF IT IS SPACE,
141 F1C5 6A                     JP $EXCHR       ; GO GET SOME MORE.
142                                             ; GET HERE MEANS IT IS A DIGIT.
143 F1C6 42                     JP $EXACC       ; SO GO COLLECT THE EXPONENT.
144                     $ESIGN:
145 F1C7 02                     SET C
146 F1C8 6D                     JP $EXCHR
147                     ;
148                     $EXACC:
149                     ; ACCUMULATE THE EXPLICIT 10'S EXPONENT.
150 F1C9 9100                   LD K, 0
151 F1CB 07                     IF C
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    18
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F1CC 91FF                    LD K, OFF       ; GET SIGN BITS SET.
F1CE AFCA                    PUSH K          ; SAVE EXPLICIT EXPONENTS SIGN.
F1D0 9300                    LD X, 0
F1D2 AFCE                    PUSH X          ; ZERO EXPLICIT EXPONENT COLLECTED SO FAR.
                     $EXCLP:
F1D4 02                      SET C
F1D5 8230C8EB                SUBC A, '0'     ; GET INTEGER FROM ASCII DIGIT.
F1D9 ACC8CE                  LD X, A
F1DC 3FC8                    POP A           ; GET EXPLICIT EXPONENT COLLECTED SO FAR.
F1DE A0C8CEF8                ADD X, A        ; X CONTAINS DIGIT + EXP.
F1E2 A0C8CEF8                ADD X, A        ; X CONTAINS DIGIT + 2*EXP.
F1E6 E7                      SHL A
F1E7 E7                      SHL A
F1E8 E7                      SHL A           ; A CONTAINS 8*EXP.
F1E9 96CEF8                  ADD A, X        ; A CONTAINS DIGIT + 10*EXP.
F1EC AFC8                    PUSH A          ; SAVE BACK ON STACK.
F1EE C0                      LDS A, M(B+)
F1EF 40                      NOP             ; GET NEXT CHAR.
F1F0 9C00                    IFEQ A, 0       ; IS IT A NULL ?
F1F2 41                      JP $A10EX       ; YES SO ADD EXPLICIT AND IMPLICIT
                                             ; 10'S EXPONENT.
                                             ; GET HERE MEANS IT IS A DIGIT,
F1F3 7F                      JP $EXCLP       ; SO GO BACK AND ACCUMULATE IT.
                     ;
                     $A10EX:
                     ; DONE COLLECTING DIGITS. ADD THE EXPLICIT AND IMPLICIT
                     ; 10'S EXPONENT COLLECTED SO FAR.
F1F4 3FC8                    POP A           ; GET EXPLICIT 10'S EXPONENT.
F1F6 3FCA                    POP K           ; GET ITS SIGN.
F1F8 8200CAFC                IFEQ K, 0       ; IS IT POSITIVE ?
F1FC 42                      JP $ADDEX       ; YES SO ADD 'EM.
F1FD 01                      COMP A
F1FE 04                      INC A           ; CHANGE TO 2'S COM.
                     $ADDEX:
F1FF AGFFFAC4F8              ADD A, W(SP-06) ; ADD IMPLICIT EXPONENT.
F204 BD7FFF                  IFGT A, 07FFF   ; IS IT NEGATIVE ?
F207 43                      JP $NEG10       ; YES, CHANGE IT.
F208 9300                    LD X, 0         ; LOAD POST. SIGN IN X.
F20A 44                      JP $ESAVE
                     $NEG10
F20B 93FF                    LD, X, OFF      ; LOAD NEG. SIGN IN X.
F20D 01                      COMP A
F20E 04                      INC A           ; MAKE IT POSITIVE.
                     $ESAVE:
F20F ACC8CC                  LD B, A         ; SAVE 10'S EXPONENT IN A.
F212 3FC8                    POP A           ; GET HI-INT.
F214 3FCA                    POP K           ; GET LO-INT.
F216 AFCE                    PUSH X          ; SAVE SIGN OF 10'S EXPONENT.
F218 AFCC                    PUSH B          ; AND ITS VALUE.
                     ;
                     ; NOW CONVERT HI-INT.LO-INT TO A NORMALIZED FLOATING POINT
                     ; NUMBER. THE BINARY EXPONENT IS COLLECTED IN B.
                     ;
F21A 9000                    IFGT A, 0       ; IF HI-INT IS NOT 0,
F21C 58                      JP $NORM2       ; NEED TO SHIFT K AND A.
F21D 8200CAFD                IFGT K, 0       ; IF HI-INT IS 0, BUT NOT LO-INT,
F221 4E                      JP $NORM1       ; NEED TO SHIFT ONLY K.
                                             ; GET HERE MEANS MANTISSA IS 0.
F222 00                      CLR A
F223 ACC8CA                  LD K, A
F226 02                      SET C
F227 8208C4EB                SUB SP, 08      ; ADJUST SP. DONE!!!
F22B 3FCC                    POP B
F22D 3FCE                    POP X
F22F 3C                      RET
                     ;
                     $NORM1:
                     ; HI-INT IS 0, SO WORK WITH LO-INT ONLY.
F230 AECA                    X A, K
F232 9210                    LD B, 010       ; LOAD 16 INTO EXPONENT COUNTER.
F234 42                      JP $NRLUP
                     ;
                     $NORM2:
                     ; HI-INT IS NOT 0, SO NEED TO HANDLE BOTH.
F235 9220                    LD B, 020       ; LOAD 32 INTO LOOP COUNTER.
                     $NRLUP:
F237 E7                      SHL A
F238 07                      IF C            ; DID A 1 COME OUT ?
F239 4D                      JP $NRDUN       ; YES IT IS NORMALIZED NOW.
F23A AECA                    X A, K
F23C E7                      SHL A
F23D 07                      IF C
F23E 96CA08                  SET K.0
F241 AECA                    X A, K
F243 AACC                    DECSZ B
F245 40                      NOP             ; SHOULD NEVER BE SKIPPED!!
F246 6F                      JP $NRLUP
                     $NRDUN:
F247 D7                      RRC A           ; RESTORE SHIFTED 1.
F248 AB00                    ST A, TMP1      ; STORE IN W(0).
F24A 3FCE                    POP X           ; GET 10'S EXPONENT.
F24C 3FC8                    POP A           ; GET 10'S EXPONENT SIGN.
F24E AE00                    X A, TMP1       ; A IS HI-INT ONCE MORE.
F250 AFCC                    PUSH B          ; SAVE BINARY EXPONENT.
F252 AFCE                    PUSH X          ; SAVE 10'S EXPONENT.
F254 AECA                    X A, K          ; HI-INT TO K, LO-INT TO A.
F256 960010                  IF TMP1.0       ; IS 10'S EXPONENT NEGATIVE ?
F259 58                      JP $DIV10       ; YES, GO TO DIVIDE BY 10.
                     ; GET HERE MEANS 10'S EXPONENT IS POSITIVE, SO MULTIPLY BY 10.
                     ; ACTUALLY, WHAT IS USED IS
                     ;       10^N = (0.625*(2^4)^N
                     ; SO MULTIPLY BY 0.625 NOW AND TAKE CARE OF 2^(4*N) LATER.
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    20
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F25A A4F26ACCAB              LD B, W($MTLO)
F25F AFCC                    PUSH B          ; SAVE LO WORD OF 0.625 ON STACK.
F261 A4F26CCCAB              LD B, W($MTHI)
F266 AFCC                    PUSH B          ; SAVE HI WORD OF 0.625 ON STACK.
F268 57                      JP $JAMIT       ; GO TO ROUTINE THAT JAMS 0.625^N
                                             ; BY REPEATED MULTIPLICATION INTO HI-INT.LO-INT.
                     ;
                     ; DEFINE SOME CONSTANTS.
F269 40                              . EVEN  ; FORCE EVEN ADDRESS.
F26A 0000            $MTLO:  .WORD 0
F26C 00A0            $MTHI:  .WORD 0A000
F26E CDCC            $DTLO:  .WORD 0CCCD
F270 CCCC            $DTHI:  .WORD 0CCCC
                     ;
                     $DIV10:
                     ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO DIVIDE BY 10.
                     ; ACTUALLY WHAT IS DONE IS
                     ;       10^(-N) = ((2^3)/(0.8))^(-N) = ((0.8)^N)*(2^(-3*N)))
                     ; SO MULTIPLY BY 0.8 NOW AND TAKE CARE OF 2^(-3*N) LATER.
F272 A4F26ECCAB              LD B, W($DTLO)
F277 AFCC                    PUSH B          ; SAVE LO WORD OF .8
F279 A4F270CCAB              LD B, W($DTHI)
F27E AFCC                    PUSH B          ; SAVE HI WORD OF .8
                     ;
                     $JAMIT:
                     ; JAM IN THE MULTIPLICATION PART NEEDED TO HANDLE THE 10'S EXP.
F280 9200                    LD B, 0         ; B IS USED TO TRACK ANY BINARY POWERS
                                             ; THAT COME UP DURING NORMALIZATION.
F282 8200CEFC                IFEQ X, 0       ; IS 10'S EXPONENT 0 ?
F286 57                      JP $JAMDN       ; YES, DONE ALREADY.
                     $JAMLP:
F287 35C0                    JSR BFMUL       ; MULTIPLY USING 32 BIT UNSIGNED.
F289 AECA                    X A, K          ; SWAP HI AND LO WORDS.
F28B E7                      SHL A
F28C 07                      IF C            ; IS THERE A CARRY ?
F28D 4A                      JP $ISNED       ; YES, SO IT IS ALREADY NORMALIZED.
F28E A9CC                    INC B           ; NEED TO SHIFT LEFT TO NORMALIZE, SO
                                             ; INCREASE B BY 1.
F290 AECA                    X A, K
F292 E7                      SHL A
F293 07                      IS C
F294 96CA08                  SET K.0
F297 43                      JP $OVR1
                     $ISNED:
F298 D7                      RRC A
F299 AECA                    X A, K
                     $OVR1:
F29B AACE                    DECSZ X         ; DONE YET ?
F29D 76                      JP $JAMLP       ; NO SO DO IT ONCE MORE.
                     ; GET HERE MEANS MULTIPLICATIONS HAVE BEEN DONE. NOW TAKE
                     ; CARE OF THE EXPONENTS.
                     $JAMDN:
F29E 3FCE                    POP X
F2A0 3FCE                    POP X           ; GET 0.625 OR 0.8 OFF THE STACK.
F2A2 3FCE                    POP X           ; GET THE 10'S EXPONENT.
F2A4 AFC8                    PUSH A          ; SAVE LO WORD OF FLP NUMBER.
F2A6 AFCA                    PUSH K          ; SAVE HI WORD OF FLP NUMBER.
F2A8 A6FFFAC4A8              LD A, W(SP-6)   ; GET THE BINARY EXPONENT THAT WAS SAVED.
F2AD 02                      SET C
F2AE 96CCEB                  SUBC A, B       ; SUBTRACT FROM IT BINARY EXPONENT COLLECTED
                                             ; DURING THE JAMMING.
F2B1 ACC8CC                  LD B, A         ; SAVE IT IN B.
F2B4 A8CE                    LD A, X         ; MOVE THE 10'S EXPONENT TO A.
F2B6 960010                  IF TMP1.0       ; IS THE 10'S EXPONENT NEGATIVE ?
F2B9 49                      JP $NAGAS       ; YES, SO GOT TO SUBTRACT.
                                             ; GET HERE MEANS 10'S EXPONENT IS
                                             ; POSITIVE, SO MUL IT BY 4.
F2BA E7                      SHL A           ; MULTIPLY BY 2.
F2BB E7                      SHL A           ; MULTIPLY BY 2 AGAIN.
F2BC 96CCF8                  ADD A, B        ; GET THE BINARY EXPONENT IN ALSO.
F2BF B8007E                  ADD A, 07E      ; AND THE IEEE BIAS.
F2C2 4C                      JP $EXCPT       ; GO CHECK FOR OVER/UNDERFLOW.
                     ;
                     $NAGAS:
                     ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO GOT TO MULTIPLY
                     ; IT BY -3.
F2C3 E7                      SHL A           ; MULTIPLY BY 2.
F2C4 96CEF8                  ADD A, X        ; ADD TO GIVE MULTIPLY BY 3.
F2C7 01                      COMP A
F2C8 04                      INC A           ; MAKE IT NEGATIVE.
F2C9 96CCF8                  ADD A, B        ; GET IN THE BINARY EXPONENT.
F2CC B8007E                  ADD A, 07E      ; AND THE IEEE BIAS.
                     $EXCPT:
                     ; CHECK FOR OVERFLOW/UNDERFLOW.
F2CF ACC4CE                  LD X, SP        ; FIRST DO SOME JUGGLING
F2D2 02                      SET C           ; TO BE COMPATIBLE WITH EXCEPTION
F2D3 820ACEEB                SUBC X, 0A      ; HANDLING IN OTHER ROUTINES.
F2D7 AFCE                    PUSH X
F2D9 BD7FFF                  IFGT A, 07FFF   ; IS BIASED EXPONENT NEGATIVE ?
F2DC B4FD3F                  JMPL UNDFL
F2DF 9C00                    IFEQ A, 0       ; IS IT 0 ?
F2E1 B4FD3A                  JMPL UNDFL      ; YES IT IS STILL UNDERFLOW.
F2E4 9DFE                    IFGT A, 0FE     ; IS IT GT THAN 254 ?
F2E6 B4FD46                  JMPL OVRFL
                     ; GET HERE MEANS VALID SP FLP NUMBER.
F2E9 3FCE                    POP X           ; X POINTS TO MANTISSA SIGN.
F2EB E7                      SHL A
F2EC E7                      SHL A
F2ED E7                      SHL A
F2EE E7                      SHL A
F2EF E7                      SHL A
F2FD E7                      SHL A
```



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    22
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F2F1 E7                         SHL A
F2F2 E7                         SHL A           ; MOVE EXPONENT TO HIGH BYTE.
F2F3 8FFA                       OR A, W(X)      ; GET THE MANTISSA SIGN IN.
F2F5 AB00                       ST A, TMP1      ; SAVE IT IN TMP1.
F2F7 3FCA                       POP K           ; FI-HI TO K.
F2F9 3FC8                       POP A           ; F1-LO TO A.
F2FB F1                         X A, W(X+)      ; SAVE F1-LO.
F2FC A8CA                       LD A, K
F2FE F1                         X A, W(X+)      ; SAVE F1-HI.
F2FF A800                       LD A, TMP1
F301 F3                         X A, W(X-)      ; SAVE F1-EXP.F1-SIGN, X POINTS TO F1-HI.
F302 3664                       JSRL SROUND     ; ROUND THE RESULT.
F304 F2                         LD A, W(X-)     ; X POINTS TO F1-HI.
F305 F2                         LD A, W(X-)     ; X POINTS TO F1-LO.
F306 AFCE                       PUSH X
F308 368C                       JSR FPAK        ; PACK IT INTO IEEE FORMAT.
F30A 3FC4                       POP SP
F30C 3FCC                       POP B
F30E 3FCE                       POP X
F310 3C                         RET
                        ;
                                        .END
```



```
                                      .FORM 'FTOA.MAC'
                                      .INCLD FTOA.MAC
                                      .TITLE FTOA
                                      .LOCAL
                        ;
                        ; THIS SUBROUTINE CONVERTS A SINGLE PRECISION, BINARY FLOATING
                        ; POINT NUMBER IN THE IEEE FORMAT TO A DECIMAL FLOATING POINT
                        ; STRING. THE DECIMAL FLOATING POINT STRING IS OBTAINED TO A
                        ; PRECISION OF 9 DECIMAL DIGITS.
                        ;
                        ; THE ALGORITHM USED IS BASED ON:
                        ; J.T. COONEN, 'AN IMPLEMENTATION GUIDE TO A PROPOSED STANDARD
                        ; FOR FLOATING POINT ARITHMETIC,' IEEE COMPUTER, JAN. 1980, PP 68-79.
                        ;
                        ; ON INPUT, THE BINARY SP FLP NUMBER IS IN REGS. K AND A.
                        ; B CONTAINS THE ADDRESS OF THE LOCATION WHERE THE DECIMAL FLOATING
                        ; POINT STRING IS TO START. NOTE THAT AT LEAST 17 BYTES ARE NEEDED
                        ; FOR THE STORAGE OF THE STRING. THE LAST BYTE IS ALWAYS NULL.
                        ;
                        ; ALL REGISTERS ARE PRESERVED BY THIS SUBROUTINE.
                        ;
                        FTOA:
F311 AFCE                       PUSH X          ; SAVE X ON THE STACK.
F313 AFCC                       PUSH B          ; SAVE B ON THE STACK.
                        ; CHECK AND SEE IF F1 IS A NAN.
F315 3605                       JSR FNACHK
F317 07                         IF C
F318 B401B4                     JMPL $NAN       ; YET IT IS, SO GET OUT.
                        ; CHECK AND SEE IF F1 IS ZERO.
F31B 36CA                       JSR FZCHK
F31D 07                         IF C
F31E B401C8                     JMPL $ZERO      ;YES IT IS, SO GET OUT.
                        ; GET HERE MEANS F1 IS A NON-ZERO, NON-NAN FLP NUMBER.
F321 ACC4CE                     LD X, SP
F324 8206C4F8                   ADD SP, 06      ; ADJUST SP.
F328 36C7                       JSR FUNPAK      ; UNPACK THE NUMBER.
                                                ; X POINTS ONE WORD PAST F1-EXP.F1-SIGN
                                                ; ON RETURN.
                        ;
                        ; COMPUTE THE EXPONENT OF 10 FOR DECIMAL FLP NO.
                        ; THIS IS DONE AS FOLLOWS:
                        ;       SUPPOSE F1 = FM * (2^M)
                        ;       LET U = M*LOG(2)        NOTE: LOG IS TO BASE 10.
                        ;       THEN V = INT(U+1-9)
                        ;       IS USED AS THE 10'S EXPONENT.
                        ;                               NOTE: INT REFERS TO INTEGER PART.
                        ;
F32A D2                         LD A, M(X-)     ; X POINTS TO F1-EXP.
F32B D2                         LD A, M(X-)     ; LOAD F1-EXP. X POINTS TO F1-SIGN.
F32C B7000000                   LD TMP1, 0      ; FIRST GUESS POSITIVE SIGN FOR EXP.
F330 B8FF82                     ADD A, 0FF82    ; REMOVE IEEE BIAS FROM F1-EXP.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    24
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

F333 AFC8                   PUSH A          ; SAVE IT ON THE STACK.
F335 AFCE                   PUSH X          ; SAVE F1-SIGN ADDRESS ALSO.
F337 07                     IF C            ; WAS THERE A CARRY ON THE LAST ADD ?
F338 46                     JP $MLOG2       ; YES, SO 2'S EXP IS POSITIVE.
F339 B700FF00               LD TMP1, 0FF    ; 2'S EXPONENT IS NEGATIVE.
F33D 01                     COMP A
F33E 04                     INC A           ; MAKE IT POSITIVE.
                    $MLOG2:
                    ; MULTIPLY M BY LOG(2).
F33F BE4D10                 MULT A, 04D10   ; LOG(2) IS 0.0100110100010000 TO 16 BITS.
                                            ; X CONTAINS INTEGER PART, AND A FRACT. PART.
F342 AECE                   X A, X          ; SWAP THE TWO.
F344 960010                 IF TMP1.0       ; WAS THE 2'S EXPONENT NEGATIVE ?
F347 41                     JP $CSIGN       ; YES, SO MAKE U NEGATIVE.
F348 4B                     JP $REMV9       ; NO, SO GO DO V = U + 1 - 9.
                    $CSIGN:
F349 01                     COMP A          ; COMP INTEGER PART.
F34A AECE                   X A, X
F34C 01                     COMP A          ; FRACTION PART.
F34D B80001                 ADD A, 01
F350 AECE                   X A, X
F352 07                     IF C
F353 04                     INC A
                    $REMV9:
F354 04                     INC A           ; INCREASE FRACTION PART.
F355 B8FFF7                 ADD A, 0FFF7    ; SUBTRACT 9.
F358 BD7FFF                 IFGT A, 07FFF   ; IS IT NEGATIVE ?
F35B 45                     JP $CHNGS       ; YES, SO CHANGE ITS SIGN.
F35C B7000000               LD TMP1, 0      ; REMEMBER POSITIVE SIGN.
F36D 4F                     JP $DIV10
                    $CHNGS:
F361 B700FF00               LD TMP1, 0FF    ; REMEMBER NEGATIVE SIGN.
F365 01                     COMP A          ; MAKE V POSITIVE.
F366 AECE                   X A, X
F368 01                     COMP A
F369 B80001                 ADD A, 01
F36C AECE                   X A, X
F36E 07                     IF C
F36F 04                     INC A
                    $DIV10:
                    ;
                    ; V = INT (U+1-9) HAS BEEN COMPUTED AND IS IN A.
                    ; NOW COMPUTE W = F1/(10^V). W SHOULD BE AN INTEGER, AND IT IS
                    ; COMPUTED TO A 32 BIT PRECISION.
                    ; THIS COMPUTATION IS DONE AS FOLLOWS:
                    ;       IF V > 0, THEN F1/(10^V) = F1*(0.8^V)*(2^(-3V)).
                    ;       IF V < 0, THEN F1/(10^V) = F1*(0.625^U)*(2^(4V)).
                    ; SO FIRST MULTIPLY THE MANTISSA OF F1 V TIMES BY 0.8 (OR 0.625)
                    ; AND THEN ADJUST THE EXPONENT OF F1. NOTE THAT THE PARTIAL PRODUCTS
                    ; IN MULTIPLYING BY 0.8 (OR 0.625) ARE KEPT NORMALIZED. THIS IS
                    ; ESSENTIAL TO PRESERVE 32 BIT ACCURACY IN THE FINAL RESULT.
```

```
                     ; SINCE THE MANTISSA OF F1 IS NORMALIZED, AND 0.8 (OR 0.625 IS ALSO
                     ; NORMALIZED, EACH PRODUCT NEEDS AT MOST 1 LEFT SHIFT FOR
                     ; RENORMALIZATION. THE SHIFTS ACCUMULATED DURING RENORMALIZATION ARE
                     ; TRACKED AND ACCOUNTED FOR IN THE CALCULATION.
F370 3FCE                    POP X           ; X NOW POINTS TO F1-SIGN.
F372 AFC8                    PUSH A          ; SAVE U ON THE STACK.
F374 ACC8CC                  LD B, A         ; MOVE V TO B ALSO.
F377 F2                      LD A, W(X-)     ; X POINTS TO F1-HI.
F378 F2                      LD A, W(X-)     ; LOAD F1-HI. X POINTS TO F1-LO.
F379 ACC8CA                  LD K, A
F37C F4                      LD A, W(X)      ; LOAD F1-LO.
F370 960010                  IF TMP1.0       ; IS V NEGATIVE?
F380 57                      JP $MUL10       ; YES, SO MULTIPLY V TIMES BY .625.
                                             ; GET HERE MEANS MULTIPLY V TIMES BY .8.
F381 A4F390CEAB              LD X, W($DTLO)
F386 AFCE                    PUSH X          ; LO WORD OF 0.8 TO STACK.
F388 A4F392CEAB              LD X, W($DTHI)
F38D AFCE                    PUSH X          ; HI WORD OF 0.8 TO STACK.
F38F 56                      JP $JAMIT       ; GO DO MULTIPLICATION.
                     ;
                             .EVEN           ; FORCE EVEN ADDRESS.
F390 CDCC            $DTLO:  .WORD 0CCCD
F392 CCCC            $DTHI:  .WORD 0CCCC
F394 0000            $MILO:  .WORD 0
F396 00A0            $MTHI:  .WORD 0A000
                     ;
                     $MUL10:
F398 A4F394CEAB              LD X, W($MTLO)
F39D AFCE                    PUSH X          ; LO WORD OF 0.625 TO STACK.
F39F A4F396CEAB              LD X, W($MTHI)
F3A4 AFCE                    PUSH X          ; HI WORD OF 0.625 TO STACK.
                     ;
                     $JAMIT:
F3A6 9300                    LD X, 0         ; INIT X TO TRACK ANY POWERS OF
                                             ; 2 GENERATED DURING NORMALIZATION
                                             ; OF PARTIAL PRODUCTS.
F3A8 8200CCFC                IFEQ B, 0       ; IS B ALREADY 0 ?
F3AC 57                      JP $JAMON       ; YES, SO SKIP MULTIPLY LOOP.
                     $JAMLP:
F3AD 36E6                    JSR BFMUL       ; MULTIPLY.
F3AF AECA                    X A, K          ; SWAP HI AND LO WORDS OF PART. PROD.
F3B1 E7                      SHL A
F3B2 07                      IF C            ; IS THERE A CARRY ?
F3B3 4A                      JP $ISNED       ; YES, SO SKIP OVER RENORMALIZATION.
                                             ; GET HERE MEANS NEED TO RENORM.
F3B4 A9CE                    INC X           ; UPDATE RENORM COUNT.
F3B6 AECA                    X A, K          ; SWAP HI AND LO PART. PROD.
F3B8 E7                      SHL A
F3B9 07                      IF C
F3BA 96CA08                  SET K.0         ; SET BIT SHIFTED OUT FROM LO WORD.
F3BD 43                      JP $OVR1
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    26
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

                     $ISNED:
F3BE D7                      RRC A           ; PUT BACK SHIFTED BIT.
F3BF AECA                    X A, K
                     $OVR1:
F3C1 AACC                    DECSZ B         ; IS B 0 YET ?
F3C3 76                      JP $JAMLP       ; NO, SO DO IT AGAIN.
                     $JAMON:
                     ; GET HERE MEANS MULTIPLICATION HAS BEEN DONE, SO TAKE CARE
                     ; OF EXPONENT.
F3C4 3FCC                    POP B
F3C6 3FCC                    POP B           ; GET RID OF 0.8 (OR 0.625) FROM STACK.
F3C8 AFC8                    PUSH A          ; SAVE LO WORD OF PROD.
F3CA AFCA                    PUSH K          ; SAVE HI WORD OF PRODUCT.
F3CC A6FFF8C4A8              LD A, W(SP-08)  ; GET F1'S BINARY EXPONENT.
F3D1 02                      SET C
F3D2 96CEEB                  SUBC A, X       ; SUBTRACT FROM IT ANYTHING COLLECTED
                                             ; DURING RENORM.
F3D5 ACC8CE                  LD X, A         ; AND SAVE IT IN X.
F3D8 A6FFFAC4A8              LD A, W(SP-06)  ; GET V FROM THE STACK.
F3DD 960010                  IF TMP1.0       ; IS V NEGATIVE ?
F3E0 49                      JP $ML4         ; YES, SO MULTIPLY V BY 4.
                                             ; GET HERE MEANS MULTIPLY V BY -3.
F3E1 E7                      SHL A           ; NOW A CONTAINS 2*V.
F3E2 A6FFFAC4F8              ADD A, W(SP-06) ; NOW A CONTAINS 3*V.
F3E7 01                      COMP A
F3E8 04                      INC A           ; NOW A CONTAINS -3*V.
F3E9 42                      JP $ADEM        ; GO FIGURE FINAL EXPONENT.
                     $ML4:
F3EA E7                      SHL A
F3EB E7                      SHL A           ; NOW A CONTAINS 4*V.
                     $ADEM:                  ;
F3EC 96CEF8                  ADD A, X        ; A SHOULD NOW BE A POSITIVE INTEGER
                                             ; IN THE RANGE 0 TO 32.
                     ; NOW CHECK AND SEE IF A HAS ENOUGH PRECISION.
F3EF 9020                    IFGT A, 020     ; NEED MORE THAN 32 BITS ?
F3F1 5A                      JP $INCRV       ; YES, SO GO INCREASE V.
F3F2 9D1B                    IFGT A, 01B     ; NEED AT LEAST 28 BITS ?
F3F4 9435                    JMP $GOON       ; YES, SO ALL IS OK. GO ON.
                                             ; GET HERE MEANS NEED MORE
                                             ; PRECISION, SO DECREASE V.
F3F6 3FC8                    POP A           ; GET HI-PROD OFF STACK.
F3F8 3FC8                    POP A           ; GET LO WORD OFF STACK.
F3FA 3FC8                    POP A           ; GET MAGN. OF V.
F3FC 96D010                  IF TMP1.0       ; IS V NEG. ?
F3FF 56                      JP $VUP         ; YES, SO GO INCR. MAGN. OF V.
                                             ; GET HERE MEANS V IS POSITIVE,
                                             ; AND NEED TO DECREMENT IT.
F400 B8FFFF                  ADD A, 0FFFF    ; SUBTRACT 1 FROM A.
F403 07                      IF C            ; GOT A CARRY ?
F404 5A                      JP $GOBAK       ; THEN OK.
F405 01                      COMP A
```

```
F406 04                      INC A
F407 B700FF00                LD TMP1, OFF    ; U CHANGES SIGN.
F40B 53                      JP $GOBAK
                     $INCRV:
F40C 3FC8                    POP A           ; GET HI PROD. OFF STACK.
F40E 3FC8                    POP A           ; GET LO PROD. OFF STACK.
F410 3FC8                    POP A           ; GET MAGN. OF V.
F412 960010                  IF TMP1.0       ; IS V NEGATIVE ?
F415 42                      JP $VDOWN       ; YES.
                     $VUP:
F416 04                      INC A
F417 47                      JP $GOBAK
                     $VDOWN:
F418 AAC8                    DECSZ A
F41A 44                      JP $GOBAK
F41B B7000000                LD TMP1, 0      ; V CHANGES SIGN.
                     $GOBAK:
F41F ACC4CE                  LD X, SP
F422 02                      SET C
F423 8204CEEB                SUBC X, 04
F427 AFCE                    PUSH X
F429 95B9                    JMP $DIV10
                     $GOON:
F42B 01                      COMP A
F42C 04                      INC A           ; NEGATE A.
F42D B80020                  ADD A, 020      ; SUBTRACT IT FROM 32.
F430 ACC8CE                  LD X, A         ; AND MOVE IT TO X.
F433 3FCA                    POP X           ; GET HI WORD OF PRODUCT.
F435 3FC8                    POP A           ; GET LO WORD OF PROD.
F437 8200CEFC                IFEQ X, 0       ; IS X 0 ?
F43B 49                      JP $INDUN       ; YES, SO ALREADY A 32 BIT INTEGER.
                     $INTFY:
                     ; NOW ADJUST THE PRODUCT TO FORM A 32 BIT INTEGER.
F43C AECA                    X A, K          ; SWAP HI AND LO WORDS.
F43E C7                      SHR A
F43F AECA                    X A, K
F441 D7                      RRC A           ; SHIFT IT RIGHT ONCE.
F442 AACE                    DECSZ X         ; X 0 YET ?
F444 68                      JP $INTFY       ; NO SO GO DO SOME MORE.
                     $INDUN:
                     ; GET HERE MEANS K.A CONTAIN THE 32 BIT INTEGER THAT IS THE
                     ; MANTISSA OF THE DECIMAL FLP NUMBER.
F445 AFC8                    PUSH A          ; SAVE LO-INT.
F447 AFCA                    PUSH K          ; SAVE HI INT.
F449 A6FFF0C4A8              LD A, W(SP-010) ; GET STARTING ADDRESS OF DECIMAL STRING.
F44E B80010                  ADD A, 010      ; ADD 16 TO IT.
F451 ACC8CC                  LD B, A         ; AND MOVE IT B.
F454 00                      CLR A
F455 C3                      XS A, M(B-)     ; OUTPUT TERMINATING NULL BYTE.
F456 40                      NOP
F457 A6FFFAC4A8              LD A, W(SP-06)  ; GET V.
```

```
254 F45C 9F0A                    DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
255                                              ; REM. IN X.
256 F45E AECE                    X A, X          ; REM TO A.
257 F460 B80030                  ADD A, 030      ; MAKE IT INTO ASCII BYTE.
258 F463 C3                      XS A, M(B-)     ; OUTPUT IT.
259 F464 40                      NOP
260 F465 A8CE                    LD A, X
261 F467 B80030                  ADD A, 030
262 F46A C3                      XS A, M(B-)
263 F46B 40                      NOP             ; FINISHED OUTPUTING EXPONENT.
264 F46C 902B                    LD A, 028       ; SAY EXP SIGN IS '+'.
265 F46E 960010                  IF TMP1.0
266 F471 902D                    LD A, 02D       ; NOPE, IT IS '-'.
267 F473 C3                      XS A, M(B-)     ; OUTPUT IT.
268 F474 40                      NOP
269 F475 9045                    LD A, 045
270 F477 C3                      XS A, M(B-)     ; OUTPUT 'E'.
271 F478 40                      NOP
272 F479 902E                    LD A, 02E
273 F47B C3                      XS A, M(B-)     ; OUTPUT '.'.
274 F47C 40                      NOP
275                      ; NOW NEED TO OUTPUT 10 DECIMAL DIGITS.
276 F47D B7000A00                LD TMP1, 0A     ; LOAD 10 INTO TMP1 AS LOOP COUNTER.
277                      $DOLUP:
278 F481 3FC8                    POP A           ; A CONTAINS HI INT.
279 F483 9F0A                    DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
280                                              ; REM. IN X.
281 F485 ACC8CA                  LD K, A
282 F488 3FC8                    POP A           ; A CONTAINS LO INT.
283 F48A AFCC                    PUSH B          ; SAVE DEC. STR. ADDR.
284 F48C ACCACC                  LD B, K         ; B CONTAINS HI-QUOT.
285 F48F 82                      .BYTE 082,0A,0C8,0EF
    F490 0A
    F491 C8
    F492 EF
286                      ; THE ABOVE 4 BYTES REPRESENT THE INSTRUCTION DIVD A, 0A.
287                      ; BECAUSE THE ASSEMBLER DOES NOT KNOW ABOUT IT YET, WE HAVE TO
288                      ; KLUDGE IT THIS WAY.
289                      ; AFTER THE DIVD, A CONTAINS THE LO-QUOT. AND X THE REM.
290 F493 ACCCCA                  LD K, B         ; MOVE HI-QUOT TO K.
291 F496 3FCC                    POP B           ; B CONTAINS DEC. STR. ADDR.
292 F498 AFC8                    PUSH A          ; SAVE LO INT.
293 F49A AFCA                    PUSH K          ; SAVE HI INT.
294 F49C A8CE                    LD A, X         ; MOVE REM TO A.
295 F49E B80030                  ADD A, 030      ; ASCII-FY IT.
296 F4A1 C3                      XS A, M(B-)     ; AND OUTPUT IT.
297 F4A2 40                      NOP
298 F4A3 AA00                    DECSZ TMP1      ; IS TMP1 0 YET ?
299 F4A5 9524                    JMP $DOLUP      ; NO, GO GET SOME MORE.
300                      ; GET HERE MEANS DONE WITH OUTPUTING MANTISSA.
301 F4A7 3FC8                    POP A
```

```
F4A9 3FC8                    POP A
F4AB 3FC8                    POP A
F4AD 3FC8                    POP A           ; GET SOME GARBAGE OFF THE STACK.
F4AF 3FCA                    POP K           ; GET F1-EXP.F1-SIGN TO K.
F4B1 9020                    LD A, 02D       ; LOAD SP INTO A.
F4B3 96CA10                  IF K.0
F4B6 902D                    LD A, 02D       ; IF MAINT. IS NEG. LOAD '-'.
F4B8 C6                      ST A, M(B)      ; OUTPUT SIGN.
F4B9 AFCA                    PUSH K          ; F1-EXP.F1-SIGN BACK ON STACK.
F4BB ACC4CE                  LD X, SP
F4BE 02                      SET C
F4BF 8206CEEB                SUBC X, 06      ; X POINTS TO F1-LO.
F4C3 AFCE                    PUSH X
F4C5 B5FBB4     +            JSR FPAK        ; PACK IT, SO RESTORING K AND A.
F4C8 3FC4                    POP SP
F4CA 3FCC                    POP B           ; RESTORE B.
F4CC 3FCE                    POP X           ; RESTORE X.
F4CE 3C                      RET
                     ;
                     $NAN:
                     ; GET HERE MEANS F1 IS A NAN.
F4CF AFC8                    PUSH A
F4D1 ACCCCE                  LD X, B
F4D4 8210CEF8                ADD X, 010
F4D8 00                      CLR A
F4D9 03                      X A, M(X-)
F4DA 9210                    LD B, 010
                     $NANLP:
F4DC 90FF                    LD A, 0FF
F4DE D3                      X A, M(X-)
F4DF AACC                    DECSZ B
F4E1 65                      JP $NANLP
F4E2 3FC8                    POP A
F4E4 3FCC                    POP B
F4E6 3FCE                    POP X
F4E8 3C                      RET
                     ;
                     $ZERO:
                     ; GET HERE MEANS F1 IS ZERO.
F4E9 AFC8                    PUSH A
F4EB ACCCCE                  LD X, B         ; X CONTAINS DECIMAL STRING ADDR.
F4EE 8210CEF8                ADD X, 010
F4F2 00                      CLR A
F4F3 D3                      X A, M(X-)      ; OUTPUT TERMINATING NULL BYTE.
F4F4 9030                    LD A, 030       ; LOAD 0 INTO A.
F4F6 D3                      X A, M(X-)
F4F7 9030                    LD A, 030
F4F9 D3                      X A, M(X-)      ; OUTPUT 00 FOR EXPONENT.
F4FA 902B                    LD A, 02B       ; LOAD '+' SIGN.
F4FC D3                      X A, M(X-)
F4FD 9045                    LD A, 045       ; LOAD 'E'
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    30
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

353 F4FF D3                        X A, M(X-)
354 F500 902E                      LD A, 02E       ; LOAD '.'.
355 F502 D3                        X A, M(X-)
356 F503 920A                      LD B, 0A
357                        $ZERLP:
358 F505 9030                      LD A, 030
359 F507 D3                        X A, M(X-)
360 F508 AACC                      DECSZ B
361 F50A 65                        JP $ZERLP
362 F508 9020                      LD A, 020       ; LOAD SP.
363 F50D D5                        X A, M(X)
364 F50E 3FC8                      POP A
365 F510 3FCC                      POP B
366 F512 3FCE                      POP X
367 F514 3C                        RET
368                        ;
369                                .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    31
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FADD.MAC

                                    .FORM, 'FADD.MAX'
                                    .INCLD FADD.MAC
                                    .TITLE FADD
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO ADD/SUBTRACT TWO SP FLOATING POINT NUMBERS.
                    ;          C = F1 + F2 OR C = F1 - F2
                    ;
                    ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                    ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                    ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                    ;
                    ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                    ; STACK POINTER ON ENTRY, THEN
                    ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                    ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                    ;
                    ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                    ;
                    FSUB:
F515 AB00                   ST A, TMP1
F517 A6FFFAC4A8             LD A, W(SP-06)  ; LOAD F2-R0.
F51C AFC8                   PUSH A          ; AND SAVE ON STACK.
F51E A6FFFAC4A8             LD A, W(SP-06)  ; LOAD F2-R1.
F523 BB8000                 XOR A, 08000    ; CHANGE THE SIGN.
F526 AFC8                   PUSH A          ; AND SAVE ON THE STACK.
F528 A800                   LD A, TMP1      ; RESTORE A.
F52A 3009                   JSR FADD        ; CALL THE ADD ROUTINE.
F52C AB00                   ST A, TMP1      ; SAVE A.
F52E 3FC8                   POP A           ; GET RID OF JUNK
F530 3FC8                   POP A           ; FROM THE STACK.
F532 A800                   LD A, TMP1      ; RESTORE A.
F534 3C                     RET
                    ;
                    FADD:
                    ; SAVE ADDRESS OF F2-R0 IN TMP1.
F535 AFCE                   PUSH X          ; SAVE X ON ENTRY.
F537 AFCC                   PUSH B          ; AND B ON ENTRY.
F539 ACC4CE                 LD X, SP
F53C 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
F541 ACCE00                 LD TMP1, X      ; AND SAVE IN TMP1.
                    ; CHECK AND SEE IF F1 IS A NAN.
F544 B5FAF9       +         JSR FNACHK
F547 07                     IF C
F548 B4FAC4                 JMPL FNAN       ; F1 IS A NAN.
                    ; CHECK AND SEE IF F2 IS A NAN.
F54B ACCACC                 LD B, K
F54E ACC8CE                 LD X, A
F551 A20200A8               LD A, W(TMP1+2)
F555 ACC8CA                 LD K, A
F558 AECE                   X A, X
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    32
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

F55A B5FAE3      +          JSR FNACHK
F55D 07                     IF C
F55E B4FAAE                 JMPL FNAN       ; F2 IS NAN.
                    ; CHECK AND SEE IF F2 IS ZERO.
F561 B5FAED      +          JSR FZCHK
F564 06                     IFN C
F565 48                     JP $F1CHK       ; F2 IS NOT ZERO. CHECK F1.
F566 ACCCCA                 LD K, B         ; F2 IS ZERO, SO ANSWER IS F1.
F569 3FCC                   POP B
F56B 3FCE                   POP X
F56D 3C                     RET
                    ; CHECK AND SEE IF F1 IS ZERO.
                    $F1CHK:
F56E ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
F571 B5FADD      +          JSR FZCHK
F574 06                     IFN C
F575 4F                     JP $NTZERO      ; JUMP SINCE F1 IS ALSO NOT ZERO.
F567 A20200AB               LD A, W(TMP1+2) ; GET HERE MEANS F1 IS ZERO,
F57A ACC8CA                 LD K, A         ; SO ANSWER IS F2.
F57D AD00A8                 LD A, W(TMP1)
F580 3FCC                   POP B
F582 3FCE                   POP X
F584 3C                     RET
                    ; GET HERE MEANS NORMAL ADDITION.
                    ; UNPACK F1 AND F2.
                    $NTZERO:
F585 ACC4CE                 LD X, SP        ; X POINTS TO F1-LO.
F588 8210C4F8               ADD SP, 010     ; MOVE SP PAST LOCAL STORAGE.
F58C AFCE                   PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
F58E B5FAD0      +          JSR FUNPAK      ; UNPACK F1.
F591 AC00CC                 LD B, TMP1      ; B NOW POINTS TO F2-R0.
F594 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
F597 E0                     LDS A, W(B+)    ; LOAD F2-R0 INTO A.
F598 40                     NOP
F599 AECA                   X A, K
F59B E4                     LD A, W(B)
F59C AECA                   X A, K          ; LOAD F2-R1 INTO K.
F59E B5FAC0      +          JSR FUNPAK      ; UNPACK F2.
                    ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
F5A1 F2                     LD A, W(X-)
F5A2 AC00CC                 LD B, TMP1
F5A5 E2                     LDS A, W(B-)
F5A6 40                     NOP
                    ; COMPARE F1-EXP AND F2-EXP.
F5A7 F2                     LD A, W(X-)     ; LOAD F2-EXP.F2-SIGN INTO A.
F5A8 B9FF00                 AND A, 0FF00    ; MASK OUT SIGN.
F5AB A6FFFCC4AB             ST A, W(SP-4)   ; SAVE IN C-SIGN.C-EXP.
F5B0 E2                     LDS A, W(B-)
F5B1 40                     NOP             ; LOAD F1-EXP.F1-SIGN INTO A.
F5B2 B9FF00                 AND A, 0FF00    ; CHANGE TO F1-EXP.00000000.
F5B5 02                     SET C
```

```
F5B6 A6FFFCC4EB              SUBC A, W(SP-4) ; SUBTRACT F2-EXP.00000000.
F5BB 06                      IFN C
F5BC 942D                    JMP $F2GTR      ; F2-EXP IS BIGGER THAN F1-EXP.
                     ; GET HERE MEANS F1-EXP IS BIGGER THAN F2-EXP.
F5BE 80C9CAAB                LD K, H(A)      ; SAVE DIFF. IN K TO BE USED AS LOOP COUNTER.
F5C2 A6FFFCC4FB              ADD A, W(SP-4)
F5C7 A6FFFCC4AB              ST A, W(SP-4)   ; RESTORE F1-EXP AND STORE IN C-SIGN.
F5CC 8217CAFD                IFGT K, 017
F5D0 51                      JP $ZROF2       ; K GT 23-DEC MEANS F2 GETS ZEROED IN SHIFTS.
                     ; LOOP TO SHIFT F2 INTO ALIGNMENT.
F5D1 8200CAFC                IFEQ K, 0
F5D5 943B                    JMP $ADDMN      ; K = 0 MEANS DONE SHIFTING.
                     $LOOP2:
F5D7 F4                      LD A, W(X)
F5D8 C7                      SHR A
F5D9 F3                      X A, W(X-)
F5DA F4                      LD A, W(X)
F5DB D7                      RRC A
F5DC F1                      X A, W(X+)
F5DD AACA                    DECSZ K
F5DF 68                      JP $LOOP2
F5E0 9430                    JMP $ADDMN
                     $ZROF2:
                     ; SET F2 MANTISSA TO 0.
F5E2 F0                      LD A, W(X+)     ; X POINTS TO F2-EXP.F2-SIGN.
F5E3 00                      CLR A
F5E4 F3                      X A, W(X-)      ; AND STORE IT BACK.
F5E5 00                      CLR A
F5E6 F3                      X A, W(X-)
F5E7 00                      CLR A
F5E8 F1                      X A, W(X+)
F5E9 9427                    JMP $ADDMN
                     ; F2 EXPONENT IS GREATER THAN F1 EXPONENT.
                     $F26TR:
F5EB 01                      COMP A
F5EC 04                      INC A           ; CHANGE DIFF IN EXP TO POSITIVE.
F5ED 80C9CAAB                LD K, H(A)      ; LOAD K WITH LOOP COUNTER.
F5F1 8217CAFD                IFGT K, 017
F5F5 51                      JP $ZROF1       ; F1 MANT. REDUCED TO 0 IN SHIFTS.
                     ; LOOP TO SHIFT F1 MANT INTO ALIGNMENT.
F5F6 8200CAFC                IFEQ K, 0
F5FA 57                      JP $ADDMN       ; K=0 MEANS DONE SHIFTING.
                     $LOOP1:
F5FB E4                      LD A, W(B)
F5FC C7                      SHR A
F5FD E3                      XS A, W(B-)
F5FE 40                      NOP
F5FF E4                      LD A, W(B)
F600 D7                      RRC A
F601 E1                      XS A, W(B+)
F602 40                      NOP
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    34
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

F603 AACA                    DECSZ K         ;
F605 6A                      JP $LOOP1
F606 4B                      JP $ADDMN
                     $ZROF1:                 ; SET F1 MANT TO 0.
F607 E0                      LOS A, W(B+)    ; B POINTS TO F1-EXP.F1-SIGN.
F608 40                      NOP
F609 00                      CLR A
F60A E3                      XS A, W(B-)     ; STORE IT BACK.
F60B 40                      NOP
F60C 00                      CLR A
F60D E3                      XS A, W(B-)
F60E 40                      NOP
F60F 00                      CLR A
F610 E1                      XS A, W(B+)
F611 40                      NOP
                     ; DETERMINE IF MANTISSAS ARE TO BE ADDED OR SUBTRACTED.
                     $ADDMN:                 ; B POINTS TO F1-HI, X TO F2-HI.
F612 E0                      LDS A, W(B+)
F613 40                      NOP
F614 F0                      LD A, W(X+)
F615 D4                      LD A, M(X)      ; LOAD F2-SIGN.
F616 D8                      XOR A, M(B)     ; XOR WITH F1-SIGN.
F617 9C00                    IFEQ A, 0
F619 9451                    JMP $TRADD      ; SAME SIGN SO GO TO ADD MANTISSA.
                     ; GET HERE MEANS TRUE SUBTRACT OF MANTISSA.
F61B F2                      LD A, W(X-)
F61C F2                      LD A, W(X-)     ; X POINTS TO F2-LO.
F61D E2                      LDS A, W(B-)
F61E 40                      NOP
F61F E2                      LDS A, W(B-)
F620 40                      NOP             ; B NOW POINTS TO F1-LO.
F621 E0                      LDS A, W(B+)
F622 40                      NOP             ; A NOW CONTAINS F1-LO.
F623 02                      SET C
F624 8FEB                    SUBC A, W(X)    ; SUBTRACT F2-LO.
F626 F1                      X A, W(X+)
F627 E0                      LDS A, W(B+)    ; A CONTAINS F1-HI.
F628 40                      NOP
F629 8FEB                    SUBC A, W(X)    ; SUBTRACT F2-HI.
F62B F1                      X A, W(X+)
F62C 07                      IF C
F62D 55                      JP $F1SIN       ; F1 GE F2, SO SIGN IS F1-SIGN.
                     ; GET HERE MEANS F1 LT F2, SO SIGN IS F2-SIGN.
F62E A6FFFCC4A8              LD A, W(SP-4)
F633 8FDA                    OR A, M(X)
F635 F3                      X A, W(X-)      ; C-EXP.C-SIGN HAS BEEN DETERMINED.
F636 F4                      LD A, W(X)
F637 D1                      COMP A
F638 F3                      X A, W(X-)
F639 F4                      LD A, W(X)
F63A 01                      COMP A
```

```
F63B B80001                  ADD A, 01
F63E F1                      X A, W(X+)
F63F 07                      IF C
F640 8FA9                    INC W(X)
F642 47                      JP $ANORM
                     ; GET HERE MEANS F1 GE F2.
                     $F1SIN:
F643 A6FFFCC4A8              LD A, W(SP-4)
F648 DA                      OR A, M(B)
F649 F3                      X A, W(X-)
                     ; NORMALIZE THE MANTISSA.
                     $ANORM:
F64A ACCECC                  LD B, X         ; B POINTS TO C-HI.
F64D E0                      LDS A, W(B+)
F64E 40                      NOP
F64F C0                      LDS A, M(B+)
F650 40                      NOP             ; B NOW POINTS TO C-EXP BYTE.
F651 9118                    LD K, 018       ; SET UP LOOP LIMIT OF 24-DEC IN K.
                     $NLOOP:
F653 F4                      LD A, W(X)
F654 E7                      SHL A
F655 07                      IF C            ; CARRY MEANS NORMALIZED.
F656 9448                    JMP $ROUND      ; SO JUMP TO ROUNDING CODE.
F658 F3                      X A, W(X-)
F659 F4                      LD A, W(X)
F65A E7                      SHL A
F65B F1                      X A, W(X+)
F65C 07                      IF C
F65D 8F08                    SET W(X).0
F65F ADCC8A                  DECSZ M(B)      ; ADJUST EXPONENT.
F662 43                      JP $OV1
F663 B4F9B8                  JMPL UNDFL      ; C-EXP ZERO MEANS UNDERFLOW.
                     $OV1:
F666 AACA                    DECSZ K         ; DECREMENT LOOP COUNTER.
F668 75                      JP $NLOOP       ; GO BACK TO LOOP.
F669 B4F9B2                  JMPL UNDFL      ; UNDERFLOW
                     ;GET HERE MEANS TRUE ADDITION OF MANTISSA.
                     $TRADD:
F66C E2                      LDS A, W(B-)
F66D 40                      NOP
F66E E2                      LDS A, W(B-)
F66F 40                      NOP             ; B NOW POINTS TO F1-HI.
F670 F2                      LD A, W(X-)
F671 F2                      LD A, W(X-)
F672 F4                      LD A, W(X)      ; LOAD F2-LO INTO A.
F673 F8                      ADD A, W(B)     ; ADD F1-LO.
F674 F1                      X A, W(X+)      ; STORE IN F2-LO.
F675 E0                      LDS A, W(B+)
F676 40                      NOP             ; B NOW POINTS TO F1-HI.
F677 F4                      LD A, W(X)      ; LOAD F2-HI INTO A.
F678 E8                      ADC A, W(B)     ; ADD F1-HI WITH CARRY FROM LO ADD.
```

```
F679 07                      IF C
F67A 4A                      JP $ADJEX        ; IF CARRY, NEED TO INCREASE EXP.
F67B F1                      X A, W(X+)       ; STORE RESULT IN F2-HI.
F67C A6FFFCC4A8              LD A, W(SP-4)    ; GET C-EXP.00000000.
F681 8FDA                    OR A, M(X)       ; INTRODUCE SIGN.
F683 F3                      K A, W(X-)       ; STORE IN F2-EXP.F2-SIGN.
F684 5B                      JP $ROUND
                     ; GET HERE MEANS NEED TO INCREASE EXP BY 1.
                     $ADJEX:
F685 D7                      RRC A
F686 F3                      X A, W(X-)
F687 F4                      LD A, W(X)
F688 D7                      RRC A
F689 F1                      X A, W(X+)
F68A F0                      LD A, W(X+)      ; X NOW POINTS TO F2-EXP.F2-SIGN.
F68B A6FFFCC4A8              LD A, W(SP-4)    ; GET C-EXP.00000000.
F690 B80100                  ADD A, 0100      ; INCREASE EXP BY 1.
F693 07                      IF C
F694 B4F998                  JMPL OVRFL
F697 BDFEFF                  IFGT A, 0FEFF    ; IS BIASED EXPONENT 255-DEC ?
F69A B4F992                  JMPL OVRFL
F69D 8FDA                    OR A, M(X)
F69F F3                      X A, W(X-)
                     ; NEED TO ROUND THE RESULT. X POINTS TO C-HI.
                     $ROUND:
F6A0 B5F9FB                  JSRL SROUND
                     ; FINAL CHECK OF EXPONENT.
F6A3 D0                      LD A, M(X+)      ; X NOW POINTS TO C-EXP.
F6A4 D2                      LD A, M(X-)
F6A5 9C00                    IFEQ A, 0
F6A7 B4F974                  JMPL UNDFL
F6AA 90FE                    IFGT A, 0FE
F6AC B4F980                  JMPL OVRFL
F6AF F2                      LD A, W(X-)
F6B0 F2                      LD A, W(X-)      ; X NOW POINTS TO C-LO.
F6B1 B5F9C8      +           JSR FPAK         ; PACK C.
F6B4 3FC4                    POP SP           ; SET UP SP FOR RETURN.
F6B6 3FCC                    POP B
F6B8 3FCE                    POP X
F6BA 3C                      RET
                     ;
                             .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    37
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FMULT.MAC

                                         .FORM 'FMULT.MAC'
                                         .INCLD FMULT.MAC
                                         .TITLE FMULT
                                         .LOCAL
                         ;
                         ; SUBROUTINE TO MULTIPLY TWO SP FLOATING POINT NUMBERS.
                         ;       C = F1*F2
                         ;
                         ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                         ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                         ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                         ;
                         ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                         ; STACK POINTER ON ENTRY, THEN
                         ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                         ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                         ;
                         ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                         ; REGS. X AND B ARE PRESERVED.
                         ;
                         FMULT:
     F6BB AFCE                   PUSH X          ; SAVE X ON ENTRY.
     F6BD AFCC                   PUSH B          ; SAVE B ON ENTRY.
                         ; SAVE ADDRESS OF F2-R0 IN TMP1.
     F6BF ACC4CE                 LD X, SP
     F6C2 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
     F6C7 ACCE00                 LD TMP1, X      ; SAVE IN TMP1.
                         ; CHECK AND SEE IF F1 IS A NAN.
     F6CA B5F973      +          JSR FNACHK
     F6CD 07                     IF C
     F6CE B4F93E                 JMPL FNAN       ; F1 IS A NAN.
                         ; CHECK AND SEE IF F2 IS A NAN.
     F6D1 ACCACC                 LD B, K
     F6D4 ACC8CE                 LD X, A
     F6D7 A20200A8               LD A, W(TMP1+2)
     F6DB ACC8CA                 LD K, A
     F6DE AECE                   X A, X
     F6E0 B5F95D      +          JSR FNACHK
     F6E3 07                     IF C
     F6E4 B4F928                 JMPL FNAN       ; F2 IS NAN.
                         ; CHECK AND SEE IF F2 IS ZERO.
     F6E7 B5F967      +          JSR FZCHK
     F6EA 07                     IF C
     F6EB 94DC                   JMP $CZERO      ; F2 IS ZERO.
                         ; CHECK AN SEE IF F1 IS ZERO.
     F6ED ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
     F6FD B5F95E      +          JSR FZCHK
     F6F3 07                     IF C
     F6F4 94D3                   JMP $CZERO      ; F1 IS ZERO.
                         ; GET HERE MEANS NORMAL MULTIPLICATION.
                         ; UNPACK F1 AND F2.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    38
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

F6F6 ACC4CE                 LD X, SP        ; X POINTS TO F1-LO.
F6F9 8210C4F8               ADD SP, 010     ; MOVE SP PAST LOCAL STORAGE.
F6FD AFCE                   PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
F6FF B5F95F      +          JSR FUNPAK      ; UNPACK F1.
F702 AC00CC                 LD B, TMP1      ; B NOW POINTS TO F2-R0.
F705 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
F708 E0                     LOS A, W(B+)    ; LOAD F2-R0 INTO A.
F709 40                     NOP
F70A AECA                   X A, K
F70C E4                     LD A, W(B)
F70D AECA                   X A, K          ; LOAD F2-R1 INTO K.
F70F B5F94F      +          JSR FUNPAK      ; UNPAK F2.
                    ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
F712 F2                     LD A, W(X-)
F713 AC00CC                 LD B, TMP1
F716 E2                     LDS A, W(B-)
F717 40                     NOP
                    ; COMPUTE C-EXP AND C-SIGN AND STORE IN F2-EXP AND F2-SIGN.
F718 F4                     LD A, W(X)      ; A IS (EEEEEEEE-F2).(SSSSSSSS-F2)
F719 C7                     SHR A           ; SHR SINCE SUM OF EXPS CAN BE 9 BITS.
F71A ACC8CA                 LD K, A         ; K IS (DEEEEEEEE-F2).(SSSSSSS-F2)
F71D E4                     LD A, W(B)      ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
F71E B9FF00                 AND A, 0FF00    ; MASK OUT SIGN BITS.
F721 C7                     SHR A           ; A IS (DEEEEEEEE-F1).(0000000)
F722 96CAF8                 ADD A, K        ; A IS (EEEEEEEEE-C).(SSSSSSS-F2)
F725 F6                     ST A, W(X)      ; STORE IN F2-SIGN.
F726 E2                     LOS A, W(B-)    ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
F727 40                     NOP
F728 99FF                   AND A, 0FF      ; MASK OUT EXP BITS.
F72A C7                     SHR A           ; A IS (000000000SSSSSSS-F1)
F72B 8FFB                   XOR A, W(X)     ; A IS (EEEEEEEEESSSSSSS-C)
F72D B8C080                 ADD A, 0C080    ; REMOVE EXCESS BIAS OF 127-DEC FROM EXP.
F730 07                     IF C
F731 46                     JP $EXCH2       ; IF CARRY, THEN NO UNDERFLOW NOW
                    ; CHECK TO SEE IF EXP IS ZERO. IF NOT, UNDERFLOW FOR SURE.
F732 E7                     SHL A
F733 07                     IF C
F734 B4F8E7                 JMPL UNDFL      ; UNDERFLOW, SO JUMP.
F737 C7                     SHR A           ; RESTORE BIT SHIFTED OUT (0).
                    ; CHECK FOR EXPONENT OVERFLOW.
                    $EXCH2:
F738 E7                     SHL A
F739 07                     IF C            ; IF C IS 1,
F73A B4F8F2                 JMPL OVRFL      ; THEN OVERFLOW FOR SURE.
F73D 96C817                 IF A.7
F740 96C808                 SET A.0         ; RESTORE LAST BIT OF SIGN.
F743 F3                     X A, W(X-)      ; STORE C-EXP. C-SIGN IN F2-EXP.F2-SIGN.
                    ;
                    ; MULTIPLY THE MANTISSA.
                    ; FIRST COMPUTE F1-HI*F2-HI.
F744 F2                     LD A, W(X-)
```

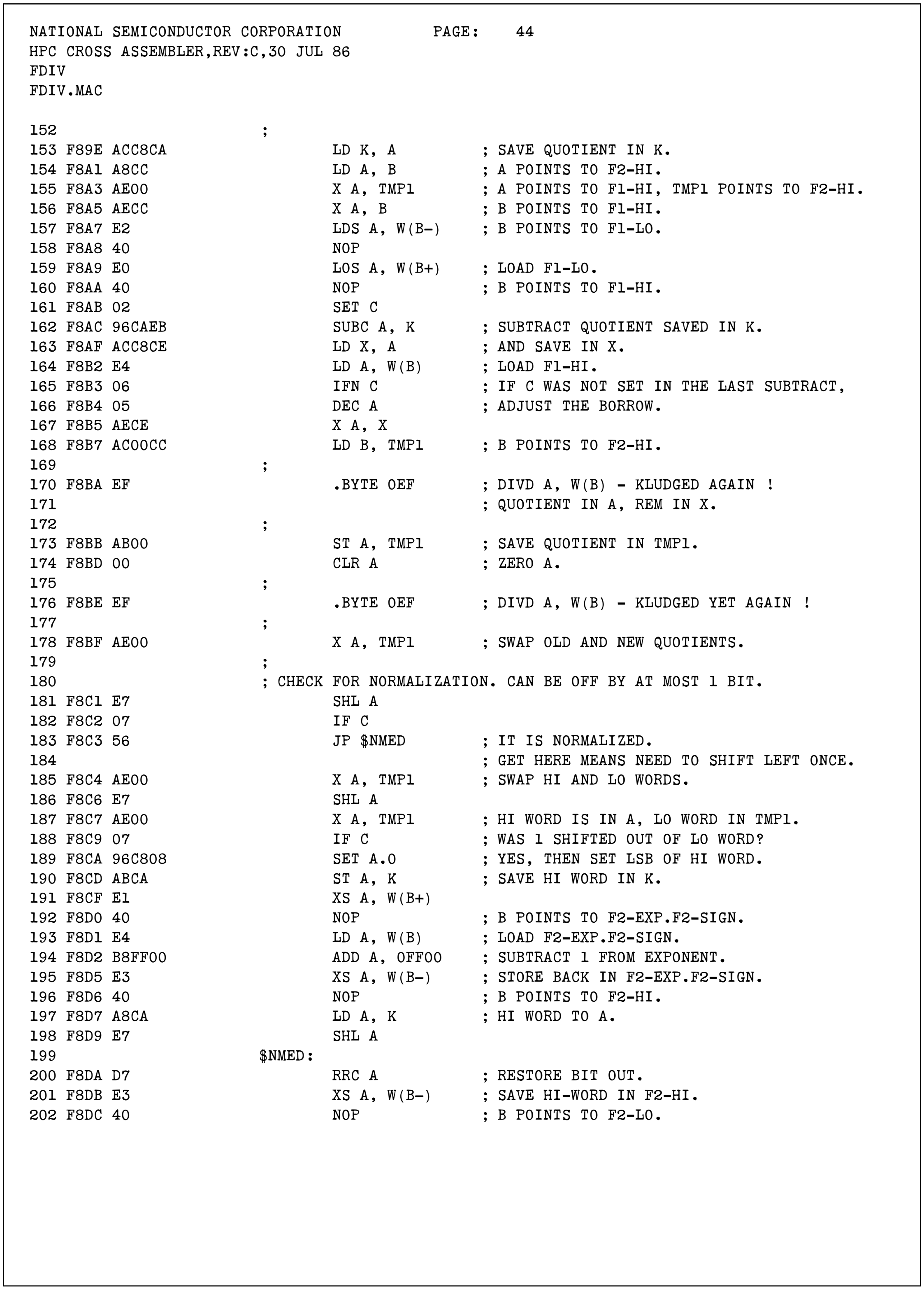
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    39
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

F745 ACCE00                     LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
F748 FE                         MULT A, W(B)
F749 A6FFFAC4AB                 ST A, W(SP-6)   ; STORE LOW WORD OF PRODUCT ON STACK.
F74E AECE                       X A, X
F750 A6FFFCC4AB                 ST A, W(SP-4)   ; STORE HIGH WORD OF PRODUCT ON STACK.
                     ; NOW COMPUTE F1-HI*F2-LO.
F755 AC00CE                     LD X, TMP1
F758 F0                         LD A, W(X+)
F759 ACCE00                     LD TMP1, X      ; TMP1 NOW POINTS TO F2-HI.
F75C FE                         MULT A, W(B)
F75D AECE                       X A, X
F75F A6FFFAC4F8                 ADD A, W(SP-6)  ; ADD LOW WORD OF LAST PROD. TO HIGH WORD.
F764 A6FFFAC4AB                 ST A, W(SP-6)
F769 07                         IF C
F76A A6FFFCC4A9                 INC W(SP-4)     ; IF CARRY, INCREASE HIGH WORD BY 1.
                     ; FINALLY COMPUTE F1-LO*F2-HI.
F76F E2                         LDS A, W(B-)    ; ADJUST B TO POINT TO F1-LO.
F770 40                         NOP
F771 AC00CE                     LD X, TMP1
F774 F4                         LD A, W(X)
F775 FE                         MULT A, W(B)
F776 AECE                       X A, X
F778 A6FFFAC4F8                 ADD A, W(SP-6)  ; ADD LOW WORD ACCUMULATED SO FAR.
F77D A6FFFAC4AB                 ST A, W(SP-6)
F782 A6FFFCC4AB                 LD A, W(SP-4)   ; A CONTAINS HIGH WORD OF PRODUCT.
F787 07                         IF C            ; IF CARRY ON LAST LOW WORD ADD,
F788 04                         INC A           ; THEN INCREASE HIGH WORD.
                     ;
                     ; MANTISSA MULTIPLICATION DONE. NOW CHECK FOR NORMALIZATION.
F789 AC00CE                     LD X, TMP1
F78C BD7FFF                     IFGT A, 07FFF   ; IS MSB OF PRODUCT 1 ?
F78F 4D                         JP $EXINC       ; YES, INCREASE MANTISSA.
                                                ; NEED TO SHIFT MANTISSA LEFT BY 1 BIT.
F790 E7                         SHL A
F791 F3                         X A, W(X-)
F792 A6FFFAC4AB                 LD A, W(SP-6)
F797 E7                         SHL A
F798 F1                         X A, W(X+)
F799 07                         IF C            ; DID SHIFT OF LOW WORD PUSH OUT A 1 ?
F79A 8F08                       SET W(X).0      ; YES SO SET LSB OF HIGH WORD.
F79C 51                         JP $ROUND       ; GO TO ROUNDING CODE.
                     $EXINC:
                     ; NEED TO INCREASE EXPONENT BY 1. REMEMBER X POINTS TO F2-HI.
F79D F3                         X A, W(X-)      ; A CONTAINS HIGH WORD, X POINTS TO F2-LO.
F79E A6FFFAC4A8                 LD A, W(SP-6)   ; GET LOW WORD.
F7A3 F1                         X A, W(X+)      ; STORE LOW WORD.
F7A4 F0                         LD A, W(X+)
F7A5 F4                         LD A, W(X)      ; GET C-EXP.C-SIGN
F7A6 B80100                     ADD A, 0100     ; INCREASE C-EXP.
F7A9 07                         IF C
F7AA B4F882                     JMPL OVRFL      ; EXPONENT OVERFLOW.
```

```
F7AD F3                      X A, W(X-)      ; NO OVERFLOW, SO SAVE C-EXP.C-SIGN.
                     ; ROUNDING CODE.
                     $ROUND:
F7AE B5F8ED                  JSRL SROUND
                     ; FINAL CHECK OF EXPONENT.
F7B1 D0                      LD A, M(X+)     ; X NOW POINTS TO C-EXP.
F7B2 D2                      LD A, M(X-)
F7B3 9C00                    IFEQ A, 0
F7B5 B4F866                  JMPL UNDFL
F7B8 9DFE                    IFGT A, OFE
F7BA B4F872                  JMPL OVRFL
F7BD F2                      LD A, W(X-)
F7BE F2                      LD A, W(X-)     ; X NOW POINTS TO C-LO.
F7BF B5F8BA      +           JSR FPAK        ; PACK C.
F7C2 3FC4                    POP SP          ; SET UP SP FOR RETURN.
F7C4 3FCC                    POP B
F7C6 3FCE                    POP X
F7C8 3C                      RET
                     ; EXCEPTION HANDLING.
                     ; C IS ZERO B'COS ONE OF F1 OR F2 IS ZERO.
                     $CZERO:
F7C9 00                      CLR A
F7CA ACC8CA                  LD K, A
F7CD 3FCC                    POP B
F7CF 3FCE                    POP X
F7D1 3C                      RET
                     ;
                             .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    41
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FDIV.MAC

                                    .FORM 'FDIV.MAC'
                                    .INCLD FDIV.MAC
                                    .TITLE FDIV
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO DIVIDE TWO SP FLOATING POINT NUMBERS.
                    ;    C = F1/F2
                    ;
                    ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                    ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                    ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                    ;
                    ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                    ; STACK POINTER ON ENTRY, THEN
                    ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                    ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                    ;
                    ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                    ;
                    FDIV:
F7D2 AFCE                   PUSH X
F7D4 AFCC                   PUSH B
                    ; SAVE ADDRESS OF F2-R0 IN TMP1.
F7D6 ACC4CE                 LD X, SP
F7D9 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
F7DE ACCE00                 LD TMP1, X      ; AND SAVE IN TMP1.
                    ; CHECK AND SEE IF F1 IS A NAN.
F7E1 85F85C      +          JSR FNACHK
F7E4 07                     IF C
F7E5 B4F827                 JMPL FNAN       ; F1 IS A NAN.
                    ; CHECK AND SEE IF F2 IS A NAN.
F7E8 ACCACC                 LD B, K
F7EB ACC8CE                 LD X, A
F7EE A20200A8               LD A, W(TMP1+2)
F7F2 ACC8CA                 LD K, A
F7F5 AECE                   X A. X
F7F7 B5F846      +          JSR FNACHK
F7FA 07                     IF C
F7FB B4F811                 JMPL FNAN       ; F2 IS NAN.
                    ; CHECK AND SEE IF F2 IS ZERO.
F7FE B5F850      +          JSR FZCHK
F801 07                     IF C
F802 B4F7FB                 JMPL DIVBY0     ; F2 IS ZERO.
                    ; CHECK AND SEE IF F1 IS ZERO.
F805 ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
F808 B5F846      +          JSR FZCHK
F80B 07                     IF C
F80C 94F1                   JMP $CZERO      ; F1 IS ZERO.
                    ; GET HERE MEANS NORMAL DIVISION.
                    ; UNPACK F1 AND F2.
F80E ACC4CE                 LD X, SP        ; X POINTS TO F1-L0.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    42
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

F811 8210C4F8               ADD SP, 010      ; MOVE SP PAST LOCAL STORAGE.
F815 AFCE                   PUSH X           ; SAVE SP ON STACK FOR QUICK RETURN.
F817 B5F847       +         JSR FUNPAK       ; UNPACK F1.
F81A AC00CC                 LD B, TMP1       ; B NOW POINTS TO F2-R0
F81D ACCE00                 LD TMP1, X       ; TMP1 NOW POINTS TO F2-LO.
F820 E0                     LDS A, W(B+)     ; LOAD F2-R0 INTO A.
F821 40                     NOP
F822 AECA                   X A, K
F824 E4                     LD A, W(B)
F825 AECA                   X A, K           ; LOAD F2-R1 INTO K.
F827 B5F837       +         JSR FUNPAK       ; UNPAK F2.
                    ;
                    ; ENSURE THAT F1-HI IS LESS THAN F2-HI.
                    ;
F82A F2                     LD A, W(X-)      ; X POINTS TO F2-EXP.F2-SIGN.
F82B F2                     LD A, W(X-)      ; X POINTS TO F2-HI.
F82C AC00CC                 LD B, TMP1       ; B POINTS TO F2-LO.
F82F E2                     LDS A, W(B-)     ; B POINTS TO F1-EXP.F1-SIGN.
F830 40                     NOP
F831 E2                     LDS A, W(B-)     ; LOAD F1-EXP.F1-SIGN.
F832 40                     NOP              ; B POINTS TO F1-HI.
F833 ACC8CA                 LD K, A          ; SAVE F1-EXP.F1-SIGN IN K.
F836 F4                     LD A, W(X)       ; LOAD F2-HI.
F837 FD                     IFGT A, W(B)     ; IS F2-HI > FI-HI ?
F838 51                     JP $FEXSN        ; YES, SO ALL IS WELL.
                                             ; GET HERE MEANS NEED TO SHR F1,
                                             ; AND INCREASE ITS EXPONENT.
F839 E0                     LOS A, W(B+)     ; GET FI-HI.
F83A 40                     NOP              ; B POINTS TO F1-EXP.F1-SIGN.
F83B AECA                   X A, K           ; SWAP F1-EXP.F1-SIGN AND F1-HI.
F83D B80100                 ADD A, 0100      ; INCREASE F1-EXP BY 1.
F840 E3                     XS A, W(B-)      ; STORE BACK IN F1-EXP.F1-SIGN.
F841 40                     NOP              ; B POINTS TO F1-HI.
F842 E4                     LD A, W(B)       ; LOAD F1-HI.
F843 C7                     SHR A
F844 E3                     XS A, W(B-)      ; STORE BACK IN F1-HI.
F845 40                     NOP              ; B POINTS TO F1-LO.
F846 E4                     LD A, W(B)       ; LOAD F1-LO.
F847 D7                     RRC A
F848 E1                     XS A, W(B+)      ; PUT IT BACK IN F1-LO.
F849 40                     NOP              ; B POINTS TO F1-HI.
                    ;
                    $FEXSN:
                    ; DETERMINE C-EXP AND C-SIGN.
F84A F0                     LD A, W(X+)      ; X POINTS TO F2-EXP.F2-SIGN.
F84B E0                     LDS A, W(B+)     ; B POINTS TO F1-EXP.F1-SIGN.
F84C 40                     NOP
F84D F4                     LD A, W(X)       ; LOAD F2-EXP.F2-SIGN.
F84E B9FF00                 AND A, 0FF00     ; MASK OUT THE SIGN.
F851 C7                     SHR A            ; ALLOW 9 BITS FOR EXP CALCULATIONS.
F852 ACC8CA                 LD K, A          ; SAVE IT IN K.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    43
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

F855 E4                         LD A, W(B)      ; LOAD F1-EXP.F1-SIGN.
F856 B9FF00                     AND A, 0FF00    ; MASK OUT SIGN.
F859 C7                         SHR A
F85A 02                         SET C
F85B 96CAEB                     SUBC A, K       ; SUBTRACT THE EXPONENTS.
                                                ; NOTE THAT NOW THE MS 9 BITS
                                                ; OF A CONTAIN A 2'S COMP. INTEGER.
F85E B7000000                   LD TMP1, 0
F862 E7                         SHL A
F863 07                         IF C
F864 B700FF00                   LD TMP1, 0FF
F868 D7                         RRC A           ; SAVE SIGN OF NUMBER IN TMP1.
F869 B83F00                     ADD A, 03F00    ; RESTORE IEEE BIAS.
F86C E7                         SHL A           ; MAKE EXPONENT 8 BITS.
F86D 06                         IFN C           ; NO CARRY ?
F86E 49                         JP $FSIGN       ; THEN ALL IS WELL.
F86F 960010                     IF TMP1.0       ; WAS EXP NEGATIVE BEFORE ?
F872 B4F7A9                     JMPL UNDFL      ; YES, SO UNDERFLOW.
F875 B4F7B7                     JMPL OVRFL      ; OTHERWISE OVERFLOW.
                        ;
                        $FSIGN:
                        ; C-EXP HAS BEEN COMPUTED. NOW FIND C-SIGN.
                        ; BUT FIRST TAKE CARE OF SPECIAL OVER/UNDERFLOW CASES.
F878 BCFF00                     IFEQ A, 0FF00
F87B B4F7B1                     JMPL OVRFL
F87E 9C00                       IFEQ A, 0
F880 B4F79B                     JMPL UNDFL
F883 AB00                       ST A, TMP1      ; SAVE C-EXP.00000000 IN TMP1.
F885 F4                         LD A, W(X)      ; LOAD F2-EXP.F2-SIGN.
F886 99FF                       AND A, 0FF      ; MASK OUT F2-EXP.
F888 FB                         XOR A, W(B)     ; A NOW HAS F1-EXP.C-SIGN.
F889 99FF                       AND A, 0FF      ; MASK OUT F1-EXP.
F88B 9600FA                     OR A, TMP1      ; BRING IN C-EXP.
F88E F3                         X A, W(X-)      ; STORE IN F2-EXP.F2-SIGN.
                                                ; X POINTS TO F2-HI.
F88F F2                         LD A, W(X-)     ; X POINTS TO F2-LO.
F890 E2                         LDS A, W(B-)    ; B POINTS TO F1-HI.
F891 40                         NOP
                        ;
                        ; NOW DO THE MANTISSA DIVISION.
                        ;
F892 F0                         LD A, W(X+)     ; LOAD F2-LO. X POINTS TO F2-HI.
F893 ACCE00                     LD TMP1, X      ; SAVE ADDRESS OF F2-HI IN TMP1.
F896 FE                         MULT A, W(B)    ; COMPUTE F2-LO*F1-HI.
                                                ; X CONTAINS MS WORD AND A IS LS WORD.
                        ;
F897 AECC                       X A, B          ; A POINTS TO F1-HI, B CONTAINS LS WORD.
F899 AE00                       X A, TMP1       ; A POINTS TO F2-HI, TMP1 POINTS TO F1-HI.
F89B AECC                       X A, B          ; A CONTAINS LS WORD, B POINTS TO F2-HI.
                        ;
F890 EF                         .BYTE 0EF       ; DIVD A, W(B) - KLUDGED !!
```



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    44
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

                     ;
F89E ACC8CA                  LD K, A         ; SAVE QUOTIENT IN K.
F8A1 A8CC                    LD A, B         ; A POINTS TO F2-HI.
F8A3 AE00                    X A, TMP1       ; A POINTS TO F1-HI, TMP1 POINTS TO F2-HI.
F8A5 AECC                    X A, B          ; B POINTS TO F1-HI.
F8A7 E2                      LDS A, W(B-)    ; B POINTS TO F1-LO.
F8A8 40                      NOP
F8A9 E0                      LOS A, W(B+)    ; LOAD F1-LO.
F8AA 40                      NOP             ; B POINTS TO F1-HI.
F8AB 02                      SET C
F8AC 96CAEB                  SUBC A, K       ; SUBTRACT QUOTIENT SAVED IN K.
F8AF ACC8CE                  LD X, A         ; AND SAVE IN X.
F8B2 E4                      LD A, W(B)      ; LOAD F1-HI.
F8B3 06                      IFN C           ; IF C WAS NOT SET IN THE LAST SUBTRACT,
F8B4 05                      DEC A           ; ADJUST THE BORROW.
F8B5 AECE                    X A, X
F8B7 AC00CC                  LD B, TMP1      ; B POINTS TO F2-HI.
                     ;
F8BA EF                      .BYTE OEF       ; DIVD A, W(B) - KLUDGED AGAIN !
                                             ; QUOTIENT IN A, REM IN X.
                     ;
F8BB AB00                    ST A, TMP1      ; SAVE QUOTIENT IN TMP1.
F8BD 00                      CLR A           ; ZERO A.
                     ;
F8BE EF                      .BYTE OEF       ; DIVD A, W(B) - KLUDGED YET AGAIN !
                     ;
F8BF AE00                    X A, TMP1       ; SWAP OLD AND NEW QUOTIENTS.
                     ;
                     ; CHECK FOR NORMALIZATION. CAN BE OFF BY AT MOST 1 BIT.
F8C1 E7                      SHL A
F8C2 07                      IF C
F8C3 56                      JP $NMED        ; IT IS NORMALIZED.
                                             ; GET HERE MEANS NEED TO SHIFT LEFT ONCE.
F8C4 AE00                    X A, TMP1       ; SWAP HI AND LO WORDS.
F8C6 E7                      SHL A
F8C7 AE00                    X A, TMP1       ; HI WORD IS IN A, LO WORD IN TMP1.
F8C9 07                      IF C            ; WAS 1 SHIFTED OUT OF LO WORD?
F8CA 96C808                  SET A.0         ; YES, THEN SET LSB OF HI WORD.
F8CD ABCA                    ST A, K         ; SAVE HI WORD IN K.
F8CF E1                      XS A, W(B+)
F8D0 40                      NOP             ; B POINTS TO F2-EXP.F2-SIGN.
F8D1 E4                      LD A, W(B)      ; LOAD F2-EXP.F2-SIGN.
F8D2 B8FF00                  ADD A, OFF00    ; SUBTRACT 1 FROM EXPONENT.
F8D5 E3                      XS A, W(B-)     ; STORE BACK IN F2-EXP.F2-SIGN.
F8D6 40                      NOP             ; B POINTS TO F2-HI.
F8D7 A8CA                    LD A, K         ; HI WORD TO A.
F8D9 E7                      SHL A
                     $NMED:
F8DA D7                      RRC A           ; RESTORE BIT OUT.
F8DB E3                      XS A, W(B-)     ; SAVE HI-WORD IN F2-HI.
F8DC 40                      NOP             ; B POINTS TO F2-LO.
```

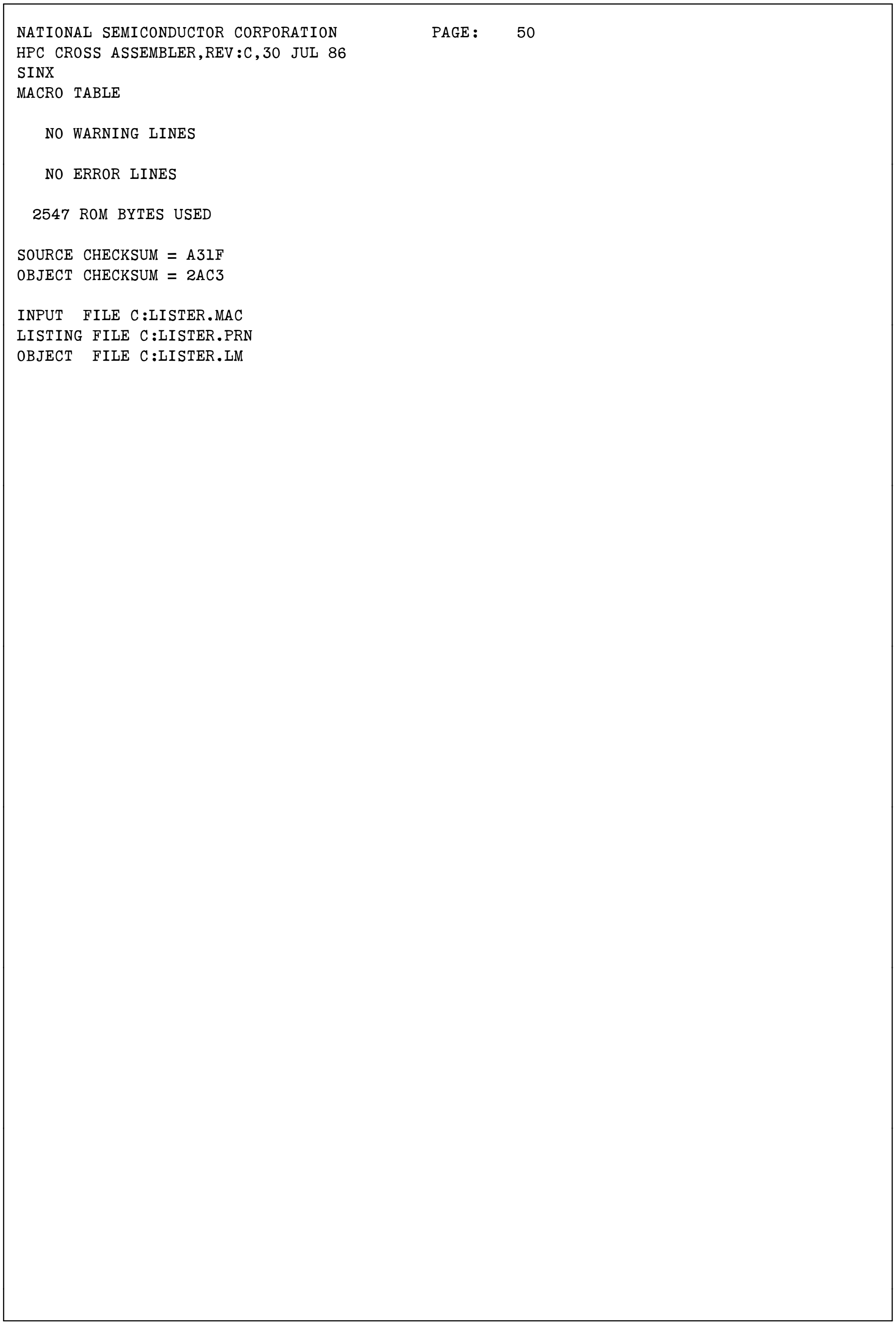
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    45
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

F8DD A800                       LD A, TMP1
F8DF E1                         XS A, W(B+)      ; SAVE C-LO.
F8E0 40                         NOP              ; B POINTS TO F2-HI.
F8E1 ACCCCE                     LD X, B          ; MOVE ADDRESS OF F2-HI TO X.
                        ;
                        ; ROUNDING CODE.
F8E4 B5F7B7                     JSRL SROUND
                        ; FINAL CHECK OF EXPONENT.
F8E7 D0                         LD A, M(X+)      ; X NOW POINTS TO C-EXP.
F8E8 D2                         LD A, M(X-)
F8E9 9C00                       IFEQ A, 0
F8EB B4F730                     JMPL UNDFL
F8EE 9DFE                       IFGT A, 0FE
F8F0 B4F73C                     JMPL OVRFL
F8F3 F2                         LD A, W(X-)
F8F4 F2                         LD A, W(X-)      ; X NOW POINTS TO C-LO.
F8F5 B5F784      +              JSR FPAK         ; PACK C.
F8F8 3FC4                       POP SP           ; SET UP SP FOR RETURN.
F8FA 3FCC                       POP B
F8FC 3FCE                       POP X
F8FE 3C                         RET
                        ; C IS ZERO B'COS F1 IS ZERO.
                        $CZERO:
F8FF 00                         CLR A
F900 ACC8CA                     LD K, A
F903 3FCC                       POP B
F905 3FCE                       POP X
F907 3C                         RET
                        ;
                                .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    46
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FSINX.MAC

  39                                     .FORM 'FSINX.MAC'
  40                                     .INCLD FSINX.MAC
   1                     ;
   2                                     .TITLE SINX
   3                                     .LOCAL
   4                     ; A VERY DIRTY APPROXIMATION TO SIN(X).
   5                     ; X SHOULD BE IN RADIANS.
   6                     ;
   7                     ; ON INPUT X SHOULD BE IN IEEE FLP FORMAT IN REGS. K AND A.
   8                     ; ON RETURN SIN(X) IS IN IEEE FLP FORMAT IN REGS. K AND A.
   9                     ;
  10                     SINX:
  11 F908 AFCE                   PUSH X          ; SAVE X.
  12 F90A AFC8                   PUSH A
  13 F90C AFCA                   PUSH K          ; X TO THE STACK.
  14 F90E 3653           -       JSRL FMULT      ; COMPUTE X^2.
  15 F910 AFC8                   PUSH A
  16 F912 AFCA                   PUSH K          ; X^2 TO THE STACK.
  17 F914 B6F994A8               LD A, W($A5LO)
  18 F918 A4F996CAAB             LD K, W($A5HI)  ; LOAD A5.
  19 F91D 3662           -       JSRL FMULT      ; COMPUTE A5*X^2.
  20 F91F AFC8                   PUSH A
  21 F921 AFCA                   PUSH K
  22 F923 B6F998A8               LD A, W($A4LO)
  23 F927 A4F99ACAAB             LD K, W($A4HI)  ; LOAD A4.
  24 F92C B5FBE6                 JSRL FSUB       ; COMPUTE A4-A5*X^2.
  25 F92F 3FCE                   POP X
  26 F931 3FCE                   POP X
  27 F933 3678           -       JSRL FMULT      ; COMPUTE
  28                                             ; X^2(A4 - A5*X^2).
  29 F935 AFC8                   PUSH A
  30 F937 AFCA                   PUSH K
  31 F939 B6F99CA8               LD A, W($A3LO)
  32 F93D A4F99ECAAB             LD K, W($A3HI)  ; LOAD A3.
  33 F942 B5FBD0                 JSRL FSUB       ; COMPUTE
  34                                             ; A3 - X^2(A4 - A5*X^2).
  35 F945 3FCE                   POP X
  36 F947 3FCE                   POP X
  37 F949 368E           -       JSRL FMULT      ; COMPUTE
  38                                             ; X^2(A3 - X^2(A4 - A5*X^2)).
  39 F94B AFC8                   PUSH A
  40 F94D AFCA                   PUSH K
  41 F94F B6F9A0A8               LD A, W($A2LO)
  42 F953 A4F9A2CAAB             LD K, W($A2HI)  ; LOAD A2.
  43 F958 B5FBBA                 JSRL FSUB       ; COMPUTE
  44                                             ; A2 - X^2(A3 - X^2(A4 - A5*X^2)).
  45 F95B 3FCE                   POP X
  46 F95D 3FCE                   POP X
  47 F95F 36A4           -       JSRL FMULT      ; COMPUTE
  48                                             ; X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
  49 F961 AFC8                   PUSH A
```

```
F963 AFCA                   PUSH K
F965 B6F9A4AB               LD A, W($A1LO)
F969 A4F9A6CAAB             LD K, W($A1HI)   ; LOAD A1.
F96E B5FBA4                 JSRL FSUB        ; COMPUTE
                                             ; A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
F971 3FCE                   POP X
F973 3FCE                   POP X
F975 36BA           -       JSRL FMULT       ; COMPUTE
                                             ; X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2)))).
F977 AFC8                   PUSH A
F979 AFCA                   PUSH K
F97B B13F80                 LD K, 03F80
F97E 00                     CLR A            ; LOAD 1.0 INTO K-A.
F97F B5FB93                 JSRL FSUB        ; COMPUTE
                                             ; 1 - ALL THE JUNK ABOVE.
F982 3FCE                   POP X
F984 3FCE                   POP X
F986 3FCE                   POP X
F988 3FCE                   POP X            ; NOW X IS AT THE TOP OF STACK.
F98A 36CF           -       JSRL FMULT       ; COMPUTE
                                     ; X(1 - X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))))).
F98C 3FCE                   POP X
F98E 3FCE                   POP X
F990 3FCE                   POP X
F992 3C                     RET
                    ;
F993 40                             .EVEN
                    ;
F994 2B32           $A5LO: .WORD 0322B
F996 D732           $A5HI: .WORD 032D7
F998 1DEF           $A4LO: .WORD 0EF1D
F99A 3836           $A4HI: .WORD 03638
F99C 010D           $A3LO: .WORD 00D01
F99E 5039           $A3HI: .WORD 03950
F9A0 8988           $A2LO: .WORD 08889
F9A2 083C           $A2HI: .WORD 03C08
F9A4 ADAA           $A1LO: .WORD 0AAAD
F9A6 2A3E           $A1HI: .WORD 03E2A
                    ;
                    ; A DIRTY APPROXIMATION TO COS(X) USING SIN(X).
                    ;
                    COSX:
F9A8 AFCE                   PUSH X
F9AA ACC8CE                 LD X, A
F9AD B6F9C8A8               LD A, W($PI2LO)
F9B1 AFC8                   PUSH A
F9B3 B6F9CAA8               LD A, W($PI2HI)
F9B7 AFC8                   PUSH A
F9B9 A8CE                   LD A, X
F9BB B5FB77                 JSRL FADD        ; COMPUTE X + PI/2.
F9BE 3FCE                   POP X
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    48
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

F9C0 3FCE                    POP X
F9C2 34BA            -       JSRL SINX       ; COMPUTE SIN(X+PI/2).
F9C4 3FCE                    POP X
F9C6 3C                      RET
                     ;
F9C7 40                              .EVEN
F9C8 DB0F            $PI2LO: .WORD 00FDB
F9CA C93F            $PI2HI: .WORD 03FC9
                     ;
                     ; A DIRTY APPROXIMATION TO TAN(X) USING SINX AND COSX.
                     ;
                     TANX:
F9CC AFCE                    PUSH X
F9CE AFCC                    PUSH B
F9D0 AFC8                    PUSH A
F9D2 AFCA                    PUSH K
F9D4 342C                    JSR COSX        ; COMPUTE COS(X)
F9D6 ACC8CE                  LD X, A
F9D9 ACCACC                  LD B, K
F9DC 3FCA                    POP K
F9DE 3FC8                    POP A
F9E0 AFCE                    PUSH X
F9E2 AFCC                    PUSH B
F9E4 34DC                    JSR SINX        ; COMPUTE SIN(X).
F9E6 3614                    JSR FDIV        ; COMPUTE TAN(X) = SIN(X)/COS(X).
F9E8 3FCC                    POP B
F9EA 3FCC                    POP B
F9EC 3FCC                    POP B
F9EE 3FCE                    POP X
F9F0 3C                      RET
                     ;
                             .END
                     ;
                     ;
FFFE 00F0                            .END LISTER
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    49
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
SYMBOL TABLE

A      00C8 W       ATOF   F13A *     B      00CC W       BFMUL  F0C7
COSX   F9A8         DIVBY0 F000       FADD   F535         FDIV   F7D2
FMULT  F6BB         FNACHK F040       FNAN   F00F         FPAK   F07C
FPERWD 0002 W       FPTRAP F09D       FSUB   F515         FTOA   F311 *
FUNPAK F061         FZCHK  F051       ISIOK  F105         K      00CA W
LISTER F000         MUL10  F118       OVRFL  F02F         PC     00C6 W
SINX   F908         SP     00C4 W     SROUND F09E         TANX   F9CC *
TMP1   0000 W       UNDFL  F01E       X      00CE W       $A10EX F1F4
$A1HI  F9A6         $A1LO  F9A4       $A2HI  F9A2         $A2LO  F9A0
$A3HI  F99E         $A3LO  F99C       $A4HI  F99A         $A4LO  F998
$A5HI  F996         $A5LO  F994       $ACCF  F1A9         $ACCM  F177
$ADDEX F1FF         $ADDMN F612       $ADEM  F3EC         $ADJEX F685
$ANORM F64A         $ANOTO F05C       $CHKOT F113         $CHNGS F361
$CSIGN F349         $CZERO F7C9       $CZERO F8FF         $DIV10 F272
$DIV10 F370         $DOLUP F481       $DTHI  F270         $DTHI  F392
$DTLO  F26E         $DTLO  F390       $ESAVE F20F         $ESIGN F1C7
$EXACC F1C9         $EXCH2 F738       $EXCHR F1BB         $EXCLP F1D4
$EXCOL F1BA         $EXCPT F2CF       $EXIN2 F0B7         $EXINC F79D
$EXIT  F0C6         $F1CHK F56E       $F1SIN F643         $F2GTR F5EB
$TEXSN F84A         $FRCOL F18B       $FSIGN F878         $GOBAK F41F
$GOON  F42B         $HIUP  F0AF       $INCOL F15C         $INCRV F40C
$INDUN F445         $INTFY F43C       $ISNAN F04C         $ISNED F298
$ISNED F3BE         $ISNXT F17D       $JAMDN F29E         $JAMDN F3C4
$JAMIT F280         $JAMIT F3A6       $JAMLP F287         $JAMLP F3AD
$LOOP1 F147         $LOOP1 F5FB       $LOOP2 F5D7         $ML4   F3EA
$MLOG2 F33F         $MSIGN F154       $MTHI  F26C         $MTHI  F396
$MTLO  F26A         $MTLO  F394       $MUL10 F398         $NAGAS F2C3
$NAN   F4CF         $NANLP F4DC       $NEG10 F20B         $NLOOP F653
$NMED  F8DA         $NORM1 F230       $NORM2 F235         $NRDUN F247
$NRLUP F237         $NTZER F585       $OV1   F666         $OVR1  F29B
$OVR1  F3C1         $PI2HI F9CA       $PI2LO F9C8         $REMV9 F354
$RNDUP F0A7         $ROUND F6A0       $ROUND F7AE         $TRADD F66C
$VDOWN F418         $VUP   F416       $ZERLP F505         $ZERO  F4E9
$ZROF1 F607         $ZROF2 F5E2
```



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    50
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
MACRO TABLE

   NO WARNING LINES

   NO ERROR LINES

  2547 ROM BYTES USED

SOURCE CHECKSUM = A31F
OBJECT CHECKSUM = 2AC3

INPUT  FILE C:LISTER.MAC
LISTING FILE C:LISTER.PRN
OBJECT  FILE C:LISTER.LM
```

A Floating Point Package for the HPC

AN-486

Lit. # 100486

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor
Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor
Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor
Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor
Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.