AN575

IEEE 754 Compliant Floating Point Routines

Author: Frank J. Testa
FJT Consulting

INTRODUCTION

This application note presents an implementation of the following floating point math routines for the PICmicro™ microcontroller families:

- float to integer conversion
- integer to float conversion
- normalize
- add/subtract
- multiply
- divide

Routines for the PIC16/17 families are provided in a modified IEEE 754 32-bit format together with versions in 24-bit reduced format.

A Glossary of terms is located on page 8.

FLOATING POINT ARITHMETIC

Although fixed point arithmetic can usually be employed in many numerical problems through the use of proper scaling techniques, this approach can become complicated and sometimes result in less efficient code than is possible using floating point methods[1]. Floating point arithmetic is essentially equivalent to arithmetic in scientific notation relative to a particular base or radix.

The base used in an implementation of floating point arithmetic is distinct from the base associated with a particular computing system. For example, the IBM System/360 is a binary computer with a hexadecimal or base-16 floating point representation, whereas the VAX together with most contemporary microcomputers are binary machines with base-2 floating point implementations. Before the establishment of the IEEE 754 floating point standard, base-2 floating point numbers were typically represented in the form

$$A = (-1)^s f \cdot 2^e,$$

$$f = \sum_{k=0}^{n-1} a(k) \cdot 2^{-(k+1)},$$

where f is the fraction or mantissa, e is the exponent or characteristic, n is the number of bits in f and $a(k)$ is the bit value where, $k = 0, \ldots, n-1$ number with a(0) = MSb, and s is the sign bit. The fraction was in normalized sign-magnitude representation with implicit MSb equal to one, and e was stored in biased form, where the bias was the magnitude of the most negative possible exponent[1,2], leading to a biased exponent eb in the form

$$eb = e + 2^{m-1},$$

where m is the number of bits in the exponent. The fraction f then satisfies the inequality

$$0.\dot{5} \le f < 1.$$

Finalization of the IEEE 754 standard[4] deviated from these conventions on several points. First, the radix point was located to the right of the MSb, yielding the representation

$$A = (-1)^s f \cdot 2^e,$$

$$f = \sum_{k=0}^{n-1} a(k) \cdot 2^{-k},$$

with f satisfying the bounds given by $1 \le f < 2$.

In order to accommodate a slot in the biased exponent format for representations of infinity to implement exact infinity arithmetic, the bias was reduced by one, yielding the biased exponent eb given by

$$eb = e + 2^{m-1} - 1.$$

In the case of single precision with $m = 8$, this results in a bias of 127. The use of biased exponents permits comparison of exponents through a simple unsigned comparator, and further results in a unique representation of zero given by $f = eb = 0$. Since our floating point implementation will not include exact infinity arithmetic

at this time, we use the IEEE 754 bias but allow the representation of the exponent to extend into this final slot, resulting in the range of exponents

$$-126 \le \boldsymbol{e} \le 128.$$

Algorithms for radix conversion are discussed in Appendix A, and can be used to produce the binary floating point representation of a given decimal number. Examples of sign-magnitude floating point representations of some decimal numbers are as follows:

Decimal	*e*	*f*
1.0	0	1.0000000
0.15625	-3	1.0100000
0.1	-4	1.10011001100....
1.23x10**3	10	1.0011001110

It is important to note that the only numbers that can be represented exactly in binary arithmetic are those which are sums of powers of two, resulting in non-terminating binary representations of some simple decimal numbers such as **0.1** as shown above, and leading to truncation errors regardless of the value of ***n***. Floating point calculations, even involving numbers admitting an exact binary representation, usually lose information after truncation to an n-bit result, and therefore require some rounding scheme to minimize such roundoff errors[1].

ROUNDING METHODS

Truncation of a binary representation to n-bits is severely biased since it always leads to a number whose absolute value is less than or equal to that of the exact value, thereby possibly causing significant error buildup during a long sequence of calculations. Simple adder-based rounding by adding the MSb to the LSb is unbiased except when the value to be rounded is equidistant from the two nearest n-bit values[1]. This small but still undesirable bias can be removed by stipulating that in the equidistant case, the n-bit value with LSb = 0 is selected, commonly referred to as the rounding to the nearest method, the default mode in the IEEE 754 standard[4,5]. The number of guard bits or extra bits of precision, is related to the sensitivity of the rounding method. Since the introduction of the hardware multiply on the PIC17[6], improvements in the floating point multiply and divide routines have provided an extra byte for guard bits, thereby offering a more sensitive rounding to the nearest method given by:

n bit value	guard bits	result
A	< 0x80	round to A
A	= 0x80	if A,LSb = 0, round to A
		if A,LSb = 1, round to A+1
A	> 0x80	round to A+1

In the equidistant case, this procedure always selects the machine number with even parity, namely, LSb = 0. However, the PIC16 implementation still uses the less sensitive single guard bit method, following the nearest neighbor rounding procedure:

n bit value	guard bit	result
A	0	round to A
A	1	if A,LSb = 0, round to A
		if A,LSb = 1, round to A+1
A+1	0	round to A+1

Currently, as a compromise between performance and rounding accuracy, a sticky bit is not used in this implementation. The lack of information regarding bits shifted out beyond the guard bits is more noticeable in the PIC16CXXX case where only one guard bit is saved.

Another interesting rounding method, is von Neumann rounding or jamming, where the exact number is truncated to n-bits and then set LSb = 1. Although the errors can be twice as large as in round to the nearest, it is unbiased and requires little more effort than truncation[1].

FLOATING POINT FORMATS

In what follows, we use the following floating point formats:

	eb	*f0*	*f1*	*f2*
IEEE754 32-bit	*s*xxx xxxx	*y* · xxx xxxx	xxxx xxxx	xxxx xxxx
MIcrochip 32-bit	xxxx xxxx	*s* · xxx xxxx	xxxx xxxx	xxxx xxxx
Microchip 24-bit	xxxx xxxx	*s* · xxx xxxx	xxxx xxxx	

Legend: *s* is the Sign bit, *y* = LSb of eb register, · = radix point

where ***eb*** is the biased 8-bit exponent, with bias = 127, ***s*** is the sign bit, and bytes **f0**, **f1** and **f2** constitute the fraction with **f0** the most significant byte with implicit MSb = 1. It is important to note that the IEEE 754 standard format[4] places the sign bit as the MSb of ***eb*** with the LSb of the exponent as the MSb of **f0**. Because of the inherent byte structure of the PIC16/17 families of microcontrollers, more efficient code was possible by adopting the above formats rather than strictly adhering to the IEEE standard. The difference between the formats consists of a rotation of the top nine bits of the representation, with a left rotate for IEEE to PIC16/17 and a right rotate for PIC16/17 to IEEE. This can be realized through the following PIC16/17 code.

IEEE_to_PIC16/17	PIC16/17_to_IEEE
`RLCF AARGB0,F`	`RLCF AARGB0,F`
`RLCF AEXP,F`	`RRCF AEXP,F`
`RRCF AARGB0,F`	`RRCF AARGB0,F`

Conversion to the 24-bit format is obtained by the rounding to the nearest from the IEEE 754 representation.

The limiting absolute values of the above floating point formats are given as follows:

	eb	*e*	|A| *f*	decimal
MAX	0xFF	128	7FFFFF	6.80564693E+38
MIN	0x01	-126	000000	1.17549435E-38

where the MSb is implicitly equal to one, and its bit location is occupied by the sign bit. The bounds for the 24-bit format are obtained by simply truncating ***f*** to 16-bits and recomputing their decimal equivalents.

EXAMPLE 1: MICROCHIP FLOAT FORMAT TO DECIMAL

To illustrate the interpretation of the previous floating point representation, consider the following simple example consisting of a 32-bit value rounded to the nearest representation of the number

$$\boldsymbol{A} = 16\pi = 50.2654824574 \approx \widehat{\boldsymbol{A}} = \text{0x84490FDB},$$

implying a biased exponent $\boldsymbol{eb} = \text{0x84}$, and the fraction or mantissa $\boldsymbol{f} = \text{0x490FDB}$. To obtain the base 2 exponent $\boldsymbol{e}$, we subtract the bias 0x7F, yielding

$$\boldsymbol{e} = \boldsymbol{eb} - \text{bias} = \text{0x84} - \text{0x7F} = \text{0x05}.$$

The fraction, with its MSb made explicit, has the binary representation

$$\begin{array}{lcccccc} & \text{C} & 9 & 0 & \text{F} & \text{D} & \text{B} \\ \boldsymbol{f} = & 1.100 & 1001 & 0000 & 1111 & 1101 & 1011 \end{array}$$

The decimal equivalent of $\boldsymbol{f}$ can then be computed by adding the respective powers of two corresponding to nonzero bits,

$$\boldsymbol{f} = 2^0 + 2^{-1} + 2^{-4} + 2^{-7} + 2^{-12} + 2^{-13} + 2^{-14} +$$

$$2^{-15} + 2^{-16} + 2^{-17} + 2^{-19} + 2^{-20} + 2^{-22} + 2^{-23} \quad = 1.5707963705,$$

evaluated in full precision on an HP48 calculator. The decimal equivalent of the representation of $\boldsymbol{A}$ can now be obtained by multiplying by the power of two defined by the exponent $\boldsymbol{e}$.

$$\widehat{\boldsymbol{A}} = 2^{e} \cdot \boldsymbol{f} = 32 \cdot 1.5707963705 = 50.265483856.$$

24-bit Format

It is important to note that the difference between this evaluation of $\widehat{\boldsymbol{A}}$ and the number $\boldsymbol{A}$ is a result of the truncation error induced by obtaining only the nearest machine representable number and not an exact representation. Alternatively, if we use the 24-bit reduced format, the result rounded to the nearest representation of $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = 16\pi = 50.2654824574 \approx \widehat{\boldsymbol{A}} = \text{0x844910},$$

leading to the fraction $\boldsymbol{f}$

$$\boldsymbol{f} = 2^0 + 2^{-1} + 2^{-4} + 2^{-7} + 2^{-11} = 1.57080078125$$

and the decimal equivalent of $\boldsymbol{A}$

$$\widehat{\boldsymbol{A}} = 2\boldsymbol{e} \cdot \boldsymbol{f} = 32 \cdot 1.57080078125 = 50.265625$$

with a correspondingly larger truncation error as expected. It is coincidence that both of these representations overestimate $\boldsymbol{A}$ in that an increment of the LSb occurs during nearest neighbor rounding in each case.

To produce the correct representation of a particular decimal number, a debugger could be used to display the internal binary representation on a host computer and make the appropriate conversion to the above format. If this approach is not feasible, algorithms for producing this representation are provided in Appendix A.

EXAMPLE 2: DECIMAL TO MICROCHIP FLOAT FORMAT

Decimal to Binary Example:

$\boldsymbol{A}$ = 0.15625 (Decimal Number)

(see algorithm A.3)

Find Exponent:

$$2^{\boldsymbol{z}} = 0.15625$$

$$\boldsymbol{z} = \frac{\ln(0.15625)}{\ln(2)} = -2.6780719$$

$$\boldsymbol{e} = \boldsymbol{int}(\boldsymbol{z}) = -3$$

Find fractional part:

$$\boldsymbol{x} = \frac{0.15625}{2^{-3}} = 1.25$$

($\boldsymbol{x}$ will always be ≥ 1)

$\boldsymbol{k} = 0$	$1.25 \geq 2^{0}$	?, yes	$\boldsymbol{a}(0) = 1$	;	$\boldsymbol{x} = 1.25 - 1 = 0.25$
$\boldsymbol{k} = 1$	$0.25 \geq 2^{-1}$	?, no	$\boldsymbol{a}(1) = 0$	;	$\boldsymbol{x} = 0.25$
$\boldsymbol{k} = 2$	$0.25 \geq 2^{-2}$	?, yes	$\boldsymbol{a}(2) = 1$	;	$\boldsymbol{x} = 0$

Therefore,

$\boldsymbol{f}$ = 1.25 decimal = 1.010 0000 0000 0000 0000 0000 binary

$\boldsymbol{A} = (-1)^{\boldsymbol{s}} \boldsymbol{f} \cdot 2^{\boldsymbol{e}}$; where $\boldsymbol{f} = \boldsymbol{x}$, $\boldsymbol{s} = 0$ (sign bit) $0.15625 = 1.25 \cdot 2^{-3}$

Now, convert 0.15625 to Microchip Float Format

$\boldsymbol{eb}$ = Biased Exponent

$\boldsymbol{eb} = \boldsymbol{e} + 7\text{Fh}$

$\boldsymbol{eb}$ = -3 + 7Fh

$\boldsymbol{eb}$ = 7Ch

Microchip Float Format:

	Exp	f0	f1	f2
0.15625 =	7C	20	00	00

Remember the MSb, $a(0) = 1$ is implied in the float number above.

FLOATING POINT EXCEPTIONS

Although the dynamic range of mathematical calculations is increased through floating point arithmetic, overflow and underflow are both possible when the limiting values of the representation are exceeded, such as in multiplication requiring the addition of exponents, or in division with the difference of exponents[2]. In these operations, fraction calculations followed by appropriate normalizing and exponent modification can also lead to overflow or underflow in special cases. Similarly, addition and subtraction after fraction alignment, followed by normalization can also lead to such exceptions.

DATA RAM REQUIREMENTS

The following contiguous data RAM locations are used by the library:

```
AARGB7 = ACCB7 = REMB3   LSB to MSB
AARGB6 = ACCB6 = REMB2
AARGB5 = ACCB5 = REMB1
AARGB4 = ACCB4 = REMB0   remainder
AARGB3 = ACCB3
AARGB2 = ACCB2
AARGB1 = ACCB1
AARGB0 = ACCB0 = ACC     AARG and ACC fract
AEXP   = EXP             AARG and ACC expon

SIGN                     sign in MSb
FPFLAGS                   exception flags,
                         option bits

BARGB3                   LSB to MSB
BARGB2
BARGB1
BARGB0                   BARG fraction
BEXP                     BARG exponent

TEMPB3
TEMPB2
TEMPB1
TEMPB0 = TEMP            temporary storage
```

The exception flags and option bits in FPFLAGS are defined as follows:

PFFLAGS	SAT	RND	DOM	NAN	FDZ	FUN	FOV	IOV
	7	6	5	4	3	2	1	0

SAT	SATurate enable bit
RND	RouNDing enable bit
DOM	DOMain error exception flag
NAN	Not-A-Number exception flag
FDZ	Floating point Divide by Zero
FUN	Floating point Underflow Flag
FOV	Floating point Overflow Flag
IOV	Integer Overflow Flag

USAGE

For the unary operations, input argument and result are in AARG. The binary operations require input arguments in AARG and BARG, and produces the result in AARG, thereby simplifying sequencing of operations.

EXCEPTION HANDLING

All routines return WREG = 0x00 upon successful completion and WREG = 0xFF, together with the appropriate FPFLAGS flag bit is set to 1 upon exception. If SAT = 0, saturation is disabled and spurious results are obtained in AARG upon an exception. If SAT = 1, saturation is enabled, and all overflow or underflow exceptions produce saturated results in AARG.

ROUNDING

With RND = 0, rounding is disabled, and simple truncation is used, resulting in some speed enhancement. If RND = 1, rounding is enabled, and rounding to the nearest LSb results.

INTEGER TO FLOAT CONVERSION

The routine FLOxxyy converts the two's complement xx-bit integer in AARG to the above yy-bit floating point representation, producing the result in AEXP, AARG. The routine initializes the exponent to move the radix point to the right of the MSb and then calls the normalize routine. An example is given by

FLO1624(12106) =
FLO1624(0x2F4A) =
0x8C3D28 =
12106.0

NORMALIZE

The routine NRMxxyy takes an unnormalized xx-bit floating point number in AEXP, AARG and left shifts the fraction and adjusts the exponent until the result has an implicit MSb = 1, producing a yy-bit result in AEXP, AARG. This routine is called by FLOxxyy, FPAyy and FPSyy, and is usually not needed explicitly by the user since all operations producing a floating point result are implicitly normalized.

FLOAT TO INTEGER CONVERSION

The routine INTxxyy converts the normalized xx-bit floating point number in AEXP, AARG, to a two's complement yy-bit integer in AARG. After removing the bias from AEXP and precluding a result of zero or integer overflow, the fraction in AARG is left shifted by AEXP and converted to two's complement representation. As an example, consider:

INT2416(123.45) =
INT2416(0x8576E6) =
0x7B =
123

ADDITION/SUBTRACTION

The floating point add routine FPAxx, takes the arguments in AEXP, AARG and BEXP, BARG and returns the sum in AEXP, AARG. If necessary, the arguments are swapped to ensure that AEXP >= BEXP, and then BARG is then aligned by right shifting by AEXP - BEXP. The fractions are then added and the result is normalized by calling NRMxx. The subtract routine FPSxx simply toggles the sign bit in BARG and calls FPAxx. Several examples are as follows:

FPA24(-0.32212E+5, 0.1120E+4) =
FPA24(0x8DFBA8, 0x890C00) =
0x8DF2E8 =
-0.31092E+5

FPS24(0.89010E+4, -0.71208E5) =
FPS24(0x8C0B14, 0x8F8B14) =
0x8F1C76 =
0.80109E+5

MULTIPLICATION

The floating point multiply routine FPMxx, takes the arguments in AEXP, AARG and BEXP, BARG and returns the product in AEXP, AARG. After testing for a zero argument, the sign and exponent of the result are computed together with testing for overflow. On the PIC17, the fractions are multiplied using the hardware multiply[6], while a standard add-shift method is used on the PIC16, in each case followed by postnormalization if necessary. For example, consider:

FPM32(-8.246268E+6, 6.327233E+6) =
FPM32(0x95FBA7F8, 95411782) =
0xACBDD0BD =
-5.217606E+13

DIVISION

The floating point divide routine FPDxx, takes the numerator in AEXP, AARG and denominator in BEXP, BARG and returns the quotient in AEXP, AARG. The PIC17 implementation uses the hardware multiply in an iterative method known as multiplicative division[6], achieving performance not possible by standard restoring or non-restoring algorithms. After a divide by zero test, an initial seed for the iteration is obtained by a table lookup, followed by a sequence of multiplicative factors for both numberator and denominator such that the denominators approach one. By a careful choice of the seed method, the quadratic convergence of the algorithm guarantees the 0.5ulp (unit in the last position) accuracy requirement in one iteration[6]. For the PIC16 family, after testing for a zero denominator, the sign and exponent of the result are computed together with testing for dividend alignment. If the argument fractions satisfy the inequality AARG >= BARG, the dividend AARG is right shifted by one bit and the exponent is adjusted, thereby resulting in AARG < BARG and the dividend is aligned. Alignment permits a valid division sequence and eliminates the need for postnormalization. After testing for overflow or underflow as appropriate, the fractions are then divided using a standard shift-subtract restoring method. A simple example is given by:

FPD24(-0.16106E+5, 0.24715E+5) =
FPD24(0x8CFBA8, 0x8D4116) =
0x7EA6D3 =
-0.65167E+0

GLOSSARY

BIASED EXPONENTS - nonnegative representation of exponents produced by adding a bias to a two's complement exponent, permitting unsigned exponent comparison together with a unique representation of zero.

FLOATING POINT UNDERFLOW - occurs when the real number to be represented is smaller in absolute value than the smallest floating point number.

FLOATING POINT OVERFLOW - occurs when the real number to be represented is larger in absolute value than the largest floating point number.

GUARD BITS - additional bits of precision carried in a calculation for improved rounding sensitivity.

LSb - least significant bit

MSb - most significant bit

NEAREST NEIGHBOR ROUNDING - an unbiased rounding method where a number to be rounded is rounded to its nearest neighbor in the representation, with the stipulation that if equidistant from its nearest neighbors, the neighbor with LSb equal to zero is selected.

NORMALIZATION - the process of left shifting the fraction of an unnormalized floating point number until the MSb equals one, while decreasing the exponent by the number of left shifts.

NSb - next significant bit just to the right of the LSb.

ONE'S COMPLEMENT - a special case of the diminished radix complement for radix two systems where the value of each bit is reversed. Although sometimes used in representing positive and negative numbers, it produces two representations of the number zero.

RADIX - the base of a given number system.

RADIX POINT - separates the integer and fractional parts of a number.

SATURATION - mode of operation where floating point numbers are fixed at there limiting values when an underflow or overflow is detected.

SIGN MAGNITUDE - representation of positive and negative binary numbers where the absolute value is expressed together with the appropriate value of the sign bit.

STICKY BIT - a bit set only if information is lost through shifting beyond the guard bits.

TRUNCATION - discarding any bits to the right of a given bit location.

TWO'S COMPLEMENT - a special case of radix complement for radix two systems where the value of each bit is reversed and the result is incremented by one. Producing a unique representation of zero, and covering the range -2^{n-1} to $2^{n-1}-1$, this is more easily applied in addition and subtraction operations and is therefore the most commonly used method of representing positive and negative numbers.

REFERENCES

1. Cavanagh, J.J.F., "Digital Computer Arithmetic," McGraw-Hill,1984.
2. Hwang, K., "Computer Arithmetic," John Wiley & Sons, 1979.
3. Scott, N.R., "Computer Number Systems & Arithmetic," Prentice Hall, 1985.
4. IEEE Standards Board, "IEEE Standard for Floating-Point Arithmetic," ANSI/IEEE Std 754-1985, IEEE, 1985.
5. Knuth, D.E., "The Art of Computer Programming, Volume 2," Addison-Wesley, 1981.
6. Testa, F. J., "AN575: Applications of the 17CXX Hardware Multiply in Math Library Routines,: Embedded Control Handbook, Microchip Technology, 1996.

APPENDIX A: ALGORITHMS FOR DECIMAL TO BINARY CONVERSION

Several algorithms for decimal to binary conversion are given below. The integer and fractional conversion algorithms are useful in both native assembly as well as high level languages. Algorithm A.3 is a more brute force method easily implemented on a calculator or in a high level language on a host computer and is portable across platforms. An ANSI C implementation of algorithm A.3 is given.

A.1 Integer conversion algorithm[3]:

Given an integer I, where $d(k)$ are the bit values of its n-bit binary representation with d(0) = LSb,

$$I = \sum_{k=0}^{n-1} d(k) \cdot 2^{k}$$

```
k=0
I(k) = I
while I(k) =! 0
   d(k) = remainder of I(k)/2
   I(k+1) = [ I(k)/2 ]
   k = k + 1
endw
```

where [ ] denotes the greatest integer function.

A.2 Fractional conversion algorithm[3]:

Given a fraction F, where $d(k)$ are the bit values of its n-bit binary representation with d(1) = MSb,

$$F = \sum_{k=1}^{n} d(k) \cdot 2^{-k}$$

```
k=0
F(k) = F
while k <= n
   d(k) = [ F(k)*2 ]
   F(k+1) = fractional part of F(k)*2
   k = k + 1
endw
```

A.3 Decimal to binary conversion algorithm:

Given a decimal number A, and the number of fraction bits n, the bits in the fraction of the above binary representation of A, $a(k)$, k = 0,2,...,n-1, where a(0) = MSb, are given by the following algorithm:

```
z = ln A / ln 2
e = int ( z )
if e > z
   e = e - 1
endif
x = A / (2**e)
k = 0
while k <= n-1
   if x >= 2**(-k)
      a(k) = 1
   else
      a(k) = 0
   endif
   x = x - a(k) * 2**(-k)
   k = k + 1
endw
```

Formally, the number A then has the floating point representation

$$A = (-1)^s f \cdot 2^e \qquad f = \sum_{k=0}^{n-1} a(k) \cdot 2^{-k}$$

A simple C implementation of algorithm A.3 is given as follows:

```
#include <stdio.h>
#include <math.h>
main()
{
    int a[32],e,k,j;
    double A,x,z;
    printf("Enter A: ");
    while(scanf("%lf",&A) == 1)
    {
        z = log(A)/log(2.);
        e = (int)z;
        if((double)e > z)e = e-1;
        x = A/pow(2.,(double)e);
        for(k=0; k<32; k++)
        {
            if(x >= pow(2.,(double)(-k)))
                a[k]=1;
            else
                a[k]=0;
            x = x - (double)a[k] *
                     pow(2., (double)(-k));
        }
        printf("e = %4i\n",e);
        printf("f = %1i.",a[0]);
        for(j=1; j<4; j++)
            printf("%1i",a[j]);
        printf(" ");
        for(k=1; k<8; k++)
        {
            for(j=0; j<4; j++)
                printf("%1i",a[k*4+j]);
            printf(" ");
        }
       printf("\n");
       printf("Enter A: ");
    }
}
```

FIGURE A-1: INTEGER TO FLOAT CONVERSION

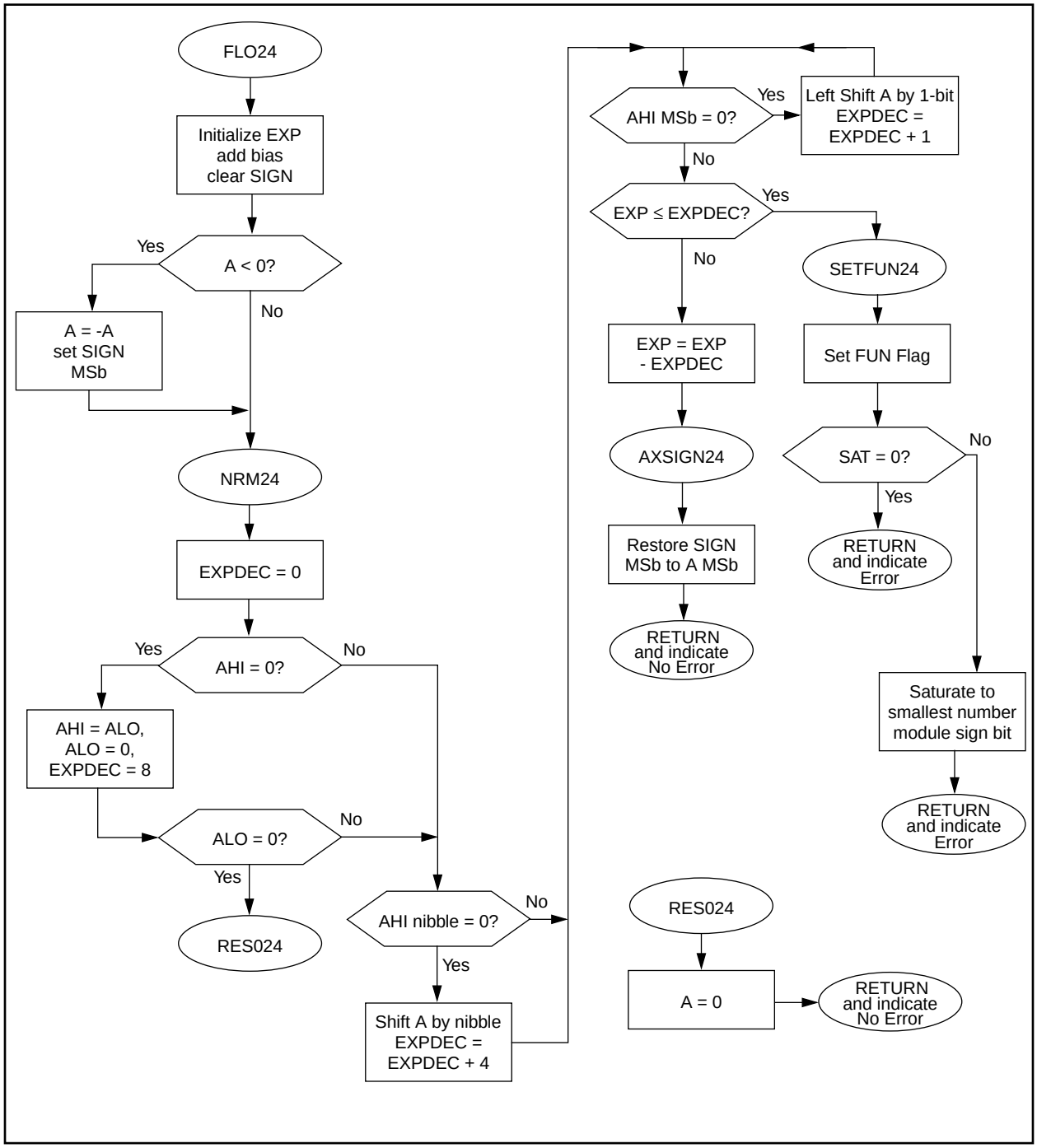

FIGURE A-2: FLOAT TO INTEGER CONVERSION

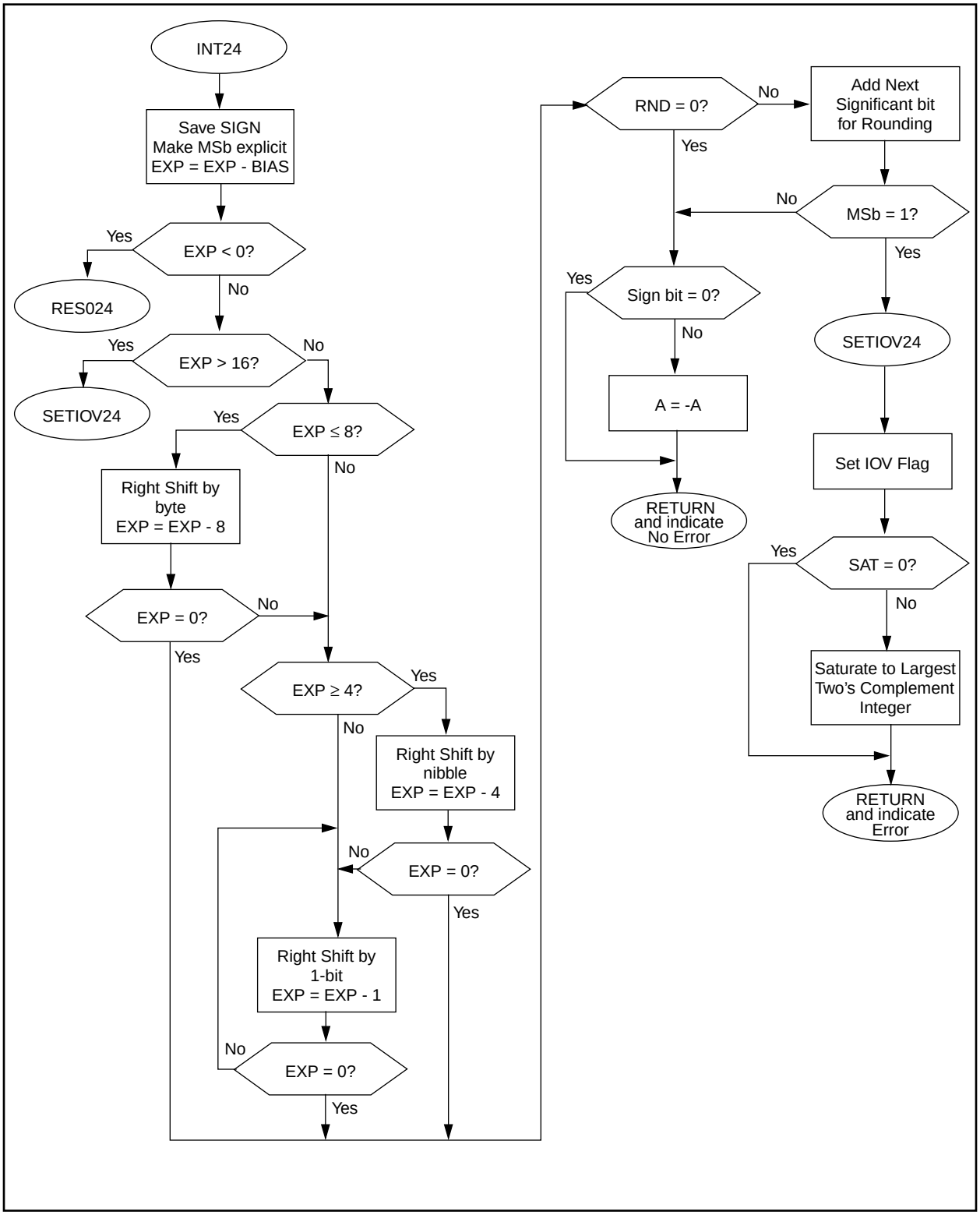

FIGURE A-3: FLOATING POINT MULTIPLY

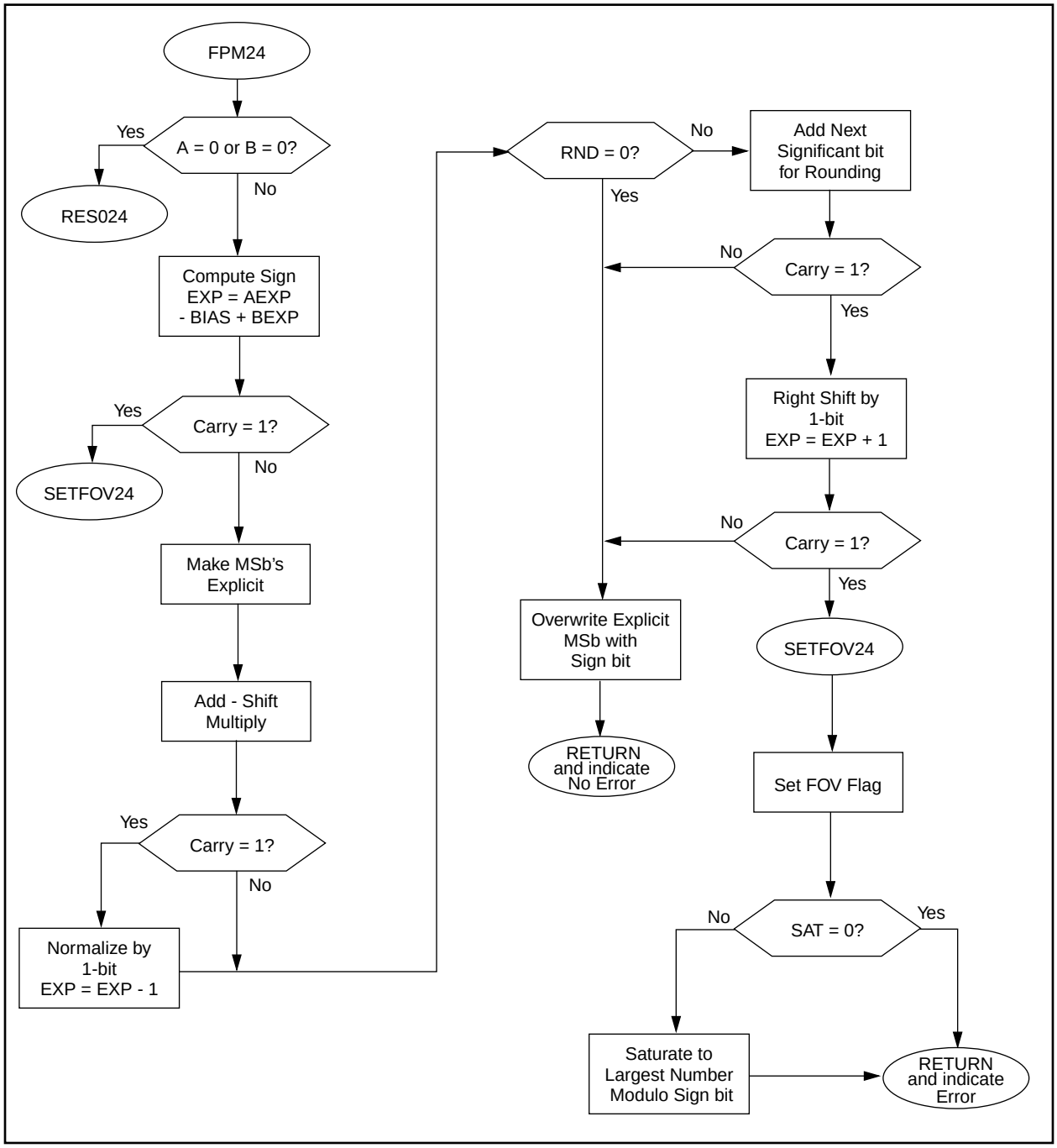

FIGURE A-4: FLOATING POINT DIVIDE

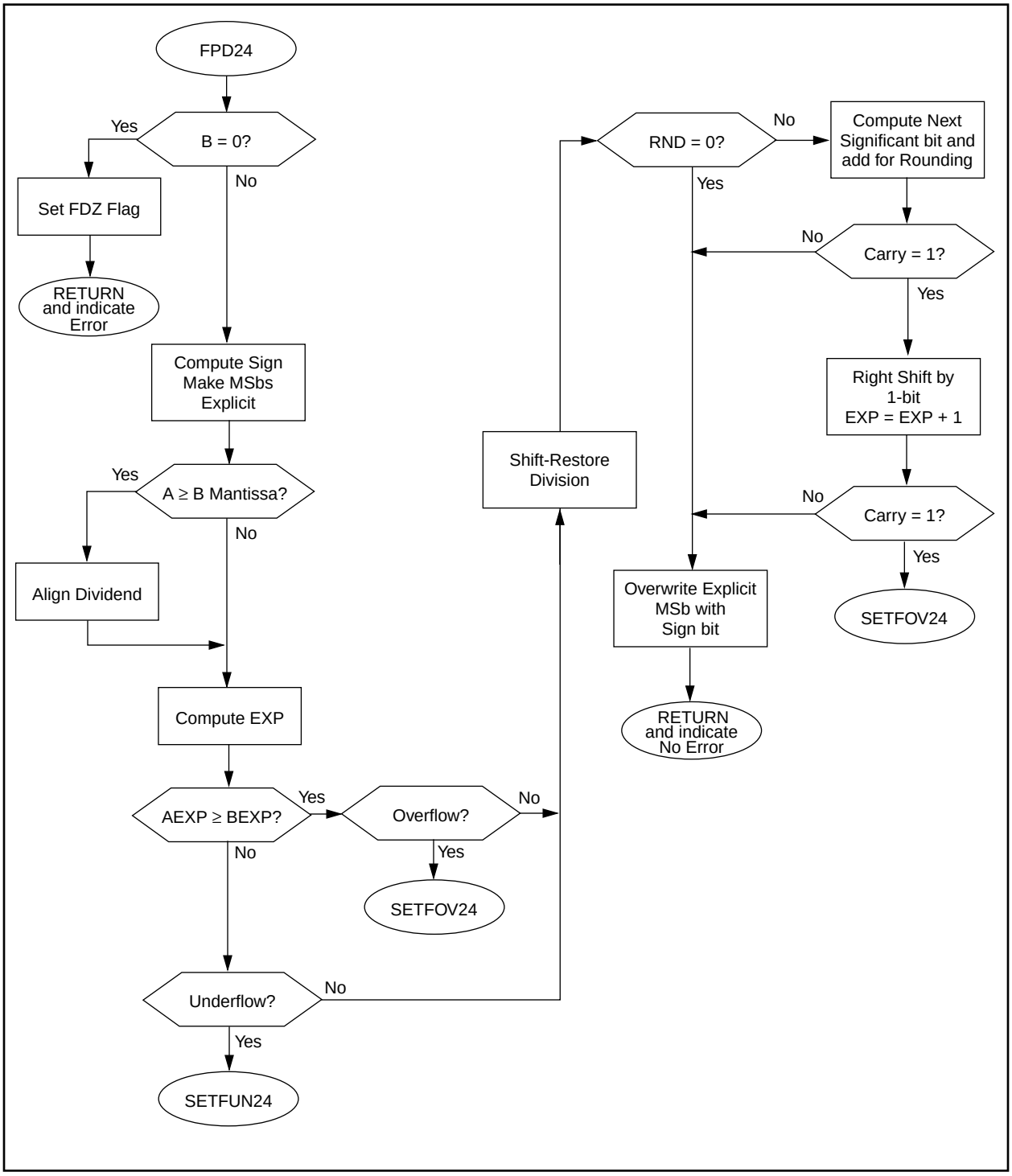

FIGURE A-5: FLOATING POINT SUBTRACT

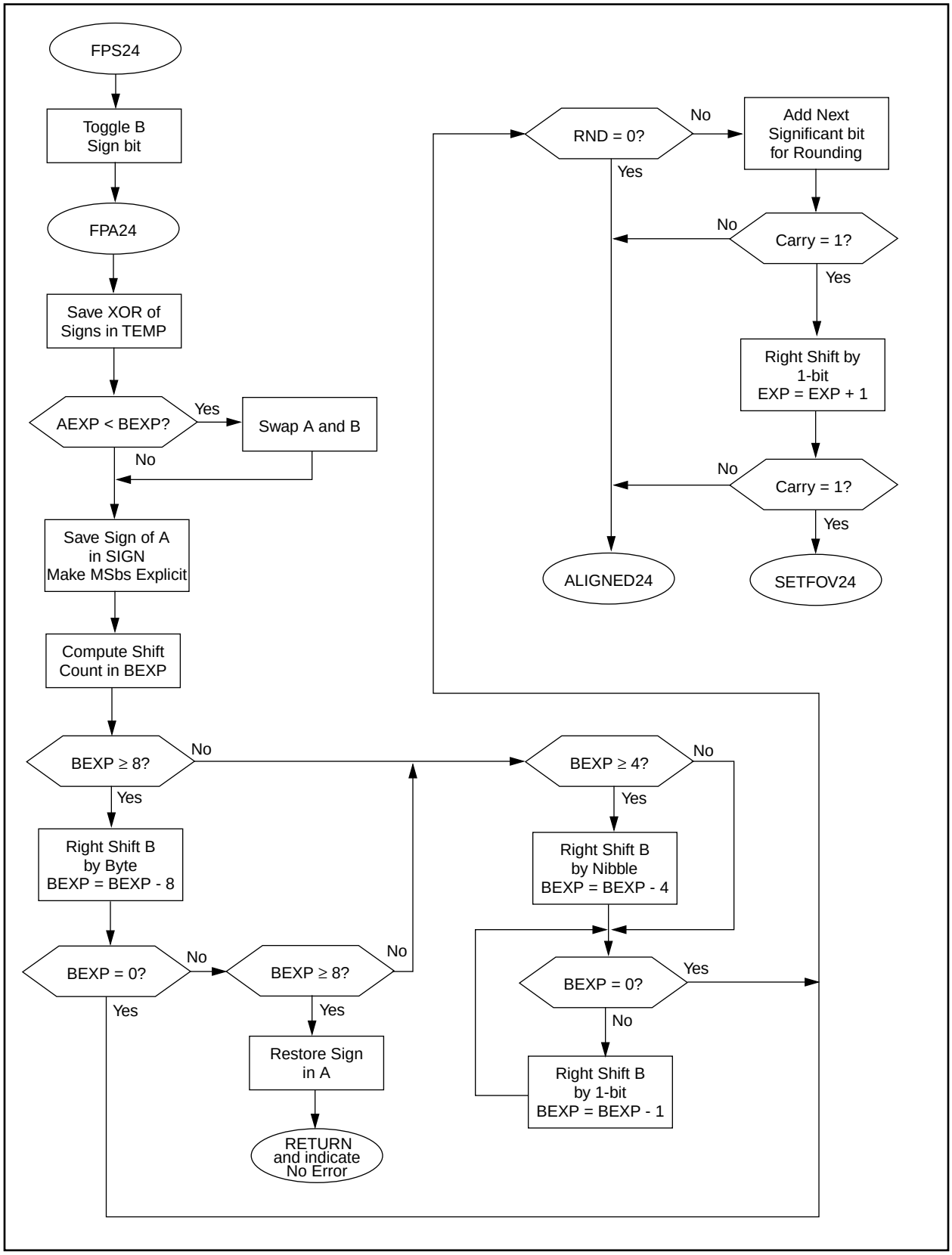

FIGURE A-6: NORMALIZATION

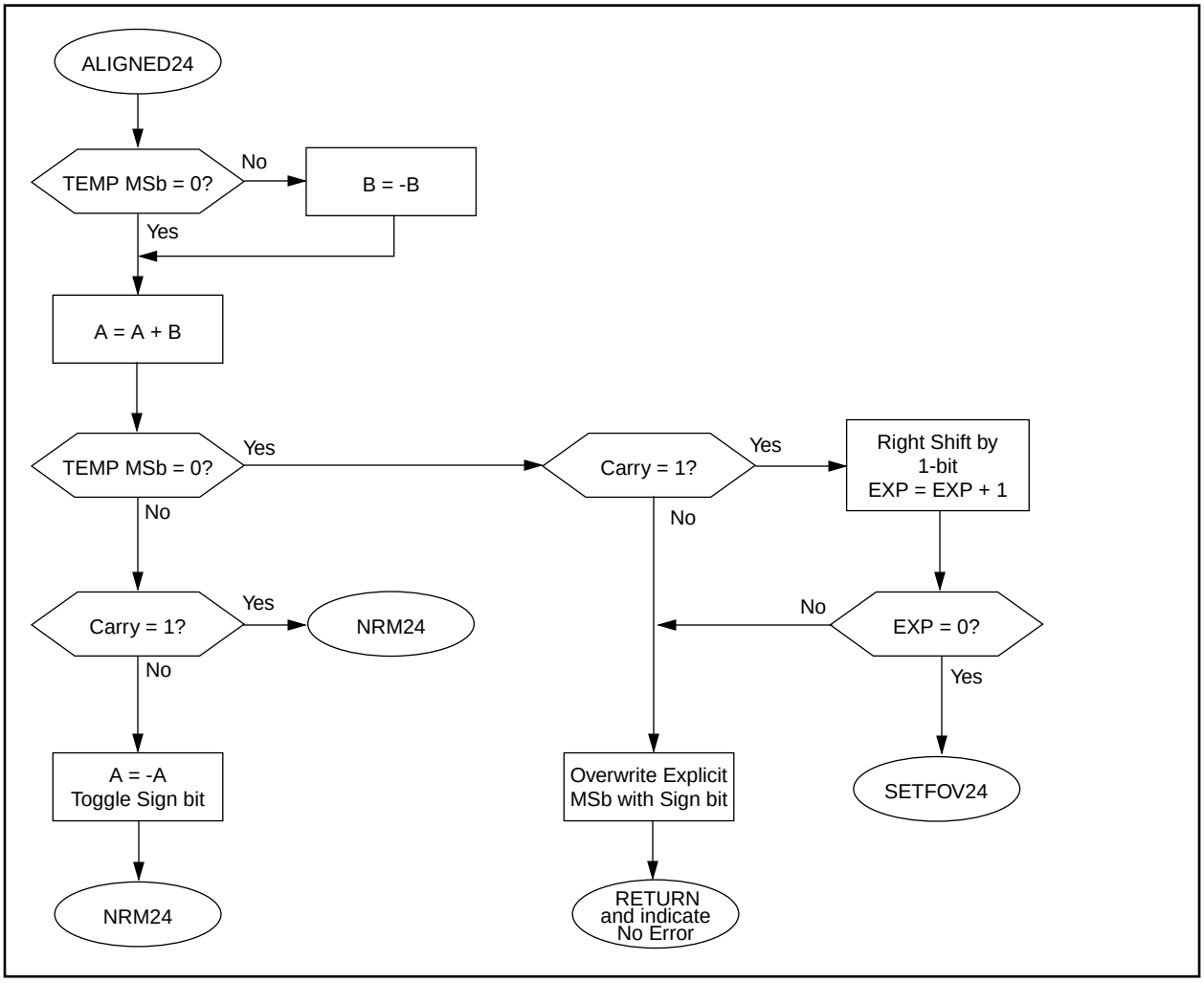

TABLE A-1: PIC17CXXX FLOATING POINT PERFORMANCE DATA

Routine	Max Cycles	Min Cycles	Program Memory	Data Memory	
FLO1624	49	34	72	6	
FLO2424	64	39	130	7	
INT2416	41	41	101	6	
INT2424	48	44	156	7	
FPA24	63	52	212	10	
FPS24	120	0	213	10	
FPM24	61	56	224	10	
FPD24	172	168	377	11	
			1465	11	Total Memory
Routine	**Max Cycles**	**Min Cycles**	**Program Memory**	**Data Memory**	
FLO2432	60	35	120	7	
FLO3232	74	40	189	8	
INT3224	47	43	155	7	
INT3232	47	43	219	8	
FPA32	66	55	329	12	
FPS32	100	83	330	12	
FPM32	101	95	382	13	
FPD32	317	312	661	14	
			2385	14	Total Memory

TABLE A-2: PIC16C5X/PIC16CXXX FLOATING POINT PERFORMANCE DATA

Routine	Max Cycles	Min Cycles	Program Memory	Data Memory	
FLO1624	81	35	37	6	
FLO2424	108	28	65	7	
INT2416	47	41	64	6	
INT2424	46	44	64	6	
FPA24	74	74	102	11	
FPS24	196	46	104	11	
FPM24	298	11	80	11	
FPD24	469	348	117	11	
			652	11	Total Memory
Routine	**Max Cycles**	**Min Cycles**	**Program Memory**	**Data Memory**	
FLO2432	83	35	52	7	
FLO3232	129	28	83	8	
INT3224	90	15	83	6	
INT3232	126	15	103	7	
FPA32	248	50	136	14	
FPS32	250	52	138	14	
FPM32	574	12	94	14	
FPD32	929	704	152	14	
			841	14	Total Memory

NOTES:

Please check the Microchip BBS for the latest version of the source code. For BBS access information, see Section 6, Microchip Bulletin Board Service information, page 6-3.

APPENDIX B:

B.1 Device Family Include File

```
;        RCS Header $Id: dev_fam.inc 1.2 1997/03/24 23:25:07 F.J.Testa Exp $

;        $Revision: 1.2 $

;  DEV_FAM.INC   Device Family Type File, Version 1.00    Microchip Technology, Inc.
;
; This file takes the defined device from the LIST directive, and specifies a
; device family type and the Reset Vector Address (in RESET_V).
;
;*******
;*******   Device Family Type, Returns one of these three Symbols (flags) set
;*******   (other two are cleared) depending on processor selected in LIST Directive:
;*******          P16C5X, P16CXX, or P17CXX
;*******   Also sets the Reset Vector Address in symbol RESET_V
;*******
;*******   File Name:   DEV_FAM.INC
;*******   Revision:    1.00.00        08/24/95       MP
;*******                1.00.01        03/21/97       AL
;*******
;
TRUE                EQU   1
FALSE               EQU   0
;
P16C5X      SET    FALSE       ; If P16C5X, use INHX8M file format.
P16CXX      SET    FALSE       ; If P16CXX, use INHX8M file format.
P17CXX      SET    FALSE       ; If P17CXX, the INHX32 file format is required
;                              ;            in the LIST directive
RESET_V     SET    0x0000      ; Default Reset Vector address of 0h
                               ;   (16Cxx and 17Cxx devices)
P16_MAP1    SET    FALSE       ; FOR 16C60/61/70/71/710/711/715/84 Memory Map
P16_MAP2    SET    FALSE       ; For all other 16Cxx Memory Maps
;
;******    16CXX  ***********
;
    IFDEF   __14000
P16CXX      SET    TRUE        ; If P14000, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C554
P16CXX      SET    TRUE        ; If P16C554, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C556
P16CXX      SET    TRUE        ; If P16C556, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C558
P16CXX      SET    TRUE        ; If P16C558, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C61
P16CXX      SET    TRUE        ; If P16C61, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
```

```
    IFDEF   __16C62
P16CXX      SET    TRUE        ; If P16C62, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C62A
P16CXX      SET    TRUE        ; If P16C62A, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C63
P16CXX      SET    TRUE        ; If P16C63, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C64
P16CXX      SET    TRUE        ; If P16C64, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C64A
P16CXX      SET    TRUE        ; If P16C64A, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C65
P16CXX      SET    TRUE        ; If P16C65, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C65A
P16CXX      SET    TRUE        ; If P16C65A, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C620
P16CXX      SET    TRUE        ; If P16C620, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C621
P16CXX      SET    TRUE        ; If P16C621, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C622
P16CXX      SET    TRUE        ; If P16C622, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C642
P16CXX      SET    TRUE        ; If P16C642, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C662
P16CXX      SET    TRUE        ; If P16C662, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C710
P16CXX      SET    TRUE        ; If P16C710, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16C71
```

```
P16CXX      SET    TRUE        ; If P16C71, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16C711
P16CXX      SET    TRUE        ; If P16C711, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16C72
P16CXX      SET    TRUE        ; If P16C72, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C73
P16CXX      SET    TRUE        ; If P16C73, use INHX8M file format.
P16_MAP2    SET    TRUE        ;
    ENDIF
;
    IFDEF   __16C73A
P16CXX      SET    TRUE        ; If P16C73A, use INHX8M file format.
P16_MAP2    SET    TRUE        ;
    ENDIF
;
    IFDEF   __16C74
P16CXX      SET    TRUE        ; If P16C74, use INHX8M file format.
P16_MAP2    SET    TRUE        ;
    ENDIF
;
    IFDEF   __16C74A
P16CXX      SET    TRUE        ; If P16C74A, use INHX8M file format.
P16_MAP2    SET    TRUE        ;
    ENDIF
;
    IFDEF   __16C84
P16CXX      SET    TRUE        ; If P16C84, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16F84
P16CXX      SET    TRUE        ; If P16F84, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16F83
P16CXX      SET    TRUE        ; If P16F83, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16CR83
P16CXX      SET    TRUE        ; If P16CR83, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16CR84
P16CXX      SET    TRUE        ; If P16CR84, use INHX8M file format.
P16_MAP1    SET    TRUE
    ENDIF
;
    IFDEF   __16C923
P16CXX      SET    TRUE        ; If P16C923, use INHX8M file format.
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16C924
P16CXX      SET    TRUE        ; If P16C924, use INHX8M file format.
```

```
P16_MAP2    SET    TRUE
    ENDIF
;
    IFDEF   __16CXX            ; Generic Processor Type
P16CXX      SET    TRUE        ; If P16CXX, use INHX8M file format.
P16_MAP2    SET    TRUE        ;
    ENDIF
;
;
;
;******    17CXX  ***********
;
;
    IFDEF   __17C42
P17CXX      SET    TRUE        ; If P17C42, the INHX32 file format is required
;                              ;            in the LIST directive
    ENDIF
;
    IFDEF   __17C43
P17CXX      SET    TRUE        ; If P17C43, the INHX32 file format is required
;                              ;            in the LIST directive
    ENDIF
;
    IFDEF   __17C44
P17CXX      SET    TRUE        ; If P17C44, the INHX32 file format is required
;                              ;            in the LIST directive
    ENDIF
;
    IFDEF   __17CXX            ; Generic Processor Type
P17CXX      SET    TRUE        ; If P17CXX, the INHX32 file format is required
;                              ;            in the LIST directive
    ENDIF
;
;******    16C5X  ***********
;
;
    IFDEF   __16C54
P16C5X      SET    TRUE        ; If P16C54, use INHX8M file format.
RESET_V     SET    0x01FF      ; Reset Vector at end of 512 words
    ENDIF
;
    IFDEF   __16C54A
P16C5X      SET    TRUE        ; If P16C54A, use INHX8M file format.
RESET_V     SET    0x01FF      ; Reset Vector at end of 512 words
    ENDIF
;
    IFDEF   __16C55
P16C5X      SET    TRUE        ; If P16C55, use INHX8M file format.
RESET_V     SET    0x01FF      ; Reset Vector at end of 512 words
    ENDIF
;
    IFDEF   __16C56
P16C5X      SET    TRUE        ; If P16C56, use INHX8M file format.
RESET_V     SET    0x03FF      ; Reset Vector at end of 1K words
    ENDIF
;
    IFDEF   __16C57
P16C5X      SET    TRUE        ; If P16C57, use INHX8M file format.
RESET_V     SET    0x07FF      ; Reset Vector at end of 2K words
    ENDIF
;
    IFDEF   __16C58A
P16C5X      SET    TRUE        ; If P16C58A, use INHX8M file format.
RESET_V     SET    0x07FF      ; Reset Vector at end of 2K words
    ENDIF
;
```

```
    IFDEF   __16C5X            ; Generic Processor Type
P16C5X      SET    TRUE        ; If P16C5X, use INHX8M file format.
RESET_V     SET    0x07FF      ; Reset Vector at end of 2K words
    ENDIF
;
;
    if ( P16C5X + P16CXX + P17CXX != 1 )
MESSG  “WARNING - USER DEFINED: One and only one device family can be selected”
MESSG  “                        May be NEW processor not defined in this file”
    endif
;
```

B.2 Math16 Include File

```
;        RCS Header $Id: math16.inc 2.4 1997/02/11 16:58:49 F.J.Testa Exp $

;        $Revision: 2.4 $

;       MATH16 INCLUDE FILE
;
;       IMPORTANT NOTE: The math library routines can be used in a dedicated application on
;       an individual basis and memory allocation may be modified with the stipulation that
;       on the PIC17, P type registers must remain so since P type specific instructions
;       were used to realize some performance improvements.

;*******************************************************************************************
;
;       GENERAL MATH LIBRARY DEFINITIONS
;
;        general literal constants

;        define assembler constants

B0                 equ        0
B1                 equ        1
B2                 equ        2
B3                 equ        3
B4                 equ        4
B5                 equ        5
B6                 equ        6
B7                 equ        7

MSB                equ        7
LSB                equ        0


;     define commonly used bits

;     STATUS bit definitions

#define        _C                STATUS,0
#define        _Z                STATUS,2

;
;       general register variables
;
    IF ( P16_MAP1 )

ACCB7            equ     0x0C
ACCB6            equ     0x0D
ACCB5            equ     0x0E
ACCB4            equ     0x0F
ACCB3            equ     0x10
ACCB2            equ     0x11
ACCB1            equ     0x12
ACCB0            equ     0x13
ACC              equ     0x13    ; most significant byte of contiguous 8 byte accumulator
;
SIGN             equ     0x15    ; save location for sign in MSB
;
TEMPB3           equ     0x1C
TEMPB2           equ     0x1D
TEMPB1           equ     0x1E
TEMPB0           equ     0x1F
TEMP             equ     0x1F    ; temporary storage
;
```

```
;       binary operation arguments
;
AARGB7          equ     0x0C
AARGB6          equ     0x0D
AARGB5          equ     0x0E
AARGB4          equ     0x0F
AARGB3          equ     0x10
AARGB2          equ     0x11
AARGB1          equ     0x12
AARGB0          equ     0x13
AARG            equ     0x13    ; most significant byte of argument A
;
BARGB3          equ     0x17
BARGB2          equ     0x18
BARGB1          equ     0x19
BARGB0          equ     0x1A
BARG            equ     0x1A    ; most significant byte of argument B
;
;       Note that AARG and ACC reference the same storage location
;
;*******************************************************************************************
;
;       FIXED POINT SPECIFIC DEFINITIONS
;
;       remainder storage
;
REMB3           equ     0x0C
REMB2           equ     0x0D
REMB1           equ     0x0E
REMB0           equ     0x0F    ; most significant byte of remainder

LOOPCOUNT       equ     0x20    ; loop counter
;
;*******************************************************************************************
;
;       FLOATING POINT SPECIFIC DEFINITIONS
;
;       literal constants
;
EXPBIAS       equ     D'127'
;
;       biased exponents
;
EXP           equ     0x14    ; 8 bit biased exponent
AEXP          equ     0x14    ; 8 bit biased exponent for argument A
BEXP          equ     0x1B    ; 8 bit biased exponent for argument B
;
;       floating point library exception flags
;
FPFLAGS       equ     0x16    ; floating point library exception flags
IOV           equ     0       ; bit0 = integer overflow flag
FOV           equ     1       ; bit1 = floating point overflow flag
FUN           equ     2       ; bit2 = floating point underflow flag
FDZ           equ     3       ; bit3 = floating point divide by zero flag
NAN           equ     4       ; bit4 = not-a-number exception flag
DOM           equ     5       ; bit5 = domain error exception flag
RND           equ     6       ; bit6 = floating point rounding flag, 0 = truncation
                              ; 1 = unbiased rounding to nearest LSB
SAT           equ     7       ; bit7 = floating point saturate flag, 0 = terminate on
                              ; exception without saturation, 1 = terminate on
                              ; exception with saturation to appropriate value

    ENDIF
;
;
    IF ( P16_MAP2 )
```

```
ACCB7           equ     0x20
ACCB6           equ     0x21
ACCB5           equ     0x22
ACCB4           equ     0x23
ACCB3           equ     0x24
ACCB2           equ     0x25
ACCB1           equ     0x26
ACCB0           equ     0x27
ACC             equ     0x27    ; most significant byte of contiguous 8 byte accumulator
;
SIGN            equ     0x29    ; save location for sign in MSB
;
TEMPB3          equ     0x30
TEMPB2          equ     0x31
TEMPB1          equ     0x32
TEMPB0          equ     0x33
TEMP            equ     0x33    ; temporary storage
;
;       binary operation arguments
;
AARGB7          equ     0x20
AARGB6          equ     0x21
AARGB5          equ     0x22
AARGB4          equ     0x23
AARGB3          equ     0x24
AARGB2          equ     0x25
AARGB1          equ     0x26
AARGB0          equ     0x27
AARG            equ     0x27    ; most significant byte of argument A
;
BARGB3          equ     0x2B
BARGB2          equ     0x2C
BARGB1          equ     0x2D
BARGB0          equ     0x2E
BARG            equ     0x2E    ; most significant byte of argument B
;
;       Note that AARG and ACC reference the same storage location
;
;*******************************************************************************************
;
;       FIXED POINT SPECIFIC DEFINITIONS
;
;       remainder storage
;
REMB3           equ     0x20
REMB2           equ     0x21
REMB1           equ     0x22
REMB0           equ     0x23    ; most significant byte of remainder

LOOPCOUNT       equ     0x34    ; loop counter
;
;*******************************************************************************************
;
;       FLOATING POINT SPECIFIC DEFINITIONS
;
;       literal constants
;
EXPBIAS         equ     D'127'
;
;       biased exponents
;
EXP             equ     0x28    ; 8 bit biased exponent
AEXP            equ     0x28    ; 8 bit biased exponent for argument A
BEXP            equ     0x2F    ; 8 bit biased exponent for argument B
;
```

```
;       floating point library exception flags
;
FPFLAGS         equ     0x2A    ; floating point library exception flags
IOV             equ     0       ; bit0 = integer overflow flag
FOV             equ     1       ; bit1 = floating point overflow flag
FUN             equ     2       ; bit2 = floating point underflow flag
FDZ             equ     3       ; bit3 = floating point divide by zero flag
NAN             equ     4       ; bit4 = not-a-number exception flag
DOM             equ     5       ; bit5 = domain error exception flag
RND             equ     6       ; bit6 = floating point rounding flag, 0 = truncation
                                ; 1 = unbiased rounding to nearest LSb
SAT             equ     7       ; bit7 = floating point saturate flag, 0 = terminate on
                                ; exception without saturation, 1 = terminate on
                                ; exception with saturation to appropriate value

;*******************************************************************************************

;        ELEMENTARY FUNCTION MEMORY

CEXP            equ        0x35
CARGB0          equ        0x36
CARGB1          equ        0x37
CARGB2          equ        0x38
CARGB3          equ        0x39

DEXP            equ        0x3A
DARGB0          equ        0x3B
DARGB1          equ        0x3C
DARGB2          equ        0x3D
DARGB3          equ        0x3E

EEXP            equ        0x3F
EARGB0          equ        0x40
EARGB1          equ        0x41
EARGB2          equ        0x42
EARGB3          equ        0x43

ZARGB0          equ        0x44
ZARGB1          equ        0x45
ZARGB2          equ        0x46
ZARGB3          equ        0x47

RANDB0          equ        0x48
RANDB1          equ        0x49
RANDB2          equ        0x4A

RANDB3          equ        0x4B

;*******************************************************************************************

;        24-BIT FLOATING POINT CONSTANTS

;        Machine precision

MACHEP24EXP     equ        0x6F                ; 1.52587890625e-5 = 2**-16
MACHEP24B0      equ        0x00
MACHEP24B1      equ        0x00

;        Maximum argument to EXP24

MAXLOG24EXP     equ        0x85                ; 88.7228391117 = log(2**128)
MAXLOG24B0      equ        0x31
MAXLOG24B1      equ        0x72

;        Minimum argument to EXP24
```

```
MINLOG24EXP     equ        0x85                ; -87.3365447506 = log(2**-126)
MINLOG24B0      equ        0xAE
MINLOG24B1      equ        0xAC

;        Maximum argument to EXP1024

MAXLOG1024EXP   equ        0x84                ; 38.531839445 = log10(2**128)
MAXLOG1024B0    equ        0x1A
MAXLOG1024B1    equ        0x21

;        Minimum argument to EXP1024

MINLOG1024EXP   equ        0x84                ; -37.9297794537 = log10(2**-126)
MINLOG1024B0    equ        0x97
MINLOG1024B1    equ        0xB8

;        Maximum representable number before overflow

MAXNUM24EXP     equ        0xFF                ; 6.80554349248E38 = (2**128) * (2 - 2**-15)
MAXNUM24B0      equ        0x7F
MAXNUM24B1      equ        0xFF

;        Minimum representable number before underflow

MINNUM24EXP     equ        0x01                ; 1.17549435082E-38 = (2**-126) * 1
MINNUM24B0      equ        0x00
MINNUM24B1      equ        0x00

;        Loss threshold for argument to SIN24 and COS24

LOSSTHR24EXP    equ        0x8B                ; 4096 = sqrt(2**24)
LOSSTHR24B0     equ        0x00
LOSSTHR24B1     equ        0x00

;*******************************************************************************************

;        32-BIT FLOATING POINT CONSTANTS

;        Machine precision

MACHEP32EXP     equ        0x67                ; 5.96046447754E-8 = 2**-24
MACHEP32B0      equ        0x00
MACHEP32B1      equ        0x00
MACHEP32B2      equ        0x00

;        Maximum argument to EXP32

MAXLOG32EXP     equ        0x85                ; 88.7228391117 = log(2**128)
MAXLOG32B0      equ        0x31
MAXLOG32B1      equ        0x72
MAXLOG32B2      equ        0x18

;        Minimum argument to EXP32

MINLOG32EXP     equ        0x85                ; -87.3365447506 = log(2**-126)
MINLOG32B0      equ        0xAE
MINLOG32B1      equ        0xAC
MINLOG32B2      equ        0x50

;        Maximum argument to EXP1032

MAXLOG1032EXP   equ        0x84                ; 38.531839445 = log10(2**128)
MAXLOG1032B0    equ        0x1A
MAXLOG1032B1    equ        0x20
MAXLOG1032B2    equ        0x9B

;        Minimum argument to EXP1032
```

```
MINLOG1032EXP   equ        0x84                ; -37.9297794537 = log10(2**-126)
MINLOG1032B0    equ        0x97
MINLOG1032B1    equ        0xB8
MINLOG1032B2    equ        0x18

;        Maximum representable number before overflow

MAXNUM32EXP     equ        0xFF                ; 6.80564774407E38 = (2**128) * (2 - 2**-23)
MAXNUM32B0      equ        0x7F
MAXNUM32B1      equ        0xFF
MAXNUM32B2      equ        0xFF

;        Minimum representable number before underflow

MINNUM32EXP     equ        0x01                ; 1.17549435082E-38 = (2**-126) * 1
MINNUM32B0      equ        0x00
MINNUM32B1      equ        0x00
MINNUM32B2      equ        0x00

;        Loss threshold for argument to SIN32 and COS32

LOSSTHR32EXP    equ        0x8B                ; 4096 = sqrt(2**24)
LOSSTHR32B0     equ        0x00
LOSSTHR32B1     equ        0x00
LOSSTHR32B2     equ        0x00


    ENDIF
```

B.3 Math17 Include File

```
;       RCS Header $Id: math17.inc 2.9 1997/01/31 02:23:41 F.J.Testa Exp $

;       $Revision: 2.9 $

;       MATH17 INCLUDE FILE
;
;       IMPORTANT NOTE: The math library routines can be used in a dedicated application on
;       an individual basis and memory allocation may be modified with the stipulation that
;       P type registers must remain so since P type specific instructions were used to
;       realize some performance improvements. This applies only to the PIC17.

;*******************************************************************************************

;       GENERAL MATH LIBRARY DEFINITIONS


;       general literal constants

;       define assembler constants

B0              equ     0
B1              equ     1
B2              equ     2
B3              equ     3
B4              equ     4
B5              equ     5
B6              equ     6
B7              equ     7

MSB             equ     7
LSB             equ     0


;     define commonly used bits

;     STATUS bit definitions

#define _C      ALUSTA,0
#define _DC     ALUSTA,1
#define _Z      ALUSTA,2
#define _OV     ALUSTA,3

;     general register variables

ACCB7           equ     0x18
ACCB6           equ     0x19
ACCB5           equ     0x1A
ACCB4           equ     0x1B
ACCB3           equ     0x1C
ACCB2           equ     0x1D
ACCB1           equ     0x1E
ACCB0           equ     0x1F
ACC             equ     0x1F    ; most significant byte of contiguous 8 byte accumulator

SIGN            equ     0x21    ; save location for sign in MSB

TEMPB3          equ     0x28
TEMPB2          equ     0x29
TEMPB1          equ     0x2A
TEMPB0          equ     0x2B
TEMP            equ     0x2B    ; temporary storage
```

```
;       binary operation arguments

AARGB7          equ     0x18
AARGB6          equ     0x19
AARGB5          equ     0x1A
AARGB4          equ     0x1B
AARGB3          equ     0x1C
AARGB2          equ     0x1D
AARGB1          equ     0x1E
AARGB0          equ     0x1F
AARG            equ     0x1F    ; most significant byte of argument A

BARGB3          equ     0x23
BARGB2          equ     0x24
BARGB1          equ     0x25
BARGB0          equ     0x26
BARG            equ     0x26    ; most significant byte of argument B


;       Note that AARG and ACC reference the same storage location

;*******************************************************************************************

;       FIXED POINT SPECIFIC DEFINITIONS

;       remainder storage

REMB3           equ     0x18
REMB2           equ     0x19
REMB1           equ     0x1A
REMB0           equ     0x1B    ; most significant byte of remainder

;*******************************************************************************************

;       FLOATING POINT SPECIFIC DEFINITIONS

;       literal constants

EXPBIAS         equ     D'127'

;       biased exponents

EXP             equ     0x20    ; 8 bit biased exponent
AEXP            equ     0x20    ; 8 bit biased exponent for argument A
BEXP            equ     0x27    ; 8 bit biased exponent for argument B

;       floating point library exception flags

FPFLAGS         equ     0x22    ; floating point library exception flags
IOV             equ     0       ; bit0 = integer overflow flag
FOV             equ     1       ; bit1 = floating point overflow flag
FUN             equ     2       ; bit2 = floating point underflow flag
FDZ             equ     3       ; bit3 = floating point divide by zero flag
NAN             equ     4       ; bit4 = not-a-number exception flag
DOM             equ     5       ; bit5 = domain error flag
RND             equ     6       ; bit6 = floating point rounding flag, 0 = truncation
                                ; 1 = unbiased rounding to nearest LSB
SAT             equ     7       ; bit7 = floating point saturate flag, 0 = terminate on
                                ; exception without saturation, 1 = terminate on
                                ; exception with saturation to appropriate value

;*******************************************************************************************
```

```
;       ELEMENTARY FUNCTION MEMORY

CEXP            equ     0x34
CARGB0          equ     0x33
CARGB1          equ     0x32
CARGB2          equ     0x31
CARGB3          equ     0x30

DEXP            equ     0x39
DARGB0          equ     0x38
DARGB1          equ     0x37
DARGB2          equ     0x36
DARGB3          equ     0x35

EEXP            equ     0x3E
EARGB0          equ     0x3D
EARGB1          equ     0x3C
EARGB2          equ     0x3B
EARGB3          equ     0x3A

FEXP            equ     0x43
FARGB0          equ     0x42
FARGB1          equ     0x41
FARGB2          equ     0x40
FARGB3          equ     0x3F

GEXP            equ     0x48
GARGB0          equ     0x47
GARGB1          equ     0x46
GARGB2          equ     0x45
GARGB3          equ     0x44

ZARGB0          equ     0x2F
ZARGB1          equ     0x2E
ZARGB2          equ     0x2D
ZARGB3          equ     0x2C

RANDB0          equ     0x4C
RANDB1          equ     0x4B
RANDB2          equ     0x4A
RANDB3          equ     0x49

;*******************************************************************************************

;       24-BIT FLOATING POINT CONSTANTS

;       Machine precision

MACHEP24EXP     equ     0x6F            ; 1.52587890625e-5 = 2**-16
MACHEP24B0      equ     0x00
MACHEP24B1      equ     0x00

;       Maximum argument to EXP24

MAXLOG24EXP     equ     0x85            ; 88.7228391117 = log(2**128)
MAXLOG24B0      equ     0x31
MAXLOG24B1      equ     0x72

;       Minimum argument to EXP24

MINLOG24EXP     equ     0x85            ; -87.3365447506 = log(2**-126)
MINLOG24B0      equ     0xAE
MINLOG24B1      equ     0xAC
```

```
;       Maximum argument to EXP1024

MAXLOG1024EXPe   qu      0x84            ; 38.531839445 = log10(2**128)
MAXLOG1024B0     equ     0x1A
MAXLOG1024B1     equ     0x21

;       Minimum argument to EXP1024

MINLOG1024EXP    equ     0x84            ; -37.9297794537 = log10(2**-126)
MINLOG1024B0     equ     0x97
MINLOG1024B1     equ     0xB8

;       Maximum representable number before overflow

MAXNUM24EXP      equ     0xFF            ; 6.80554349248E38 = (2**128) * (2 - 2**-15)
MAXNUM24B0       equ     0x7F
MAXNUM24B1       equ     0xFF

;       Minimum representable number before underflow

MINNUM24EXP      equ     0x01            ; 1.17549435082E-38 = (2**-126) * 1
MINNUM24B0       equ     0x00
MINNUM24B1       equ     0x00

;       Loss threshold for argument to SIN24 and COS24

LOSSTHR24EXP     equ     0x8A            ; LOSSTHR = sqrt(2**24)*PI/4
LOSSTHR24B0      equ     0x49
LOSSTHR24B1      equ     0x10

;*******************************************************************************************

;       32-BIT FLOATING POINT CONSTANTS

;       Machine precision

MACHEP32EXP      equ     0x67            ; 5.96046447754E-8 = 2**-24
MACHEP32B0       equ     0x00
MACHEP32B1       equ     0x00
MACHEP32B2       equ     0x00

;       Maximum argument to EXP32

MAXLOG32EXP      equ     0x85            ; 88.7228391117 = log(2**128)
MAXLOG32B0       equ     0x31
MAXLOG32B1       equ     0x72
MAXLOG32B2       equ     0x18

;       Minimum argument to EXP32

MINLOG32EXP      equ     0x85            ; -87.3365447506 = log(2**-126)
MINLOG32B0       equ     0xAE
MINLOG32B1       equ     0xAC
MINLOG32B2       equ     0x50

;       Maximum argument to EXP1032

MAXLOG1032EXP    equ     0x84            ; 38.531839445 = log10(2**128)
MAXLOG1032B0     equ     0x1A
MAXLOG1032B1     equ     0x20
MAXLOG1032B2     equ     0x9B
```

```
;       Minimum argument to EXP1032

MINLOG1032EXP    equ     0x84            ; -37.9297794537 = log10(2**-126)
MINLOG1032B0     equ     0x97
MINLOG1032B1     equ     0xB8
MINLOG1032B2     equ     0x18

;       Maximum representable number before overflow

MAXNUM32EXP      equ     0xFF            ; 6.80564774407E38 = (2**128) * (2 - 2**-23)
MAXNUM32B0       equ     0x7F
MAXNUM32B1       equ     0xFF
MAXNUM32B2       equ     0xFF

;       Minimum representable number before underflow

MINNUM32EXP      equ     0x01            ; 1.17549435082E-38 = (2**-126) * 1
MINNUM32B0       equ     0x00
MINNUM32B1       equ     0x00
MINNUM32B2       equ     0x00

;       Loss threshold for argument to SIN32 and COS32

LOSSTHR32EXP     equ     0x8A            ; LOSSTHR = sqrt(2**24)*PI/4
LOSSTHR32B0      equ     0x49
LOSSTHR32B1      equ     0x0F
LOSSTHR32B2      equ     0xDB
```

Please check the Microchip BBS for the latest version of the source code. For BBS access information, see Section 6, Microchip Bulletin Board Service information, page 6-3.

APPENDIX C: PIC16CXXX 24-BIT FLOATING POINT LIBRARY

```
;       RCS Header $Id: fp24.a16 2.7 1996/10/07 13:50:29 F.J.Testa Exp $

;       $Revision: 2.7 $

;       PIC16 24-BIT FLOATING POINT LIBRARY
;
;       Unary operations: both input and output are in AEXP,AARG
;
;       Binary operations: input in AEXP,AARG and BEXP,BARG with output in AEXP,AARG
;
;       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
;       for an error condition specified in FPFLAGS.
;
;       All timings are worst case cycle counts
;
;         Routine               Function
;
;       FLO1624 16 bit integer to 24 bit floating point conversion
;       FLO24
;
;               Timing:            RND
;                                0       1
;
;                       0       83      83
;                  SAT
;                       1       88      88
;
;       NRM2424         24 bit normalization of unnormalized 24 bit floating point numbers
;       NRM24
;
;               Timing:            RND
;                                0       1
;
;                       0       72      72
;                  SAT
;                       1       77      77
;
;       INT2416         24 bit floating point to 16 bit integer conversion
;       INT24
;
;               Timing:            RND
;                                0       1
;
;                       0       83      89
;                  SAT
;                       1       83      92
;
;       FLO2424         24 bit integer to 24 bit floating point conversion
;
;               Timing:            RND
;                                0       1
;
;                       0       108     117
;                  SAT
;                       1       108     123
;
```

```
;       NRM3224          32 bit normalization of unnormalized 24 bit floating point numbers
;
;               Timing:            RND
;                                0       1
;
;                       0       94      103
;                  SAT
;                       1       94      109
;
;       INT2424         24 bit floating point to 24 bit integer conversion
;
;               Timing:            RND
;                                0        1
;
;                       0       105      113
;                  SAT
;                       1       105      115
;
;       FPA24           24 bit floating point add
;
;               Timing:            RND
;                                0       1
;
;                       0       197     208
;                  SAT
;                       1       197     213
;
;       FPS24           24 bit floating point subtract
;
;               Timing:            RND
;                                0       1
;
;                       0       199     240
;                  SAT
;                       1       199     215
;
;       FPM24           24 bit floating point multiply
;
;               Timing:            RND
;                                0       1
;
;                       0       298     309
;                  SAT
;                       1       298     313
;
;       FPD24           24 bit floating point divide
;
;               Timing:            RND
;                                0       1
;
;                       0       472     494
;                  SAT
;                       1       472     498
;


;*******************************************************************************************
;*******************************************************************************************
;
;       24-bit floating point representation
;
;       EXPONENT        8 bit biased exponent
;                       It is important to note that the use of biased exponents produces
;                       a unique representation of a floating point 0, given by
;                       EXP = HIGHBYTE = LOWBYTE = 0x00, with 0 being the only
;                       number with EXP = 0.
```

```
;
;       HIGHBYTE        8 bit most significant byte of fraction in sign-magnitude representation,
;                       with SIGN = MSB, implicit MSB = 1 and radix point to the right of MSB
;
;       LOWBYTE         8 bit least significant byte of sign-magnitude fraction
;
;       EXPONENT        HIGHBYTE        LOWBYTE
;
;       xxxxxxxx        S.xxxxxxx       xxxxxxxx
;
;                        |
;                      RADIX
;                      POINT
;

;*******************************************************************************************
;*******************************************************************************************

;       Integer to float conversion

;       Input:  16 bit 2's complement integer right justified in AARGB0, AARGB1

;       Use:    CALL    FLO1624    or    CALL    FLO24

;       Output: 24 bit floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     11+72 = 83 clks         SAT = 0
;                       11+77 = 88 clks         SAT = 1

;       Min Timing:     7+14 = 21 clks          AARG = 0
;                       7+18 = 25 clks

;       PM: 11+26 = 37                          DM: 6

;-------------------------------------------------------------------------------------------

FLO1624

FLO24           MOVLW           D'15'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                MOVF            AARGB0,W
                MOVWF           SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM2424
                COMF            AARGB1,F                ; if < 0, negate and set MSB in SIGN
                COMF            AARGB0,F
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

;*******************************************************************************************

;       Normalization routine

;       Input:  24 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               with sign in SIGN, MSB and other bits zero.

;       Use:    CALL    NRM2424    or    CALL    NRM24

;       Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  NORMALIZE( AARG )
```

```
;       Max Timing:     10+6+7*7+7 = 72 clks            SAT = 0
;                       10+6+7*7+1+11 = 77 clks         SAT = 1

;       Min Timing:     14 clks                         AARG = 0
;                       5+9+4 = 18 clks

;       PM: 26                                          DM: 6

;-----------------------------------------------------------------------------------------

NRM2424
NRM24
                CLRF            TEMP                ; clear exponent decrement
                MOVF            AARGB0,W            ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM2424
                MOVF            AARGB1,W            ; if so, shift 8 bits by move
                MOVWF           AARGB0
                BTFSC           _Z                  ; if highbyte=0, result=0
                GOTO            RES024
                CLRF            AARGB1
                BSF             TEMP,3

NORM2424        MOVF            TEMP,W
                SUBWF           EXP,F
                BTFSS           _Z
                BTFSS           _C
                GOTO            SETFUN24

                BCF             _C                   ; clear carry bit

NORM2424A       BTFSC           AARGB0,MSB           ; if MSB=1, normalization done
                GOTO            FIXSIGN24
                RLF             AARGB1,F             ; otherwise, shift left and
                RLF             AARGB0,F             ; decrement EXP
                DECFSZ          EXP,F
                GOTO            NORM2424A

                GOTO            SETFUN24             ; underflow if EXP=0

FIXSIGN24       BTFSS           SIGN,MSB
                BCF             AARGB0,MSB           ; clear explicit MSB if positive
                RETLW           0

RES024          CLRF            AARGB0               ; result equals zero
                CLRF            AARGB1
                CLRF            AARGB2               ; clear extended byte
                CLRF            EXP
                RETLW           0


;*****************************************************************************************
;*****************************************************************************************

;       Integer to float conversion

;       Input:  24 bit 2’s complement integer right justified in AARGB0, AARGB1, AARGB2

;       Use:    CALL    FLO2424

;       Output: 24 bit floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     14+94 = 108 clks                RND = 0
;                       14+103 = 117 clks               RND = 1, SAT = 0
;                       14+109 = 123 clks               RND = 1, SAT = 1
```

```
;       Min Timing:     6+28 = 34 clks                  AARG = 0
;                       6+22 = 28 clks

;       PM: 14+51 = 65                                  DM: 7

;-----------------------------------------------------------------------------------------

FLO2424         MOVLW           D'23'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                CLRF            SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM3224
                COMF            AARGB2,F                ; if < 0, negate and set MSB in SIGN
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BSF             SIGN,MSB

;*****************************************************************************************

;       Normalization routine

;       Input:  32 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2, with sign in SIGN,MSB

;       Use:    CALL    NRM3224

;       Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     21+6+7*8+7+4 = 94 clks  RND = 0
;                       21+6+7*8+20+4 = 103 clks        RND = 1, SAT = 0
;                       21+6+7*8+19+11 = 109 clks       RND = 1, SAT = 1

;       Min Timing:     22+6 = 28 clks                  AARG = 0
;                       5+9+4+4 = 22 clks

;       PM: 51                                          DM: 7

;-----------------------------------------------------------------------------------------

NRM3224         CLRF            TEMP                    ; clear exponent decrement
                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM3224
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                MOVF            AARGB2,W
                MOVWF           AARGB1
                CLRF            AARGB2
                BSF             TEMP,3                  ; increase decrement by 8

                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM3224
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                CLRF            AARGB1
                BCF             TEMP,3                  ; increase decrement by 8
                BSF             TEMP,4
```

```
                MOVF            AARGB0,W                ; if highbyte=0, result=0
                BTFSC           _Z
                GOTO            RES024

NORM3224        MOVF            TEMP,W
                SUBWF           EXP,F
                BTFSS           _Z
                BTFSS           _C
                GOTO            SETFUN24

                BCF             _C                      ; clear carry bit

NORM3224A       BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            NRMRND3224
                RLF             AARGB2,F                ; otherwise, shift left and
                RLF             AARGB1,F                ; decrement EXP
                RLF             AARGB0,F
                DECFSZ          EXP,F
                GOTO            NORM3224A
                GOTO            SETFUN24                ; underflow if EXP=0

NRMRND3224      BTFSC           FPFLAGS,RND
                BTFSS           AARGB1,LSB
                GOTO            FIXSIGN24
                BTFSS           AARGB2,MSB              ; round if next bit is set
                GOTO            FIXSIGN24
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

                BTFSS           _Z                      ; has rounding caused carryout?
                GOTO            FIXSIGN24
                RRF             AARGB0,F                ; if so, right shift
                RRF             AARGB1,F
                INCF            EXP,F
                BTFSC           _Z                      ; check for overflow
                GOTO            SETFOV24
                GOTO            FIXSIGN24

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1

;       Use:    CALL    INT2416         or      CALL    INT24

;       Output: 16 bit 2's complement integer right justified in AARGB0, AARGB1

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     29+6*6+5+13 = 83 clks           RND = 0
;                       29+6*6+5+19 = 89 clks           RND = 1, SAT = 0
;                       29+6*6+5+22 = 92 clks           RND = 1, SAT = 1

;       Min Timing:     18+5+7 = 30 clks

;       PM: 63                                          DM: 6

;-------------------------------------------------------------------------------------------

INT2416
INT24
                MOVF            EXP,W                   ; test for zero argument
```

```
                BTFSC           _Z
                RETLW           0x00

                MOVF            AARGB0,W                ; save sign in SIGN
                MOVWF           SIGN
                BSF             AARGB0,MSB              ; make MSB explicit

                MOVLW           EXPBIAS+D'15'           ; remove bias from EXP
                SUBWF           EXP,F
                BTFSS           EXP,MSB
                GOTO            SETIOV16
                COMF            EXP,F
                INCF            EXP,F

                MOVLW           8                       ; do byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT2416
                MOVWF           EXP
                RLF             AARGB1,F                ; rotate next bit for rounding
                MOVF            AARGB0,W
                MOVWF           AARGB1
                CLRF            AARGB0

                MOVLW           8                       ; do byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT2416
                MOVWF           EXP
                RLF             AARGB1,F                ; rotate next bit for rounding
                CLRF            AARGB1
                MOVF            EXP,W
                BTFSS           _Z
                BCF             _C
                GOTO            SHIFT2416OK

TSHIFT2416      MOVF            EXP,W                   ; shift completed if EXP = 0
                BTFSC           _Z
                GOTO            SHIFT2416OK

SHIFT2416       BCF             _C
                RRF             AARGB0,F                ; right shift by EXP
                RRF             AARGB1,F
                DECFSZ          EXP,F
                GOTO            SHIFT2416

SHIFT2416OK     BTFSC           FPFLAGS,RND
                BTFSS           AARGB1,LSB
                GOTO            INT2416OK
                BTFSS           _C                      ; round if next bit is set
                GOTO            INT2416OK
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

                BTFSC           AARGB0,MSB              ; test for overflow
                GOTO            SETIOV16

INT2416OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                RETLW           0
```

```
SETIOV16        BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0                  ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 16 bit integer
                MOVLW           0xFF
                MOVWF           AARGB0                  ; SIGN = 0, 0x 7F FF
                MOVWF           AARGB1                  ; SIGN = 1, 0x 80 00
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG


;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1

;       Use:    CALL    INT2424

;       Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     41+6*7+6+16 = 105 clks                RND = 0
;                       41+6*7+6+24 = 113 clks                RND = 1, SAT = 0
;                       41+6*7+6+26 = 115 clks                RND = 1, SAT = 1

;       Min Timing:     5 clks

;       PM: 82                                                DM: 6

;-------------------------------------------------------------------------------------------

INT2424
                CLRF            AARGB2
                MOVF            EXP,W                   ; test for zero argument
                BTFSC           _Z
                RETLW           0x00

                MOVF            AARGB0,W                ; save sign in SIGN
                MOVWF           SIGN
                BSF             AARGB0,MSB              ; make MSB explicit

                MOVLW           EXPBIAS+D'23'           ; remove bias from EXP
                SUBWF           EXP,F
                BTFSS           EXP,MSB
                GOTO            SETIOV24
                COMF            EXP,F
                INCF            EXP,F

                MOVLW           8                       ; do byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT2424
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                MOVF            AARGB1,W
                MOVWF           AARGB2
                MOVF            AARGB0,W
                MOVWF           AARGB1
```

```
                CLRF            AARGB0

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT2424
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                MOVF            AARGB1,W
                MOVWF           AARGB2
                CLRF            AARGB1

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT2424
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                CLRF            AARGB2
                MOVF            EXP,W
                BTFSS           _Z
                BCF             _C
                GOTO            SHIFT2424OK

TSHIFT2424      MOVF            EXP,W                   ; shift completed if EXP = 0
                BTFSC           _Z
                GOTO            SHIFT2424OK

SHIFT2424       BCF             _C
                RRF             AARGB0,F                ; right shift by EXP
                RRF             AARGB1,F
                RRF             AARGB2,F
                DECFSZ          EXP,F
                GOTO            SHIFT2424

SHIFT2424OK     BTFSC           FPFLAGS,RND
                BTFSS           AARGB2,LSB
                GOTO            INT2424OK
                BTFSS           _C
                GOTO            INT2424OK
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BTFSC           AARGB0,MSB              ; test for overflow
                GOTO            SETIOV24

INT2424OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB0,F
                COMF            AARGB1,F
                COMF            AARGB2,F
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                RETLW           0

IRES024         CLRF            AARGB0                  ; integer result equals zero
                CLRF            AARGB1
                CLRF            AARGB2
                RETLW           0

SETIOV24        BSF             FPFLAGS,IOV             ; set integer overflow flag
```

```
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0                  ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 24 bit integer
                MOVLW           0xFF
                MOVWF           AARGB0                  ; SIGN = 0, 0x 7F FF FF
                MOVWF           AARGB1                  ; SIGN = 1, 0x 80 00 00
                MOVWF           AARGB2
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG


;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Multiply

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;       Use:    CALL    FPM24

;       Output: 24 bit floating point product in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG * BARG

;       Max Timing:     25+15*16+15+18 = 298 clks       RND = 0
;                       25+15*16+15+29 = 309 clks       RND = 1, SAT = 0
;                       25+15*16+15+33 = 313 clks       RND = 1, SAT = 1

;       Min Timing:     6+5 = 11 clks                   AARG * BARG = 0
;                       24+15*11+14+15 = 218 clks

;       PM: 80                                          DM: 11

;-------------------------------------------------------------------------------------------

FPM24           MOVF            AEXP,W                  ; test for zero arguments
                BTFSS           _Z
                MOVF            BEXP,W
                BTFSC           _Z
                GOTO            RES024

M24BNE0         MOVF            AARGB0,W
                XORWF           BARGB0,W
                MOVWF           SIGN                    ; save sign in SIGN

                MOVF            BEXP,W
                ADDWF           EXP,F
                MOVLW           EXPBIAS-1
                BTFSS           _C
                GOTO            MTUN24

                SUBWF           EXP,F
                BTFSC           _C
                GOTO            SETFOV24                ; set multiply overflow flag
                GOTO            MOK24

MTUN24          SUBWF           EXP,F
                BTFSS           _C
                GOTO            SETFUN24

MOK24           MOVF            AARGB0,W
```

```
            MOVF        AARGB0,W
            MOVWF       AARGB2              ; move result to AARG
            MOVF        AARGB1,W
            MOVWF       AARGB3
            BSF         AARGB2,MSB          ; make argument MSB's explicit
            BSF         BARGB0,MSB
            BCF         _C
            CLRF        AARGB0              ; clear initial partial product
            CLRF        AARGB1
            MOVLW       D'16'
            MOVWF       TEMP                ; initialize counter

MLOOP24     BTFSS       AARGB3,LSB          ; test next bit
            GOTO        MNOADD24

MADD24      MOVF        BARGB1,W
            ADDWF       AARGB1,F
            MOVF        BARGB0,W
            BTFSC       _C
            INCFSZ      BARGB0,W
            ADDWF       AARGB0,F

MNOADD24    RRF         AARGB0,F
            RRF         AARGB1,F
            RRF         AARGB2,F
            RRF         AARGB3,F
            BCF         _C
            DECFSZ      TEMP,F
            GOTO        MLOOP24

            BTFSC       AARGB0,MSB          ; check for postnormalization
            GOTO        MROUND24
            RLF         AARGB2,F
            RLF         AARGB1,F
            RLF         AARGB0,F
            DECF        EXP,F

MROUND24    BTFSC       FPFLAGS,RND
            BTFSS       AARGB1,LSB
            GOTO        MUL24OK
            BTFSS       AARGB2,MSB          ; round if next bit is set
            GOTO        MUL24OK
            INCF        AARGB1,F
            BTFSC       _Z
            INCF        AARGB0,F

            BTFSS       _Z                  ; has rounding caused carryout?
            GOTO        MUL24OK
            RRF         AARGB0,F            ; if so, right shift
            RRF         AARGB1,F
            INCF        EXP,F
            BTFSC       _Z                  ; check for overflow
            GOTO        SETFOV24

MUL24OK     BTFSS       SIGN,MSB
            BCF         AARGB0,MSB          ; clear explicit MSB if positive
            RETLW       0

SETFOV24    BSF         FPFLAGS,FOV         ; set floating point underflag
            BTFSS       FPFLAGS,SAT         ; test for saturation
            RETLW       0xFF                ; return error code in WREG

            MOVLW       0xFF
            MOVWF       AEXP                ; saturate to largest floating
            MOVWF       AARGB0              ; point number = 0x FF 7F FF
            MOVWF       AARGB1              ; modulo the appropriate sign bit
```

```
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Divide

;       Input:  24 bit floating point dividend in AEXP, AARGB0, AARGB1
;               24 bit floating point divisor in BEXP, BARGB0, BARGB1

;       Use:    CALL    FPD24

;       Output: 24 bit floating point quotient in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG / BARG

;       Max Timing:     32+13+15*26+25+12 = 472 clks    RND = 0
;                       32+13+15*26+25+34 = 494 clks    RND = 1, SAT = 0
;                       32+13+15*26+25+38 = 498 clks    RND = 1, SAT = 1

;       Min Timing:     7+5 = 12 clks

;       PM: 120                                         DM: 11

;-----------------------------------------------------------------------------------------

FPD24           MOVF            BEXP,W                  ; test for divide by zero
                BTFSC           _Z
                GOTO            SETFDZ24

                MOVF            AEXP,W
                BTFSC           _Z
                GOTO            RES024

D24BNE0         MOVF            AARGB0,W
                XORWF           BARGB0,W
                MOVWF           SIGN                    ; save sign in SIGN
                BSF             AARGB0,MSB              ; make argument MSB’s explicit
                BSF             BARGB0,MSB

TALIGN24        CLRF            TEMP                    ; clear align increment
                MOVF            AARGB0,W
                MOVWF           AARGB2                  ; test for alignment
                MOVF            AARGB1,W
                MOVWF           AARGB3

                MOVF            BARGB1,W
                SUBWF           AARGB3, f
                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W
                SUBWF           AARGB2, f

                CLRF            AARGB2
                CLRF            AARGB3

                BTFSS           _C
                GOTO            DALIGN24OK

                BCF             _C                      ; align if necessary
                RRF             AARGB0,F
                RRF             AARGB1,F
                RRF             AARGB2,F
                MOVLW           0x01
```

```
                MOVWF           TEMP                    ; save align increment

DALIGN24OK      MOVF            BEXP,W                  ; compare AEXP and BEXP
                SUBWF           EXP,F
                BTFSS           _C
                GOTO            ALTB24

AGEB24          MOVLW           EXPBIAS-1
                ADDWF           TEMP,W
                ADDWF           EXP,F
                BTFSC           _C
                GOTO            SETFOV24
                GOTO            DARGOK24                ; set overflow flag

ALTB24          MOVLW           EXPBIAS-1
                ADDWF           TEMP,W
                ADDWF           EXP,F
                BTFSS           _C
                GOTO            SETFUN24                ; set underflow flag

DARGOK24        MOVLW           D'16'                   ; initialize counter
                MOVWF           TEMPB1

DLOOP24         RLF             AARGB3,F                ; left shift
                RLF             AARGB2,F
                RLF             AARGB1,F
                RLF             AARGB0,F
                RLF             TEMP,F

                MOVF            BARGB1,W                ; subtract
                SUBWF           AARGB1,F
                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W
                SUBWF           AARGB0,F

                RLF             BARGB0,W
                IORWF           TEMP,F

                BTFSS           TEMP,LSB                ; test for restore
                GOTO            DREST24

                BSF             AARGB3,LSB
                GOTO            DOK24

DREST24         MOVF            BARGB1,W                ; restore if necessary
                ADDWF           AARGB1,F
                MOVF            BARGB0,W
                BTFSC           _C
                INCF            BARGB0,W
                ADDWF           AARGB0,F
                BCF             AARGB3,LSB

DOK24           DECFSZ          TEMPB1,F
                GOTO            DLOOP24

DROUND24        BTFSC           FPFLAGS,RND
                BTFSS           AARGB3,LSB
                GOTO            DIV24OK
                BCF             _C
                RLF             AARGB1,F                ; compute next significant bit
                RLF             AARGB0,F                ; for rounding
                RLF             TEMP,F

                MOVF            BARGB1,W                ; subtract
                SUBWF           AARGB1,F
```

```
                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W
                SUBWF           AARGB0,F

                RLF             BARGB0,W
                IORWF           TEMP,W
                ANDLW           0x01

                ADDWF           AARGB3,F
                BTFSC           _C
                INCF            AARGB2,F

                BTFSS           _Z                      ; test if rounding caused carryout
                GOTO            DIV24OK
                RRF             AARGB2,F
                RRF             AARGB3,F
                INCF            EXP,F
                BTFSC           _Z                      ; test for overflow
                GOTO            SETFOV24

DIV24OK         BTFSS           SIGN,MSB
                BCF             AARGB2,MSB              ; clear explicit MSB if positive

                MOVF            AARGB2,W
                MOVWF           AARGB0                  ; move result to AARG
                MOVF            AARGB3,W
                MOVWF           AARGB1

                RETLW           0

SETFUN24        BSF             FPFLAGS,FUN             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                MOVLW           0x01                    ; saturate to smallest floating
                MOVWF           AEXP                    ; point number = 0x 01 00 00
                CLRF            AARGB0                  ; modulo the appropriate sign bit
                CLRF            AARGB1
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

SETFDZ24        BSF             FPFLAGS,FDZ             ; set divide by zero flag
                RETLW           0xFF

;*****************************************************************************************
;*****************************************************************************************

;       Floating Point Subtract

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;       Use:    CALL FPS24

;       Output: 24 bit floating point sum in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     2+197 = 199 clks                RND = 0
;                       2+208 = 210 clks                RND = 1, SAT = 0
;                       2+213 = 215 clks                RND = 1, SAT = 1

;       Min Timing:     2+12 = 14 clks
```

```
;       PM: 2+112 = 114                                 DM: 11

;-----------------------------------------------------------------------------------------

FPS24           MOVLW           0x80
                XORWF           BARGB0,F

;*****************************************************************************************

;       Floating Point Add

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;       Use:    CALL FPA24

;       Output: 24 bit floating point sum in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     25+28+6*6+5+31+72 = 197 clks    RND = 0
;                       25+28+6*6+5+42+72 = 208 clks    RND = 1, SAT = 0
;                       25+28+6*6+5+42+77 = 213 clks    RND = 1, SAT = 1

;       Min Timing:     8+4 = 12 clks

;       PM: 112                                         DM: 11

;-----------------------------------------------------------------------------------------

FPA24           MOVF            AARGB0,W                ; exclusive or of signs in TEMP
                XORWF           BARGB0,W
                MOVWF           TEMP

                CLRF            AARGB2                  ; clear extended byte
                CLRF            BARGB2

                MOVF            AEXP,W                  ; use AARG if AEXP >= BEXP
                SUBWF           BEXP,W
                BTFSS           _C
                GOTO            USEA24

                MOVF            BEXP,W                  ; use BARG if AEXP < BEXP
                MOVWF           AARGB4                  ; therefore, swap AARG and BARG
                MOVF            AEXP,W
                MOVWF           BEXP
                MOVF            AARGB4,W
                MOVWF           AEXP

                MOVF            BARGB0,W
                MOVWF           AARGB4
                MOVF            AARGB0,W
                MOVWF           BARGB0
                MOVF            AARGB4,W
                MOVWF           AARGB0

                MOVF            BARGB1,W
                MOVWF           AARGB4
                MOVF            AARGB1,W
                MOVWF           BARGB1
                MOVF            AARGB4,W
                MOVWF           AARGB1

USEA24          MOVF            BEXP,W                  ; return AARG if BARG = 0
                BTFSC           _Z
                RETLW           0x00
```

```
                MOVF            AARGB0,W
                MOVWF           SIGN                    ; save sign in SIGN
                BSF             AARGB0,MSB              ; make MSB's explicit
                BSF             BARGB0,MSB

                MOVF            BEXP,W                  ; compute shift count in BEXP
                SUBWF           AEXP,W
                MOVWF           BEXP
                BTFSC           _Z
                GOTO            ALIGNED24

                MOVLW           8
                SUBWF           BEXP,W
                BTFSS           _C                      ; if BEXP >= 8, do byte shift
                GOTO            ALIGNB24
                MOVWF           BEXP
                MOVF            BARGB1,W                ; keep for postnormalization
                MOVWF           BARGB2
                MOVF            BARGB0,W
                MOVWF           BARGB1
                CLRF            BARGB0

                MOVLW           8
                SUBWF           BEXP,W
                BTFSS           _C                      ; if BEXP >= 8, BARG = 0 relative to AARG
                GOTO            ALIGNB24
                MOVF            SIGN,W
                MOVWF           AARGB0
                RETLW           0x00

ALIGNB24        MOVF            BEXP,W                  ; already aligned if BEXP = 0
                BTFSC           _Z
                GOTO            ALIGNED24

ALOOPB24        BCF             _C                      ; right shift by BEXP
                RRF             BARGB0,F
                RRF             BARGB1,F
                RRF             BARGB2,F
                DECFSZ          BEXP,F
                GOTO            ALOOPB24

ALIGNED24       BTFSS           TEMP,MSB                ; negate if signs opposite
                GOTO            AOK24
                COMF            BARGB2,F
                COMF            BARGB1,F
                COMF            BARGB0,F
                INCF            BARGB2,F
                BTFSC           _Z
                INCF            BARGB1,F
                BTFSC           _Z
                INCF            BARGB0,F

AOK24
                MOVF            BARGB2,W
                ADDWF           AARGB2,F
                MOVF            BARGB1,W
                BTFSC           _C
                INCFSZ          BARGB1,W
                ADDWF           AARGB1,F
                MOVF            BARGB0,W
                BTFSC           _C
                INCFSZ          BARGB0,W
                ADDWF           AARGB0,F

                BTFSC           TEMP,MSB
```

```
            GOTO        ACOMP24
            BTFSS       _C
            GOTO        NRMRND3224

            RRF         AARGB0,F            ; shift right and increment EXP
            RRF         AARGB1,F
            RRF         AARGB2,F
            INCFSZ      AEXP,F
            GOTO        NRMRND3224
            GOTO        SETFOV24

ACOMP24     BTFSC       _C
            GOTO        NRM3224             ; normalize and fix sign

            COMF        AARGB2,F
            COMF        AARGB1,F            ; negate, toggle sign bit and
            COMF        AARGB0,F            ; then normalize
            INCF        AARGB2,F
            BTFSC       _Z
            INCF        AARGB1,F
            BTFSC       _Z
            INCF        AARGB0,F

            MOVLW       0x80
            XORWF       SIGN,F
            GOTO        NRM24
```

NOTES:

Please check the Microchip BBS for the latest version of the source code. For BBS access information, see Section 6, Microchip Bulletin Board Service information, page 6-3.

APPENDIX D: PIC17CXXX 24-BIT FLOATING POINT LIBRARY

```
;       RCS Header $Id: fp24.a17 2.8 1996/12/21 20:59:37 F.J.Testa Exp $

;       $Revision: 2.8 $

;       PIC17 24-BIT FLOATING POINT LIBRARY
;
;       Unary operations: both input and output are in AEXP,AARG
;
;       Binary operations: input in AEXP,AARG and BEXP,BARG with output in AEXP,AARG
;
;       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
;       for an error condition specified in FPFLAGS.
;
;       Max timings are worst case cycle counts, while Min timings are non-exception
;       best case cycle counts.
;
;         Routine               Function
;
;       FLO1624         16 bit integer to 24 bit floating point conversion
;       FLO24
;
;       Max Timing:                RND
;                             0       1
;
;                       0       48      48
;                  SAT
;                       1       55      55
;
;       NRM2424  24 bit normalization of unnormalized 24 bit floating point numbers
;       NRM24
;
;       Max Timing:                RND
;                             0       1
;
;                       0       39      39
;                  SAT
;                       1       46      46
;
;
;       INT2416         24 bit floating point to 16 bit integer conversion
;       INT24
;
;       Max Timing:                RND
;                             0       1
;
;                       0       58      66
;                  SAT
;                       1       58      70
;
;       FLO2424         24 bit integer to 24 bit floating point conversion
;
;       Max Timing:                RND
;                             0       1
;
;                       0       64      77
;                  SAT
;                       1       64      83
;
```

```
;       NRM3224 24 bit normalization of unnormalized 32 bit floating point numbers
;
;       Max Timing:                RND
;                           0       1
;
;                       0       52      65
;                  SAT
;                       1       52      71
;
;
;       INT2424         24 bit floating point to 24 bit integer conversion
;
;       Max Timing:                RND
;                           0       1
;
;                       0       71      79
;                  SAT
;                       1       71      82
;
;       FPA24           24 bit floating point add
;
;       Max Timing:                RND
;                           0       1
;
;                       0       133     146
;                  SAT
;                       1       133     152
;
;       FPS24           24 bit floating point subtract
;
;       Max Timing:                RND
;                           0       1
;
;                       0       134     147
;                  SAT
;                       1       134     153
;
;       FPM24           24 bit floating point multiply
;
;       Max Timing:                RND
;                           0       1
;
;                       0       60      72
;                  SAT
;                       1       60      79
;
;       FPD24           24 bit floating point divide
;
;       Max Timing:                RND
;                           0       1
;
;                       0       176     185
;                  SAT
;                       1       176     192
;
```

```
;*****************************************************************************************
;*****************************************************************************************

;       24 bit floating point representation

;       EXPONENT        8 bit biased exponent

;                       It is important to note that the use of biased exponents produces
;                       a unique representation of a floating point 0, given by
;                       EXP = HIGHBYTE = LOWBYTE = 0x00, with 0 being the only
;                       number with EXP = 0.

;       HIGHBYTE        8 bit most significant byte of fraction in sign-magnitude representation,
;                       with SIGN = MSB, implicit MSB = 1 and radix point to the right of MSB

;       LOWBYTE         8 bit least significant byte of sign-magnitude fraction

;       EXPONENT        HIGHBYTE        LOWBYTE

;       xxxxxxxx        S.xxxxxxx       xxxxxxxx

;                        |
;                      RADIX
;                      POINT


;*****************************************************************************************
;*****************************************************************************************

;       Integer to float conversion

;       Input:  16 bit 2's complement integer right justified in AARGB0, AARGB1

;       Use:    CALL    FLO1624
;               CALL    FLO24

;       Output: 24 bit floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     9+39 = 48 clks          SAT = 0
;                       9+45 = 54 clks          SAT = 1

;       Min Timing:     6+15 = 21 clks          AARG = 0
;                       6+20 = 26 clks

;       PM: 9+68 = 77                           DM: 6

;-----------------------------------------------------------------------------------------

FLO1624

FLO24           MOVLW           D'15'+EXPBIAS   ; initialize exponent and add bias
                MOVWF           EXP
                MOVPF           AARGB0,SIGN     ; save sign in SIGN
                BTFSS           AARGB0,MSB      ; test sign
                GOTO            NRM24
                COMF            AARGB1,F        ; if < 0, negate, set MSB in SIGN
                COMF            AARGB0,F
                INFSNZ          AARGB1,F
                INCF            AARGB0,F
```

```
;*****************************************************************************************

;       Normalization routine

;       Input:  24 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               with sign in SIGN,MSB.

;       Use:    CALL    NRM2424
;               CALL    NRM24

;       Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     3+12+16+8 = 39 clks             SAT = 0
;                       3+12+16+15 = 46 clks            SAT = 1

;       Min Timing:     10+5 = 15 clks                  AARG = 0
;                       3+5+4+8 = 20 clks

;       PM: 68                                          DM: 6

;-----------------------------------------------------------------------------------------

NRM2424
NRM24
                CLRF            TEMP,W                  ; clear exponent decrement
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM2424A

TNIB2424        MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM2424
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F
                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1

                BSF             TEMP,2                  ; increase decrement by 4

NORM2424        BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN2424
                RLCF            AARGB1,F                ; otherwise, shift left and
                RLCF            AARGB0,F                ; increment decrement
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN2424
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN2424             ; 3 left shifts are required
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F

TNORMUN2424     MOVFP           TEMP,WREG               ; if EXP <= decrement in TEMP,
                CPFSGT          EXP                     ; floating point underflow has
                GOTO            SETFUN24                ; occured
                SUBWF           EXP,F                   ; otherwise, compute EXP
```

```
FIXSIGN24       BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive
                RETLW           0

NRM2424A        MOVFP           AARGB1,AARGB0           ; if so, shift 8 bits by move
                CLRF            AARGB1,F
                BSF             TEMP,3                  ; increase decrement by 8
                CPFSGT          AARGB0                  ; if highbyte=0, result=0
                GOTO            RES024

                MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM2424A
                SWAPF           AARGB0,F                ; if so, shift 4 bits

                BSF             TEMP,2                  ; increase decrement by 4

NORM2424A       BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN2424
                RLCF            AARGB0,F                ; otherwise, shift left and
                INCF            TEMP,F                  ; increment decrement
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN2424
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN2424             ; 3 left shifts are required
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN2424

RES024          CLRF            AARGB0,F                ; result equals zero
                CLRF            AARGB1,F
                CLRF            AARGB2,F                ; clear extended byte
                CLRF            EXP,F
                RETLW           0

;*******************************************************************************************
;*******************************************************************************************

;       Integer to float conversion

;       Input:  24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Use:    CALL    FLO2424

;       Output: 24 bit floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     12+52 = 64 clks                 RND = 0
;                       12+65 = 77 clks                 RND = 1, SAT = 0
;                       12+71 = 83 clks                 RND = 1, SAT = 1

;       Min Timing:     6+24 = 30 clks                  AARG = 0
;                       6+24 = 30 clks

;       PM: 12+121 = 133                                DM: 7

;-------------------------------------------------------------------------------------------
```

```
FLO2424         MOVLW           D'23'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM3224
                CLRF            WREG,F                  ; if < 0, negate, set MSB in SIGN
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

;*******************************************************************************************

;       Normalization routine

;       Input:  32 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2 with sign in SIGN,MSB.

;       Use:    CALL    NRM3224

;       Output: 24 bit normalized floating point number in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     21+19+12 = 52 clks              RND = 0
;                       21+19+25 = 65 clks              RND = 1, SAT = 0
;                       21+19+31 = 71 clks              RND = 1, SAT = 1

;       Min Timing:     4+7+7+5 = 24 clks               AARG = 0
;                       3+5+4+8+4 = 24 clks

;       PM: 122                                         DM: 7

;-------------------------------------------------------------------------------------------

NRM3224         CLRF            TEMP,W                  ; clear exponent decrement
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM3224A

TNIB3224        MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM3224
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1
                SWAPF           AARGB2,W
                ANDLW           0x0F
                ADDWF           AARGB1,F

                SWAPF           AARGB2,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB2

                BSF             TEMP,2                  ; increase decrement by 4

NORM3224        BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
```

```
                GOTO            TNORMUN3224
                RLCF            AARGB2,F                ; otherwise, shift left and
                RLCF            AARGB1,F                ; increment decrement
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3224
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN3224             ; 3 left shifts are required
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F

TNORMUN3224     MOVFP           TEMP,WREG               ; if EXP <= decrement in TEMP,
                CPFSGT          EXP                     ; floating point underflow has
                GOTO            SETFUN24                ; occured
                SUBWF           EXP,F                   ; otherwise, compute EXP

NRMRND3224
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB2,MSB              ; is NSB > 0x80?
                GOTO            FIXSIGN24
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB2                  ; if NSB = 0x80, select even
                RRCF            AARGB1,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                BTFSS           _C                      ; has rounding caused carryout?
                GOTO            FIXSIGN24
                RRCF            AARGB0,F                ; if so, right shift
                RRCF            AARGB1,F
                INFSNZ          EXP,F                   ; test for floating point overflow
                GOTO            SETFOV24
                GOTO            FIXSIGN24

NRM3224A        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                MOVFP           AARGB2,AARGB1
                CLRF            AARGB2,W
                BSF             TEMP,3                  ; increase decrement by 8
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM3224B

TNIB3224A       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM3224A
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1

                BSF             TEMP,2                  ; increase decrement by 4

NORM3224A       BCF             _C                      ; clear carry bit
```

```
                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN3224
                RLCF            AARGB1,F                ; otherwise, shift left and
                RLCF            AARGB0,F                ; increment decrement
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3224
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN3224             ; 3 left shifts are required
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN3224

NRM3224B        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                CLRF            AARGB1,W
                BCF             TEMP,3                  ; increase decrement by 8
                BSF             TEMP,4
                CPFSGT          AARGB0                  ; if highbyte=0, result=0
                GOTO            RES024

TNIB3224B       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM3224B
                SWAPF           AARGB0,F                ; if so, shift 4 bits

                BSF             TEMP,2                  ; increase decrement by 4

NORM3224B       BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN3224
                RLCF            AARGB0,F                ; otherwise, shift left and
                INCF            TEMP,F                  ; increment decrement
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3224
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN3224             ; 3 left shifts are required
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN3224

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1

;       Use:    CALL    INT2416
;               CALL    INT24

;       Output: 16 bit 2's complement integer right justified in AARGB0, AARGB1

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     10+36+12 = 58 clks              RND = 0
;                       10+36+20 = 66 clks              RND = 1, SAT = 0
;                       10+36+24 = 70 clks              RND = 1, SAT = 1
```

```
;       Min Timing:     4 clks

;       PM: 127                                         DM: 6

;-----------------------------------------------------------------------------------------------

INT2416
INT24
                CLRF            AARGB2,W
                CPFSGT          EXP                     ; test for zero argument
                RETLW           0x00
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BSF             AARGB0,MSB              ; make MSB explicit

                MOVLW           EXPBIAS+D'15'           ; remove bias+15 from EXP
                SUBWF           EXP,W

                BTFSS           WREG,MSB                ; if >= 15, integer overflow
                GOTO            SETIOV2416              ; will occur
                NEGW            EXP,F

                MOVLW           7                       ; do byte shift if EXP >= 8
                CPFSGT          EXP
                GOTO            SNIB2416
                SUBWF           EXP,F                   ; EXP = EXP - 7
                MOVFP           AARGB1,AARGB2           ; save for rounding
                MOVFP           AARGB0,AARGB1
                CLRF            AARGB0,F
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0

                CPFSGT          EXP
                GOTO            SNIB2416A
                SUBWF           EXP,F                   ; EXP = EXP - 7
                MOVFP           AARGB1,AARGB2           ; save for rounding
                CLRF            AARGB1,F
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0

SNIB2416B       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT2416B
                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0

SHIFT2416B      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                GOTO            SHIFT2416OK

SNIB2416A       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT2416A
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB1,W
                MOVFP           WREG,AARGB2             ; save for rounding
```

```
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0

SHIFT2416A      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB1,F                ; right shift by EXP
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                GOTO            SHIFT2416OK

SNIB2416        MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT2416
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB1,W
                MOVFP           WREG,AARGB2             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                SWAPF           AARGB0,W
                ANDLW           0xF0
                ADDWF           AARGB1,F
                SWAPF           AARGB0,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB0
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0

SHIFT2416       BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB0,F                ; right shift by EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2416OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F

SHIFT2416OK
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB2,MSB              ; is NSB > 0x80?
                GOTO            INT2416OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB2                  ; if NSB = 0x80, select even
                RRCF            AARGB1,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTFSC           AARGB0,MSB
```

```
                GOTO            SETIOV2416

INT2416OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB1,F
                COMF            AARGB0,F
                INFSNZ          AARGB1,F
                INCF            AARGB0,F
                RETLW           0

SETIOV2416      BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0,F                ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 16 bit integer
                SETF            AARGB0,F                ; SIGN = 0, 0x 7F FF
                MOVPF           AARGB0,AARGB1           ; SIGN = 1, 0x 80 00
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1

;       Use:    CALL    INT2424

;       Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     11+45+15 = 71 clks              RND = 0
;                       11+45+23 = 79 clks              RND = 1, SAT = 0
;                       11+45+26 = 82 clks              RND = 1, SAT = 1

;       Min Timing:     4 clks

;       PM: 185                                         DM: 7

;-----------------------------------------------------------------------------------------

INT2424
                CLRF            AARGB2,W
                CPFSGT          EXP                     ; test for zero argument
                RETLW           0x00
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BSF             AARGB0,MSB              ; make MSB explicit

                CLRF            AARGB3,F

                MOVLW           EXPBIAS+D'23'           ; remove bias+23 from EXP
                SUBWF           EXP,W

                BTFSS           WREG,MSB                ; if >= 23, integer overflow
                GOTO            SETIOV2424              ; will occur
                NEGW            EXP,F

                MOVLW           7                       ; do byte shift if EXP >= 8
                CPFSGT          EXP
                GOTO            SNIB2424
                SUBWF           EXP,F                   ; EXP = EXP - 7
                MOVFP           AARGB2,AARGB3           ; save for rounding
```

```
            MOVFP       AARGB1,AARGB2
            MOVFP       AARGB0,AARGB1
            CLRF        AARGB0,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0

            CPFSGT      EXP                     ; do another byte shift if EXP >= 8
            GOTO        SNIB2424A
            SUBWF       EXP,F                   ; EXP = EXP - 7
            MOVFP       AARGB2,AARGB3           ; save for rounding
            MOVFP       AARGB1,AARGB2
            CLRF        AARGB1,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0

            CPFSGT      EXP                     ; do another byte shift if EXP >= 8
            GOTO        SNIB2424B
            SUBWF       EXP,F                   ; EXP = EXP - 7
            MOVFP       AARGB2,AARGB3           ; save for rounding
            CLRF        AARGB2,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0

SNIB2424C   MOVLW       3                       ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT2424C
            SWAPF       AARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB3
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0

SHIFT2424C  BCF         _C                      ; at most 3 right shifts are required
            RRCF        AARGB3,F                ; right shift by EXP
            DCFSNZ      EXP,F
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            DCFSNZ      EXP,F
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            GOTO        SHIFT2424OK

SNIB2424B   MOVLW       3                       ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT2424B
            SUBWF       EXP,F                   ; EXP = EXP - 3
            SWAPF       AARGB2,W
            MOVPF       WREG,AARGB3             ; save for rounding
            ANDLW       0x0F
            MOVPF       WREG,AARGB2
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0

SHIFT2424B  BCF         _C                      ; at most 3 right shifts are required
            RRCF        AARGB2,F                ; right shift by EXP
            RRCF        AARGB3,F
            DCFSNZ      EXP,F
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            DCFSNZ      EXP,F
            GOTO        SHIFT2424OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB2,F
```

```
                RRCF            AARGB3,F
                GOTO            SHIFT2424OK

SNIB2424A       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT2424A
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB2,W
                MOVPF           WREG,AARGB3             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0

SHIFT2424A      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB1,F                ; right shift by EXP
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                GOTO            SHIFT2424OK

SNIB2424        MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT2424
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB2,W
                MOVPF           WREG,AARGB3             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1

                SWAPF           AARGB0,W
                ANDLW           0xF0
                ADDWF           AARGB1,F

                SWAPF           AARGB0,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB0
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0
```

```
SHIFT2424       BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB0,F                ; right shift by EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT2424OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F

SHIFT2424OK
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB3,MSB              ; is NSB > 0x80?
                GOTO            INT2424OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB3                  ; if NSB = 0x80, select even
                RRCF            AARGB2,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTFSC           AARGB0,MSB
                GOTO            SETIOV2424

INT2424OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                RETLW           0

SETIOV2424      BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0,F                ; saturate to largest two’s
                BTFSS           SIGN,MSB                ; complement 24 bit integer
                SETF            AARGB0,F                ; SIGN = 0, 0x 7F FF FF
                MOVPF           AARGB0,AARGB1           ; SIGN = 1, 0x 80 00 00
                MOVPF           AARGB0,AARGB2
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Multiply

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1
```

```
;       Use:    CALL    FPM24

;       Output: 24 bit floating point product in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG * BARG

;       Max Timing:     19+33+8 = 60 clks               RND = 0
;                       19+33+20 = 72 clks              RND = 1, SAT = 0
;                       19+33+27 = 79 clks              RND = 1, SAT = 1

;       Min Timing:     5+5 = 10 clks                   AARG * BARG = 0
;                       13+5+23+11 = 52 clks

;       PM: 80                                          DM: 11

;-----------------------------------------------------------------------------------------------

FPM24           CLRF            AARGB2,W                ; test for zero arguments
                CPFSEQ          BEXP
                CPFSGT          AEXP
                GOTO            RES024

M24BNE0         MOVFP           AARGB0,WREG
                XORWF           BARGB0,W
                MOVPF           WREG,SIGN               ; save sign in SIGN

                MOVFP           BEXP,WREG
                ADDWF           EXP,F
                MOVLW           EXPBIAS-1
                BTFSS           _C
                GOTO            MTUN24

                SUBWF           EXP,F                   ; remove bias and overflow test
                BTFSC           _C
                GOTO            SETFOV24
                GOTO            MOK24

MTUN24          SUBWF           EXP,F                   ; remove bias and underflow test
                BTFSS           _C
                GOTO            SETFUN24

MOK24           BSF             AARGB0,MSB              ; make argument MSB’s explicit
                BSF             BARGB0,MSB

                MOVPF           AARGB1,TEMPB1

                MOVFP           AARGB1,WREG             ; multiply mantissas
                MULWF           BARGB1
                MOVPF           PRODH,AARGB2

                MOVFP           AARGB0,WREG
                MULWF           BARGB0
                MOVPF           PRODH,AARGB0
                MOVPF           PRODL,AARGB1

                MULWF           BARGB1
                MOVPF           PRODL,WREG
                ADDWF           AARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB1,F
                CLRF            WREG,F
                ADDWFC          AARGB0,F

                MOVFP           TEMPB1,WREG
                MULWF           BARGB0
```

```
                MOVPF           PRODL,WREG
                ADDWF           AARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB1,F
                CLRF            WREG,F
                ADDWFC          AARGB0,F

                BTFSC           AARGB0,MSB              ; check for postnormalization
                GOTO            MROUND24
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                DECF            EXP,F
                BTFSC           _Z
                GOTO            SETFUN24

MROUND24
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB2,MSB              ; is NSB > 0x80?
                GOTO            MUL24OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB2                  ; if NSB = 0x80, select even
                RRCF            AARGB1,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                BTFSS           _C                      ; has rounding caused carryout?
                GOTO            MUL24OK
                RRCF            AARGB0,F                ; if so, right shift
                RRCF            AARGB1,F
                INFSNZ          EXP,F                   ; test for floating point overflow
                GOTO            SETFOV24

MUL24OK         BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive

                RETLW           0

SETFOV24        BSF             FPFLAGS,FOV             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                SETF            AEXP,F                  ; saturate to largest floating
                SETF            AARGB0,F                ; point number = 0x FF 7F FF
                SETF            AARGB1,F                ; modulo the appropriate sign bit
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*****************************************************************************************
;*****************************************************************************************

;       Floating Point Divide

;       Input:  24 bit floating point dividend in AEXP, AARGB0, AARGB1
;               24 bit floating point divisor in BEXP, BARGB0, BARGB1

;       Use:    CALL    FPD24

;       Output: 24 bit floating point quotient in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG / BARG
```

```
;       Max Timing:     10+12+30+38+67+19 = 176 clks    RND = 0
;                       10+12+30+38+67+28 = 185 clks    RND = 1, SAT = 0
;                       10+12+30+38+67+35 = 192 clks    RND = 1, SAT = 1

;       Min Timing:     6+5 = 11 clks                   AARG = 0

;       PM: 201+256 = 457                               DM: 13

;        In addition to those registers defined in MATH17.INC, this routine uses
;        TBLPTRL and TBLPTRH without saving and restoring.

;-----------------------------------------------------------------------------------------------

FPD24SEED        macro

;       Timing:     12 clks

;        PM: 11+257 = 268

;        generation of F0 using 16 bit zeroth degree minimax approximations to the
;        reciprocal of BARG, with the top 8 explicit bits of BARG as a pointer.

                MOVLW               HIGH (IBTBL256M)    ; access table for F0
                MOVWF               TBLPTRH
                RLCF                BARGB1,W
                RLCF                BARGB0,W
                ADDLW               LOW (IBTBL256M)
                MOVWF               TBLPTRL
                BTFSC               _C
                INCF                TBLPTRH,F
                TABLRD              0,1,TEMPB0
                TLRD                1,TEMPB0
                TLRD                0,TEMPB1

                endm

;-----------------------------------------------------------------------------------------------

FPD24SEEDS        macro

;       Timing:     64 clks

;        PM: 62+17 = 79

;        generation of F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 4 explicit bits of BARG as a pointer
;        and the remaining 11 explicit bits as the argument to linear interpolation.

                MOVLW               HIGH (IBTBL16I)     ; access table for F0
                MOVWF               TBLPTRH
                RLCF                BARGB0,W
                SWAPF               WREG,F
                ANDLW               0x0F
                ADDLW               LOW (IBTBL16I)
                MOVWF               TBLPTRL
                BTFSC               _C
                INCF                TBLPTRH,F
                TABLRD              0,1,TEMPB0
                TLRD                1,TEMPB0
                TABLRD              0,0,TEMPB1
                TLRD                1,AARGB4
                TLRD                0,AARGB5

                MOVFP               AARGB5,WREG         ; calculate difference
                SUBWF               TEMPB1,W
                MOVWF               AARGB5
```

```
                MOVFP               AARGB4,WREG
                SUBWFB              TEMPB0,W
                MOVWF               AARGB4

                MOVPF               AARGB5,TBLPTRL
                MOVFP               BARGB0,WREG
                ANDLW               0x07
                MOVPF               WREG,TEMPB2

                MOVFP               AARGB5,WREG
                MULWF               BARGB1
                MOVPF               PRODL,TBLPTRH

                MOVFP               AARGB4,WREG
                MULWF               TEMPB2
                MOVPF               PRODH,AARGB4
                MOVPF               PRODL,AARGB5

                MULWF               BARGB1
                MOVPF               PRODL,WREG
                ADDWF               TBLPTRH,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB5,F
                CLRF                WREG,F
                ADDWFC              AARGB4,F

                MOVFP               TBLPTRL,WREG
                MULWF               TEMPB2
                MOVPF               PRODL,WREG
                ADDWF               TBLPTRH,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB5,F
                CLRF                WREG,F
                ADDWFC              AARGB4,F

                CLRF                _C
                RRCF                AARGB4,F
                RRCF                AARGB5,F
                RRCF                TBLPTRL,F
                CLRF                _C
                RRCF                AARGB4,F
                RRCF                AARGB5,F
                RRCF                TBLPTRL,F
                CLRF                _C
                RRCF                AARGB4,F
                RRCF                AARGB5,F
                RRCF                TBLPTRL,F

                MOVFP               TBLPTRL,WREG
                SUBWF               TEMPB1,F
                MOVPF               AARGB5,WREG
                SUBWFB              TEMPB0,F                ; F0

                endm

;-----------------------------------------------------------------------------------------

FPD24           CLRF                TEMPB3,W                ; clear exponent modification
                CPFSGT              BEXP                    ; test for divide by zero
                GOTO                SETFDZ24

                CPFSGT              AEXP
                GOTO                RES024

D24BNE0         MOVFP               AARGB0,WREG
                XORWF               BARGB0,W
```

```
            MOVPF       WREG,SIGN           ; save sign in SIGN

            BSF         AARGB0,MSB          ; make argument MSB's explicit
            BSF         BARGB0,MSB

            FPD24SEED                       ; generation of F0

            MOVPF       AARGB1,AARGB4       ; A0 = F0 * A

            MOVFP       AARGB1,WREG
            MULWF       TEMPB1
            MOVPF       PRODH,AARGB2
            MOVPF       PRODL,AARGB3

            MOVFP       AARGB0,WREG
            MULWF       TEMPB0
            MOVPF       PRODH,AARGB0
            MOVPF       PRODL,AARGB1

            MULWF       TEMPB1
            MOVPF       PRODL,WREG
            ADDWF       AARGB2,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB1,F
            CLRF        WREG,F
            ADDWFC      AARGB0,F

            MOVFP       AARGB4,WREG
            MULWF       TEMPB0
            MOVPF       PRODL,WREG
            ADDWF       AARGB2,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB1,F
            CLRF        WREG,F
            ADDWFC      AARGB0,F

            BTFSC       AARGB0,MSB
            GOTO        DAOK24
            RLCF        AARGB2,F
            RLCF        AARGB1,F
            RLCF        AARGB0,F
            DECF        TEMPB3,F

DAOK24      MOVFP       BARGB1,AARGB4       ; B0 = F0 * B

            MOVFP       BARGB1,WREG
            MULWF       TEMPB1
            MOVPF       PRODH,BARGB2
            MOVPF       PRODL,BARGB3

            MOVFP       BARGB0,WREG
            MULWF       TEMPB0
            MOVPF       PRODH,BARGB0
            MOVPF       PRODL,BARGB1

            MULWF       TEMPB1
            MOVPF       PRODL,WREG
            ADDWF       BARGB2,F
            MOVPF       PRODH,WREG
            ADDWFC      BARGB1,F
            CLRF        WREG,F
            ADDWFC      BARGB0,F

            MOVFP       AARGB4,WREG
            MULWF       TEMPB0
            MOVPF       PRODL,WREG
```

```
                ADDWF           BARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          BARGB1,F
                CLRF            WREG,F
                ADDWFC          BARGB0,F

                BTFSS           BARGB0,MSB
                BTFSC           BARGB0,MSB-1
                GOTO            DBOK24
                RLCF            BARGB3,F
                RLCF            BARGB2,F
                RLCF            BARGB1,F
                RLCF            BARGB0,F
                INCF            TEMPB3,F

DBOK24
                COMF            BARGB2,F                ; F1 = 2 - B0
                COMF            BARGB1,F
                COMF            BARGB0,F
                INCF            BARGB2,F
                ADDWFC          BARGB1,F
                ADDWFC          BARGB0,F

                MOVPF           AARGB0,TEMPB0           ; A1 = F1 * A
                MOVPF           AARGB1,TEMPB1
                MOVPF           AARGB2,TEMPB2

                MOVFP           AARGB1,WREG
                MULWF           BARGB1
                MOVPF           PRODH,AARGB2
                MOVPF           PRODL,AARGB3

                MULWF           BARGB2
                MOVPF           PRODH,WREG
                ADDWF           AARGB3,F
                CLRF            WREG,F
                ADDWFC          AARGB2,F

                MOVFP           TEMPB2,WREG
                MULWF           BARGB1
                MOVPF           PRODH,WREG
                ADDWF           AARGB3,F
                CLRF            WREG,F
                ADDWFC          AARGB2,F

                MOVFP           AARGB0,WREG
                MULWF           BARGB2
                MOVPF           PRODL,WREG
                ADDWF           AARGB3,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB2,F
                MOVFP           AARGB0,WREG
                MULWF           BARGB1
                CLRF            AARGB1,W
                ADDWFC          AARGB1,F
                MOVPF           PRODL,WREG
                ADDWF           AARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB1,F

                MOVFP           TEMPB2,WREG
                MULWF           BARGB0
                MOVPF           PRODL,WREG
                ADDWF           AARGB3,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB2,F
```

```
            CLRF            AARGB0,W
            ADDWFC          AARGB1,F
            ADDWFC          AARGB0,F

            MOVFP           TEMPB1,WREG
            MULWF           BARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB2,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB1,F
            CLRF            WREG,F
            ADDWFC          AARGB0,F

            MOVFP           TEMPB0,WREG
            MULWF           BARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB1,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB0,F

            BTFSC           AARGB0,MSB          ; postnormalization
            GOTO            DNORM24
            RLCF            AARGB2,F
            RLCF            AARGB1,F
            RLCF            AARGB0,F
            DECF            TEMPB3,F
DNORM24
            BTFSC           AARGB0,MSB
            GOTO            DEXP24
            RLCF            AARGB2,F
            RLCF            AARGB1,F
            RLCF            AARGB0,F
            DECF            TEMPB3,F
DEXP24
            MOVFP           BEXP,WREG           ; compute AEXP - BEXP
            SUBWF           EXP,F
            MOVLW           EXPBIAS+1           ; add bias + 1 for scaling of F0
            BTFSS           _C
            GOTO            ALTB24

AGEB24      ADDWF           TEMPB3,W
            ADDWF           EXP,F               ; if AEXP > BEXP, test for overflow
            BTFSC           _C
            GOTO            SETFOV24
            GOTO            DROUND24

ALTB24      ADDWF           TEMPB3,W
            ADDWF           EXP,F                ; if AEXP < BEXP, test for underflow
            BTFSS           _C
            GOTO            SETFUN24

DROUND24
            BTFSC           FPFLAGS,RND         ; is rounding enabled?
            BTFSS           AARGB2,MSB          ; is NSB > 0x80?
            GOTO            DIV24OK
            BSF             _C                  ; set carry for rounding
            MOVLW           0x80
            CPFSGT          AARGB2              ; if NSB = 0x80, select even
            RRCF            AARGB1,W            ; using lsb in carry
            CLRF            WREG,F
            ADDWFC          AARGB1,F
            ADDWFC          AARGB0,F

            BTFSS           _C                  ; test if rounding caused carryout
            GOTO            DIV24OK
            RRCF            AARGB0,F
```

```
                RRCF            AARGB1,F
                INFSNZ          EXP,F                   ; test for overflow
                GOTO            SETFOV24

DIV24OK         BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive

                RETLW           0

SETFUN24        BSF             FPFLAGS,FUN             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                MOVLW           0x01                    ; saturate to smallest floating
                MOVPF           WREG,AEXP               ; point number = 0x 01 00 00
                CLRF            AARGB0,F                ; modulo the appropriate sign bit
                CLRF            AARGB1,F
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

SETFDZ24        BSF             FPFLAGS,FDZ             ; set floating point divide by zero
                RETLW           0xFF                    ; flag and return error code in
                                                        ; WREG

;-----------------------------------------------------------------------------------------------

;        table of 16 bit approximations to the reciprocal of BARG,
;        with the top 4 explicit bits of BARG as a pointer and the
;        remaining 11 explicit bits as the argument to linear interpolation.

IBTBL16I
                DATA        0xFFFF
                DATA        0xF0F1
                DATA        0xE38E
                DATA        0xD794
                DATA        0xCCCD
                DATA        0xC30C
                DATA        0xBA2F
                DATA        0xB216
                DATA        0xAAAB
                DATA        0xA3D7
                DATA        0x9D8A
                DATA        0x97B4
                DATA        0x9249
                DATA        0x8D3E
                DATA        0x8889
                DATA        0x8421
                DATA        0x8001

;        table of 16 bit zeroth degree minimax approximations to the
;        reciprocal of BARG, with the top 8 explicit bits of BARG as a pointer.

IBTBL256M
                DATA        0xFF81
                DATA        0xFE83
                DATA        0xFD87
                DATA        0xFC8D
                DATA        0xFB95
                DATA        0xFA9E
                DATA        0xF9AA
                DATA        0xF8B7
                DATA        0xF7C7
                DATA        0xF6D8
                DATA        0xF5EB
                DATA        0xF4FF
```

```
DATA        0xF416
DATA        0xF32E
DATA        0xF248
DATA        0xF163
DATA        0xF080
DATA        0xEF9F
DATA        0xEEC0
DATA        0xEDE2
DATA        0xED06
DATA        0xEC2B
DATA        0xEB52
DATA        0xEA7A
DATA        0xE9A4
DATA        0xE8D0
DATA        0xE7FD
DATA        0xE72B
DATA        0xE65B
DATA        0xE58D
DATA        0xE4C0
DATA        0xE3F4
DATA        0xE32A
DATA        0xE261
DATA        0xE199
DATA        0xE0D3
DATA        0xE00E
DATA        0xDF4B
DATA        0xDE89
DATA        0xDDC8
DATA        0xDD09
DATA        0xDC4A
DATA        0xDB8D
DATA        0xDAD2
DATA        0xDA17
DATA        0xD95E
DATA        0xD8A6
DATA        0xD7EF
DATA        0xD73A
DATA        0xD686
DATA        0xD5D2
DATA        0xD520
DATA        0xD470
DATA        0xD3C0
DATA        0xD311
DATA        0xD264
DATA        0xD1B7
DATA        0xD10C
DATA        0xD062
DATA        0xCFB9
DATA        0xCF11
DATA        0xCE6A
DATA        0xCDC4
DATA        0xCD1F
DATA        0xCC7B
DATA        0xCBD8
DATA        0xCB37
DATA        0xCA96
DATA        0xC9F6
DATA        0xC957
DATA        0xC8B9
DATA        0xC81C
DATA        0xC780
DATA        0xC6E5
DATA        0xC64B
DATA        0xC5B2
DATA        0xC51A
DATA        0xC483
```

```
DATA        0xC3EC
DATA        0xC357
DATA        0xC2C2
DATA        0xC22E
DATA        0xC19C
DATA        0xC10A
DATA        0xC078
DATA        0xBFE8
DATA        0xBF59
DATA        0xBECA
DATA        0xBE3C
DATA        0xBDAF
DATA        0xBD23
DATA        0xBC98
DATA        0xBC0D
DATA        0xBB84
DATA        0xBAFB
DATA        0xBA72
DATA        0xB9EB
DATA        0xB964
DATA        0xB8DF
DATA        0xB859
DATA        0xB7D5
DATA        0xB751
DATA        0xB6CE
DATA        0xB64C
DATA        0xB5CB
DATA        0xB54A
DATA        0xB4CA
DATA        0xB44B
DATA        0xB3CC
DATA        0xB34E
DATA        0xB2D1
DATA        0xB254
DATA        0xB1D8
DATA        0xB15D
DATA        0xB0E3
DATA        0xB069
DATA        0xAFF0
DATA        0xAF77
DATA        0xAEFF
DATA        0xAE88
DATA        0xAE11
DATA        0xAD9B
DATA        0xAD26
DATA        0xACB1
DATA        0xAC3D
DATA        0xABC9
DATA        0xAB56
DATA        0xAAE4
DATA        0xAA72
DATA        0xAA01
DATA        0xA990
DATA        0xA920
DATA        0xA8B1
DATA        0xA842
DATA        0xA7D3
DATA        0xA766
DATA        0xA6F8
DATA        0xA68C
DATA        0xA620
DATA        0xA5B4
DATA        0xA549
DATA        0xA4DF
DATA        0xA475
DATA        0xA40C
```

```
DATA        0xA3A3
DATA        0xA33A
DATA        0xA2D2
DATA        0xA26B
DATA        0xA204
DATA        0xA19E
DATA        0xA138
DATA        0xA0D3
DATA        0xA06E
DATA        0xA00A
DATA        0x9FA6
DATA        0x9F43
DATA        0x9EE0
DATA        0x9E7E
DATA        0x9E1C
DATA        0x9DBA
DATA        0x9D59
DATA        0x9CF9
DATA        0x9C99
DATA        0x9C39
DATA        0x9BDA
DATA        0x9B7C
DATA        0x9B1D
DATA        0x9AC0
DATA        0x9A62
DATA        0x9A05
DATA        0x99A9
DATA        0x994D
DATA        0x98F1
DATA        0x9896
DATA        0x983B
DATA        0x97E1
DATA        0x9787
DATA        0x972E
DATA        0x96D5
DATA        0x967C
DATA        0x9624
DATA        0x95CC
DATA        0x9574
DATA        0x951D
DATA        0x94C7
DATA        0x9470
DATA        0x941A
DATA        0x93C5
DATA        0x9370
DATA        0x931B
DATA        0x92C7
DATA        0x9273
DATA        0x921F
DATA        0x91CC
DATA        0x9179
DATA        0x9127
DATA        0x90D5
DATA        0x9083
DATA        0x9031
DATA        0x8FE0
DATA        0x8F90
DATA        0x8F3F
DATA        0x8EEF
DATA        0x8EA0
DATA        0x8E50
DATA        0x8E02
DATA        0x8DB3
DATA        0x8D65
DATA        0x8D17
DATA        0x8CC9
```

```
                DATA        0x8C7C
                DATA        0x8C2F
                DATA        0x8BE2
                DATA        0x8B96
                DATA        0x8B4A
                DATA        0x8AFE
                DATA        0x8AB3
                DATA        0x8A68
                DATA        0x8A1E
                DATA        0x89D3
                DATA        0x8989
                DATA        0x893F
                DATA        0x88F6
                DATA        0x88AD
                DATA        0x8864
                DATA        0x881B
                DATA        0x87D3
                DATA        0x878B
                DATA        0x8744
                DATA        0x86FC
                DATA        0x86B5
                DATA        0x866F
                DATA        0x8628
                DATA        0x85E2
                DATA        0x859C
                DATA        0x8557
                DATA        0x8511
                DATA        0x84CC
                DATA        0x8487
                DATA        0x8443
                DATA        0x83FF
                DATA        0x83BB
                DATA        0x8377
                DATA        0x8334
                DATA        0x82F1
                DATA        0x82AE
                DATA        0x826B
                DATA        0x8229
                DATA        0x81E7
                DATA        0x81A5
                DATA        0x8164
                DATA        0x8122
                DATA        0x80E1
                DATA        0x80A1
                DATA        0x8060
                DATA        0x8020

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Subtract

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;       Use:    CALL FPS24

;       Output: 24 bit floating point difference in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     1+133 = 134 clks                RND = 0
;                       1+146 = 147 clks                RND = 1, SAT = 0
;                       1+152 = 153 clks                RND = 1, SAT = 1

;       Min Timing:     1+10 = 11 clks
```

```
;       PM: 1+298 = 299                                 DM: 10

;-----------------------------------------------------------------------------------------

FPS24           BTG             BARGB0,MSB          ; toggle sign bit for subtraction

;*****************************************************************************************

;       Floating Point Add

;       Input:  24 bit floating point number in AEXP, AARGB0, AARGB1
;               24 bit floating point number in BEXP, BARGB0, BARGB1

;       Use:    CALL FPA24

;       Output: 24 bit floating point sum in AEXP, AARGB0, AARGB1

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     81+52 = 133 clks            RND = 0
;                       81+65 = 146 clks            RND = 1, SAT = 0
;                       81+71 = 152 clks            RND = 1, SAT = 1

;       Min Timing:     10 clks

;       PM: 298                                     DM: 10

;-----------------------------------------------------------------------------------------

FPA24           MOVFP           AARGB0,WREG         ; exclusive or of signs in TEMP
                XORWF           BARGB0,W
                MOVPF           WREG,TEMP

                CLRF            AARGB2,F            ; clear extended byte

                MOVFP           AEXP,WREG           ; use AARG if AEXP >= BEXP
                CPFSGT          BEXP
                GOTO            USEA24

USEB24          MOVFP           BARGB0,WREG         ; use BARG if AEXP < BEXP
                MOVPF           WREG,SIGN           ; save sign in SIGN
                BSF             BARGB0,MSB          ; make MSB’s explicit
                BSF             AARGB0,MSB

                MOVFP           AEXP,WREG           ; compute shift count in BEXP
                MOVPF           WREG,TEMPB1
                MOVFP           BEXP,WREG
                MOVPF           WREG,AEXP

                CLRF            WREG,F
                CPFSGT          TEMPB1              ; return BARG if AARG = 0
                GOTO            BRETURN24
                MOVFP           TEMPB1,WREG
                SUBWF           BEXP,F
                BTFSC           _Z
                GOTO            BLIGNED24

                MOVLW           7
                CPFSGT          BEXP                ; do byte shift if BEXP >= 8
                GOTO            BNIB24
                SUBWF           BEXP,F              ; BEXP = BEXP - 7
                MOVFP           AARGB1,AARGB2       ; keep for postnormalization
                MOVFP           AARGB0,AARGB1
                CLRF            AARGB0,F
                DCFSNZ          BEXP,F              ; BEXP = BEXP - 1
```

```
                GOTO            BLIGNED24

                CPFSGT          BEXP                    ; do byte shift if BEXP >= 8
                GOTO            BNIB24A
                SUBWF           BEXP,F                  ; BEXP = BEXP - 7
                MOVFP           AARGB1,AARGB2           ; keep for postnormalization
                CLRF            AARGB1,F
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            BLIGNED24

                CPFSGT          BEXP                    ; if BEXP still >= 8, then
                GOTO            BNIB24B                 ; AARG = 0 relative to BARG

BRETURN24       MOVFP           SIGN,AARGB0             ; return BARG
                MOVFP           BARGB1,AARGB1
                CLRF            AARGB2,F
                RETLW           0x00

BNIB24B         MOVLW           3                       ; do nibbleshift if BEXP >= 4
                CPFSGT          BEXP
                GOTO            BLOOP24B
                SUBWF           BEXP,F                  ; BEXP = BEXP -3
                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            BLIGNED24               ; aligned if BEXP = 0

BLOOP24B        BCF             _C                      ; right shift by BEXP
                RRCF            AARGB2,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED24               ; aligned if BEXP = 0
                BCF             _C
                RRCF            AARGB2,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED24               ; aligned if BEXP = 0
                BCF             _C                      ; at most 3 right shifts are
                RRCF            AARGB2,F                ; possible
                GOTO            BLIGNED24

BNIB24A         MOVLW           3                       ; do nibbleshift if BEXP >= 4
                CPFSGT          BEXP
                GOTO            BLOOP24A
                SUBWF           BEXP,F                  ; BEXP = BEXP -3
                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F
                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            BLIGNED24               ; aligned if BEXP = 0

BLOOP24A        BCF             _C                      ; right shift by BEXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED24               ; aligned if BEXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED24               ; aligned if BEXP = 0
```

```
            BCF         _C                  ; at most 3 right shifts are
            RRCF        AARGB1,F            ; possible
            RRCF        AARGB2,F
            GOTO        BLIGNED24

BNIB24      MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        BLOOP24
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       AARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB2
            SWAPF       AARGB1,W
            ANDLW       0xF0
            ADDWF       AARGB2,F
            SWAPF       AARGB1,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB1
            SWAPF       AARGB0,W
            ANDLW       0xF0
            ADDWF       AARGB1,F
            SWAPF       AARGB0,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB0
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED24           ; aligned if BEXP = 0

BLOOP24     BCF         _C                  ; right shift by BEXP
            RRCF        AARGB0,F
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED24           ; aligned if BEXP = 0
            BCF         _C
            RRCF        AARGB0,F
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED24           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        AARGB0,F            ; possible
            RRCF        AARGB1,F
            RRCF        AARGB2,F

BLIGNED24   CLRF        BARGB2,W
            BTFSS       TEMP,MSB            ; negate if signs opposite
            GOTO        AOK24
            COMF        AARGB2,F
            COMF        AARGB1,F
            COMF        AARGB0,F
            INCF        AARGB2,F
            ADDWFC      AARGB1,F
            ADDWFC      AARGB0,F
            GOTO        AOK24

USEA24      TSTFSZ      BEXP                ; return AARG if BARG = 0
            GOTO        BNE024
            RETLW       0x00

BNE024      CLRF        BARGB2,F
            MOVPF       AARGB0,SIGN         ; save sign in SIGN
            BSF         AARGB0,MSB          ; make MSB’s explicit
            BSF         BARGB0,MSB

            MOVFP       BEXP,WREG           ; compute shift count in BEXP
            SUBWF       AEXP,W
```

```
            MOVPF       WREG,BEXP
            BTFSC       _Z
            GOTO        ALIGNED24

            MOVLW       7
            CPFSGT      BEXP                ; do byte shift if BEXP >= 8
            GOTO        ANIB24

            SUBWF       BEXP,F              ; BEXP = BEXP - 7
            MOVFP       BARGB1,WREG
            MOVPF       WREG,BARGB2         ; keep for postnormalization
            MOVFP       BARGB0,WREG
            MOVPF       WREG,BARGB1
            CLRF        BARGB0,F
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED24

            MOVLW       7
            CPFSGT      BEXP                ; if BEXP still >= 8, then
            GOTO        ANIB24A             ; BARG = 0 relative to AARG

            SUBWF       BEXP,F              ; BEXP = BEXP - 7
            MOVFP       BARGB1,WREG
            MOVPF       WREG,BARGB2         ; keep for postnormalization
            CLRF        BARGB1,F
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED24

            MOVLW       7
            CPFSGT      BEXP                ; if BEXP still >= 8, then
            GOTO        ANIB24B             ; BARG = 0 relative to AARG

            MOVFP       SIGN,AARGB0         ; return AARG
            RETLW       0x00

ANIB24B     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP24B
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB2
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED24           ; aligned if BEXP = 0

ALOOP24B    BCF         _C                  ; right shift by BEXP
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        BARGB2,F            ; possible
            GOTO        ALIGNED24

ANIB24A     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP24A
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB2
            SWAPF       BARGB1,W
            ANDLW       0xF0
```

```
            ADDWF       BARGB2,F
            SWAPF       BARGB1,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB1
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED24           ; aligned if BEXP = 0

ALOOP24A    BCF         _C                  ; right shift by BEXP
            RRCF        BARGB1,F
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C
            RRCF        BARGB1,F
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        BARGB1,F            ; possible
            RRCF        BARGB2,F
            GOTO        ALIGNED24

ANIB24      MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP24
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB2
            SWAPF       BARGB1,W
            ANDLW       0xF0
            ADDWF       BARGB2,F
            SWAPF       BARGB1,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB1
            SWAPF       BARGB0,W
            ANDLW       0xF0
            ADDWF       BARGB1,F
            SWAPF       BARGB0,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB0
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED24           ; aligned if BEXP = 0

ALOOP24     BCF         _C                  ; right shift by BEXP
            RRCF        BARGB0,F
            RRCF        BARGB1,F
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C
            RRCF        BARGB0,F
            RRCF        BARGB1,F
            RRCF        BARGB2,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED24           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        BARGB0,F            ; possible
            RRCF        BARGB1,F
            RRCF        BARGB2,F

ALIGNED24   CLRF        AARGB2,W
            BTFSS       TEMP,MSB            ; negate if signs opposite
            GOTO        AOK24
            COMF        BARGB2,F
            COMF        BARGB1,F
```

```
                COMF            BARGB0,F
                INCF            BARGB2,F
                ADDWFC          BARGB1,F
                ADDWFC          BARGB0,F

AOK24           MOVFP           BARGB2,WREG
                ADDWF           AARGB2,F
                MOVFP           BARGB1,WREG             ; add
                ADDWFC          AARGB1,F
                MOVFP           BARGB0,WREG
                ADDWFC          AARGB0,F

                BTFSC           TEMP,MSB
                GOTO            ACOMP24
                BTFSS           _C
                GOTO            NRMRND3224

                RRCF            AARGB0,F                ; shift right and increment EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                INCFSZ          AEXP,F
                GOTO            NRMRND3224
                GOTO            SETFOV24                ; set floating point overflow flag

ACOMP24         BTFSC           _C
                GOTO            NRM3224                 ; normalize and fix sign

                COMF            AARGB2,F                ; negate, toggle sign bit and
                COMF            AARGB1,F                ; then normalize
                COMF            AARGB0,F
                INCF            AARGB2,F
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTG             SIGN,MSB
                GOTO            NRM3224
```

Please check the Microchip BBS for the latest version of the source code. For BBS access information, see Section 6, Microchip Bulletin Board Service information, page 6-3.

APPENDIX E: PIC16CXXX 32-BIT FLOATING POINT LIBRARY

```
;       RCS Header $Id: fp32.a16 2.8 1996/10/07 13:50:59 F.J.Testa Exp $

;       $Revision: 2.8 $

;       PIC16 32-BIT FLOATING POINT LIBRARY
;
;       Unary operations: both input and output are in AEXP,AARG
;
;       Binary operations: input in AEXP,AARG and BEXP,BARG with output in AEXP,AARG
;
;       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
;       for an error condition specified in FPFLAGS.
;
;       All timings are worst case cycle counts
;
;         Routine               Function
;
;       FLO2432         24 bit integer to 32 bit floating point conversion
;       FLO32
;
;               Timing:            RND
;                                0       1
;
;                       0       104     104
;                  SAT
;                       1       110     110
;
;       NRM3232   32 bit normalization of unnormalized 32 bit floating point numbers
;       NRM32
;
;               Timing:            RND
;                                0       1
;
;                       0       90      90
;                  SAT
;                       1       96      96
;
;
;       INT3224         32 bit floating point to 24 bit integer conversion
;       INT32
;
;
;               Timing:            RND
;                                0       1
;
;                       0       104      112
;                  SAT
;                       1       104      114
;
;       FLO3232 32 bit integer to 32 bit floating point conversion
;
;               Timing:            RND
;                                0       1
;
;                       0       129     145
;                  SAT
;                       1       129     152
;
```

```
;       NRM4032 32 bit normalization of unnormalized 40 bit floating point numbers
;
;               Timing:            RND
;                               0       1
;
;                       0       112     128
;                  SAT
;                       1       112     135
;
;
;       INT3232         32 bit floating point to 32 bit integer conversion
;
;
;               Timing:            RND
;                               0       1
;
;                       0       130     137
;                  SAT
;                       1       130     137
;
;       FPA32           32 bit floating point add
;
;               Timing:            RND
;                               0       1
;
;                       0       251     265
;                  SAT
;                       1       251     271
;
;       FPS32           32 bit floating point subtract
;
;               Timing:            RND
;                               0       1
;
;                       0       253     267
;                  SAT
;                       1       253     273
;
;       FPM32           32 bit floating point multiply
;
;               Timing:            RND
;                               0       1
;
;                       0       574     588
;                  SAT
;                       1       574     591
;
;       FPD32           32 bit floating point divide
;
;               Timing:            RND
;                               0       1
;
;                       0       932     968
;                  SAT
;                       1       932     971
;
;
;*******************************************************************************************
;*******************************************************************************************
;
;       32 bit floating point representation
;
;       EXPONENT        8 bit biased exponent
;
;                       It is important to note that the use of biased exponents produces
;                       a unique representation of a floating point 0, given by
```

```
;                       EXP = HIGHBYTE = MIDBYTE = LOWBYTE = 0x00, with 0 being
;                       the only number with EXP = 0.
;
;       HIGHBYTE        8 bit most significant byte of fraction in sign-magnitude representation,
;                       with SIGN = MSB, implicit MSB = 1 and radix point to the right of MSB
;
;       MIDBYTE         8 bit middle significant byte of sign-magnitude fraction
;
;       LOWBYTE         8 bit least significant byte of sign-magnitude fraction
;
;       EXPONENT        HIGHBYTE        MIDBYTE         LOWBYTE
;
;       xxxxxxxx        S.xxxxxxx       xxxxxxxx        xxxxxxxx
;
;                        |
;                      RADIX
;                      POINT
;
;*****************************************************************************************
;*****************************************************************************************

;       Integer to float conversion

;       Input:  24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Use:    CALL    FLO2432 or      CALL    FLO32

;       Output: 32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     14+90 = 104 clks                SAT = 0
;                       14+96 = 110 clks                SAT = 1

;       Min Timing:     6+28 = 34 clks                  AARG = 0
;                       6+18 = 24 clks

;       PM: 14+38 = 52                                  DM: 7

;-----------------------------------------------------------------------------------------

FLO2432
FLO32           MOVLW           D'23'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                CLRF            SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM3232
                COMF            AARGB2,F                ; if < 0, negate and set MSB in SIGN
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BSF             SIGN,MSB
```

```
;*****************************************************************************************

;       Normalization routine

;       Input:  32 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2, with sign in SIGN,MSB

;       Use:    CALL    NRM3232 or      CALL    NRM32

;       Output: 32 bit normalized floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     21+6+7*8+7 = 90 clks            SAT = 0
;                       21+6+7*8+1+12 = 96 clks         SAT = 1

;       Min Timing:     22+6 = 28 clks                  AARG = 0
;                       5+9+4 = 18 clks

;       PM: 38                                          DM: 7

;-----------------------------------------------------------------------------------------

NRM3232

NRM32           CLRF            TEMP                    ; clear exponent decrement
                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM3232
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                MOVF            AARGB2,W
                MOVWF           AARGB1
                CLRF            AARGB2
                BSF             TEMP,3                  ; increase decrement by 8

                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM3232
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                CLRF            AARGB1
                BCF             TEMP,3                  ; increase decrement by 8
                BSF             TEMP,4

                MOVF            AARGB0,W                ; if highbyte=0, result=0
                BTFSC           _Z
                GOTO            RES032

NORM3232        MOVF            TEMP,W
                SUBWF           EXP,F
                BTFSS           _Z
                BTFSS           _C
                GOTO            SETFUN32

                BCF             _C                      ; clear carry bit

NORM3232A       BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            FIXSIGN32
                RLF             AARGB2,F                ; otherwise, shift left and
                RLF             AARGB1,F                ; decrement EXP
                RLF             AARGB0,F
                DECFSZ          EXP,F
                GOTO            NORM3232A

                GOTO            SETFUN32                ; underflow if EXP=0
```

```
FIXSIGN32       BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive
                RETLW           0

RES032          CLRF            AARGB0                  ; result equals zero
                CLRF            AARGB1
                CLRF            AARGB2
                CLRF            AARGB3
                CLRF            EXP
                RETLW           0

;*****************************************************************************************
;*****************************************************************************************

;       Integer to float conversion

;       Input:  32 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2,
;               AARGB3

;       Use:    CALL    FLO3232

;       Output: 32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     17+112 = 129 clks               RND = 0
;                       17+128 = 145 clks               RND = 1, SAT = 0
;                       17+135 = 152 clks               RND = 1, SAT = 1

;       Min Timing:     6+39 = 45 clks                  AARG = 0
;                       6+22 = 28 clks

;       PM: 17+66 = 83                                  DM: 8

;-----------------------------------------------------------------------------------------

FLO3232         MOVLW           D'31'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                CLRF            SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM4032
                COMF            AARGB3,F                ; if < 0, negate and set MSB in SIGN
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB3,F
                BTFSC           _Z
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BSF             SIGN,MSB
```

```
;*****************************************************************************************

;       Normalization routine

;       Input:  40 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2, AARGB3 with sign in SIGN,MSB

;       Use:    CALL    NRM4032

;       Output: 32 bit normalized floating point number in AEXP, AARGB0, AARGB1, AARGB2,
;               AARGB3

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     38+6*9+12+8 = 112 clks          RND = 0
;                       38+6*9+12+24 = 128 clks         RND = 1, SAT = 0
;                       38+6*9+12+31 = 135 clks         RND = 1, SAT = 1

;       Min Timing:     33+6 = 39 clks                  AARG = 0
;                       5+9+8 = 22 clks

;       PM: 66                                          DM: 8

;-----------------------------------------------------------------------------------------

NRM4032         CLRF            TEMP                    ; clear exponent decrement
                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM4032
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                MOVF            AARGB2,W
                MOVWF           AARGB1
                MOVF            AARGB3,W
                MOVWF           AARGB2
                CLRF            AARGB3
                BSF             TEMP,3                  ; increase decrement by 8

                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM4032
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                MOVF            AARGB2,W
                MOVWF           AARGB1
                CLRF            AARGB2
                BCF             TEMP,3                  ; increase decrement by 8
                BSF             TEMP,4

                MOVF            AARGB0,W                ; test if highbyte=0
                BTFSS           _Z
                GOTO            NORM4032
                MOVF            AARGB1,W                ; if so, shift 8 bits by move
                MOVWF           AARGB0
                CLRF            AARGB1
                BSF             TEMP,3                  ; increase decrement by 8

                MOVF            AARGB0,W                ; if highbyte=0, result=0
                BTFSC           _Z
                GOTO            RES032

NORM4032        MOVF            TEMP,W
                SUBWF           EXP,F
                BTFSS           _Z
                BTFSS           _C
                GOTO            SETFUN32
```

```
                BCF             _C                      ; clear carry bit

NORM4032A       BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            NRMRND4032
                RLF             AARGB3,F                ; otherwise, shift left and
                RLF             AARGB2,F                ; decrement EXP
                RLF             AARGB1,F
                RLF             AARGB0,F
                DECFSZ          EXP,F
                GOTO            NORM4032A

                GOTO            SETFUN32                ; underflow if EXP=0

NRMRND4032      BTFSC           FPFLAGS,RND
                BTFSS           AARGB2,LSB
                GOTO            FIXSIGN32
                BTFSS           AARGB3,MSB              ; round if next bit is set
                GOTO            FIXSIGN32
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

                BTFSS           _Z                      ; has rounding caused carryout?
                GOTO            FIXSIGN32
                RRF             AARGB0,F                ; if so, right shift
                RRF             AARGB1,F
                RRF             AARGB2,F
                INCF            EXP,F
                BTFSC           _Z                      ; check for overflow
                GOTO            SETFOV32
                GOTO            FIXSIGN32

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Use:    CALL    INT3224         or      CALL    INT32

;       Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     40+6*7+6+16 = 104 clks          RND = 0
;                       40+6*7+6+24 = 112 clks          RND = 1, SAT = 0
;                       40+6*7+6+26 = 114 clks          RND = 1, SAT = 1

;       Min Timing:     4 clks

;       PM: 82                                          DM: 6

;-------------------------------------------------------------------------------------------

INT3224
INT32
                MOVF            EXP,W                   ; test for zero argument
                BTFSC           _Z
                RETLW           0x00

                MOVF            AARGB0,W                ; save sign in SIGN
                MOVWF           SIGN
```

```
                BSF             AARGB0,MSB              ; make MSB explicit

                MOVLW           EXPBIAS+D'23'           ; remove bias from EXP
                SUBWF           EXP,F
                BTFSS           EXP,MSB
                GOTO            SETIOV3224
                COMF            EXP,F
                INCF            EXP,F

                MOVLW           8                       ; do byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3224
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                MOVF            AARGB1,W
                MOVWF           AARGB2
                MOVF            AARGB0,W
                MOVWF           AARGB1
                CLRF            AARGB0

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3224
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                MOVF            AARGB1,W
                MOVWF           AARGB2
                CLRF            AARGB1

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3224
                MOVWF           EXP
                RLF             AARGB2,F                ; rotate next bit for rounding
                CLRF            AARGB2
                MOVF            EXP,W
                BTFSS           _Z
                BCF             _C
                GOTO            SHIFT3224OK

TSHIFT3224      MOVF            EXP,W                   ; shift completed if EXP = 0
                BTFSC           _Z
                GOTO            SHIFT3224OK

SHIFT3224       BCF             _C
                RRF             AARGB0,F                ; right shift by EXP
                RRF             AARGB1,F
                RRF             AARGB2,F
                DECFSZ          EXP,F
                GOTO            SHIFT3224

SHIFT3224OK     BTFSC           FPFLAGS,RND
                BTFSS           AARGB2,LSB
                GOTO            INT3224OK
                BTFSS           _C
                GOTO            INT3224OK
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BTFSC           AARGB0,MSB              ; test for overflow
                GOTO            SETIOV3224
```

```
INT3224OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB0,F
                COMF            AARGB1,F
                COMF            AARGB2,F
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                RETLW           0

IRES03224       CLRF            AARGB0                  ; integer result equals zero
                CLRF            AARGB1
                CLRF            AARGB2
                RETLW           0

SETIOV3224      BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0                  ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 24 bit integer
                MOVLW           0xFF
                MOVWF           AARGB0                  ; SIGN = 0, 0x 7F FF FF
                MOVWF           AARGB1                  ; SIGN = 1, 0x 80 00 00
                MOVWF           AARGB2
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Use:    CALL    INT3232

;       Output: 32 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2,
;               AARGB3

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     54+6*8+7+21 = 130 clks          RND = 0
;                       54+6*8+7+29 = 137 clks          RND = 1, SAT = 0
;                       54+6*8+7+29 = 137 clks          RND = 1, SAT = 1

;       Min Timing:     5 clks

;       PM: 102                                         DM: 7

;-------------------------------------------------------------------------------------------

INT3232
                CLRF            AARGB3
                MOVF            EXP,W                   ; test for zero argument
                BTFSC           _Z
                RETLW           0x00

                MOVF            AARGB0,W                ; save sign in SIGN
                MOVWF           SIGN
                BSF             AARGB0,MSB              ; make MSB explicit
```

```
                MOVLW           EXPBIAS+D'31'           ; remove bias from EXP
                SUBWF           EXP,F
                BTFSS           EXP,MSB
                GOTO            SETIOV32
                COMF            EXP,F
                INCF            EXP,F

                MOVLW           8                       ; do byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3232
                MOVWF           EXP
                RLF             AARGB3,F                ; rotate next bit for rounding
                MOVF            AARGB2,W
                MOVWF           AARGB3
                MOVF            AARGB1,W
                MOVWF           AARGB2
                MOVF            AARGB0,W
                MOVWF           AARGB1
                CLRF            AARGB0

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3232
                MOVWF           EXP
                RLF             AARGB3,F                ; rotate next bit for rounding
                MOVF            AARGB2,W
                MOVWF           AARGB3
                MOVF            AARGB1,W
                MOVWF           AARGB2
                CLRF            AARGB1

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3232
                MOVWF           EXP
                RLF             AARGB3,F                ; rotate next bit for rounding
                MOVF            AARGB2,W
                MOVWF           AARGB3
                CLRF            AARGB2

                MOVLW           8                       ; do another byte shift if EXP >= 8
                SUBWF           EXP,W
                BTFSS           _C
                GOTO            TSHIFT3232
                MOVWF           EXP
                RLF             AARGB3,F                ; rotate next bit for rounding
                CLRF            AARGB3
                MOVF            EXP,W
                BTFSS           _Z
                BCF             _C
                GOTO            SHIFT3232OK

TSHIFT3232      MOVF            EXP,W                   ; shift completed if EXP = 0
                BTFSC           _Z
                GOTO            SHIFT3232OK

SHIFT3232       BCF             _C
                RRF             AARGB0,F                ; right shift by EXP
                RRF             AARGB1,F
                RRF             AARGB2,F
                RRF             AARGB3,F
                DECFSZ          EXP,F
                GOTO            SHIFT3232
```

```
SHIFT3232OK     BTFSC           FPFLAGS,RND
                BTFSS           AARGB3,LSB
                GOTO            INT3232OK
                BTFSS           _C
                GOTO            INT3232OK
                INCF            AARGB3,F
                BTFSC           _Z
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                BTFSC           AARGB0,MSB              ; test for overflow
                GOTO            SETIOV3224

INT3232OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB0,F
                COMF            AARGB1,F
                COMF            AARGB2,F
                COMF            AARGB3,F
                INCF            AARGB3,F
                BTFSC           _Z
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F
                RETLW           0

IRES032         CLRF            AARGB0                  ; integer result equals zero
                CLRF            AARGB1
                CLRF            AARGB2
                CLRF            AARGB3
                RETLW           0

SETIOV32        BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0                  ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 32 bit integer
                MOVLW           0xFF
                MOVWF           AARGB0                  ; SIGN = 0, 0x 7F FF FF FF
                MOVWF           AARGB1                  ; SIGN = 1, 0x 80 00 00 00
                MOVWF           AARGB2
                MOVWF           AARGB3
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG
```

```
;*****************************************************************************************
;*****************************************************************************************

;       Floating Point Multiply

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL    FPM32

;       Output: 32 bit floating point product in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG * BARG

;       Max Timing:     26+23*22+21+21 = 574 clks       RND = 0
;                       26+23*22+21+35 = 588 clks       RND = 1, SAT = 0
;                       26+23*22+21+38 = 591 clks       RND = 1, SAT = 1

;       Min Timing:     6+6 = 12 clks                   AARG * BARG = 0
;                       24+23*11+21+17 = 315 clks

;       PM: 94                                          DM: 14

;-----------------------------------------------------------------------------------------

FPM32           MOVF            AEXP,W                  ; test for zero arguments
                BTFSS           _Z
                MOVF            BEXP,W
                BTFSC           _Z
                GOTO            RES032

M32BNE0         MOVF            AARGB0,W
                XORWF           BARGB0,W
                MOVWF           SIGN                    ; save sign in SIGN

                MOVF            BEXP,W
                ADDWF           EXP,F
                MOVLW           EXPBIAS-1
                BTFSS           _C
                GOTO            MTUN32

                SUBWF           EXP,F
                BTFSC           _C
                GOTO            SETFOV32                ; set multiply overflow flag
                GOTO            MOK32

MTUN32          SUBWF           EXP,F
                BTFSS           _C
                GOTO            SETFUN32

MOK32           MOVF            AARGB0,W
                MOVWF           AARGB3
                MOVF            AARGB1,W
                MOVWF           AARGB4
                MOVF            AARGB2,W
                MOVWF           AARGB5
                BSF             AARGB3,MSB              ; make argument MSB’s explicit
                BSF             BARGB0,MSB
                BCF             _C
                CLRF            AARGB0                  ; clear initial partial product
                CLRF            AARGB1
                CLRF            AARGB2
                MOVLW           D’24’
                MOVWF           TEMP                    ; initialize counter

MLOOP32         BTFSS           AARGB5,LSB              ; test next bit
```

```
                GOTO            MNOADD32

MADD32          MOVF            BARGB2,W
                ADDWF           AARGB2,F
                MOVF            BARGB1,W
                BTFSC           _C
                INCFSZ          BARGB1,W
                ADDWF           AARGB1,F

                MOVF            BARGB0,W
                BTFSC           _C
                INCFSZ          BARGB0,W
                ADDWF           AARGB0,F

MNOADD32        RRF             AARGB0,F
                RRF             AARGB1,F
                RRF             AARGB2,F
                RRF             AARGB3,F
                RRF             AARGB4,F
                RRF             AARGB5,F
                BCF             _C
                DECFSZ          TEMP,F
                GOTO            MLOOP32

                BTFSC           AARGB0,MSB              ; check for postnormalization
                GOTO            MROUND32
                RLF             AARGB3,F
                RLF             AARGB2,F
                RLF             AARGB1,F
                RLF             AARGB0,F
                DECF            EXP,F

MROUND32        BTFSC           FPFLAGS,RND
                BTFSS           AARGB2,LSB
                GOTO            MUL32OK
                BTFSS           AARGB3,MSB
                GOTO            MUL32OK
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

                BTFSS           _Z                      ; has rounding caused carryout?
                GOTO            MUL32OK
                RRF             AARGB0,F                ; if so, right shift
                RRF             AARGB1,F
                RRF             AARGB2,F
                INCF            EXP,F
                BTFSC           _Z                      ; check for overflow
                GOTO            SETFOV32

MUL32OK         BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive

                RETLW           0

SETFOV32        BSF             FPFLAGS,FOV             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                MOVLW           0xFF
                MOVWF           AEXP                    ; saturate to largest floating
                MOVWF           AARGB0                  ; point number = 0x FF 7F FF FF
                MOVWF           AARGB1                  ; modulo the appropriate sign bit
                MOVWF           AARGB2
```

```
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Divide

;       Input:  32 bit floating point dividend in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point divisor in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL    FPD32

;       Output: 32 bit floating point quotient in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG / BARG

;       Max Timing:     43+12+23*36+35+14 = 932 clks    RND = 0
;                       43+12+23*36+35+50 = 968 clks    RND = 1, SAT = 0
;                       43+12+23*36+35+53 = 971 clks    RND = 1, SAT = 1

;       Min Timing:     7+6 = 13 clks

;       PM: 155                                         DM: 14

;-------------------------------------------------------------------------------------------

FPD32           MOVF            BEXP,W                  ; test for divide by zero
                BTFSC           _Z
                GOTO            SETFDZ32

                MOVF            AEXP,W
                BTFSC           _Z
                GOTO            RES032

D32BNE0         MOVF            AARGB0,W
                XORWF           BARGB0,W
                MOVWF           SIGN                    ; save sign in SIGN
                BSF             AARGB0,MSB              ; make argument MSB’s explicit
                BSF             BARGB0,MSB

TALIGN32        CLRF            TEMP                    ; clear align increment
                MOVF            AARGB0,W
                MOVWF           AARGB3                  ; test for alignment
                MOVF            AARGB1,W
                MOVWF           AARGB4
                MOVF            AARGB2,W
                MOVWF           AARGB5

                MOVF            BARGB2,W
                SUBWF           AARGB5,F
                MOVF            BARGB1,W
                BTFSS           _C
                INCFSZ          BARGB1,W

TS1ALIGN32      SUBWF           AARGB4,F
                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W

TS2ALIGN32      SUBWF           AARGB3,F

                CLRF            AARGB3
                CLRF            AARGB4
                CLRF            AARGB5
```

```
                BTFSS           _C
                GOTO            DALIGN32OK

                BCF             _C                      ; align if necessary
                RRF             AARGB0,F
                RRF             AARGB1,F
                RRF             AARGB2,F
                RRF             AARGB3,F
                MOVLW           0x01
                MOVWF           TEMP                    ; save align increment

DALIGN32OK      MOVF            BEXP,W                  ; compare AEXP and BEXP
                SUBWF           EXP,F
                BTFSS           _C
                GOTO            ALTB32

AGEB32          MOVLW           EXPBIAS-1
                ADDWF           TEMP,W
                ADDWF           EXP,F
                BTFSC           _C
                GOTO            SETFOV32
                GOTO            DARGOK32                ; set overflow flag

ALTB32          MOVLW           EXPBIAS-1
                ADDWF           TEMP,W
                ADDWF           EXP,F
                BTFSS           _C
                GOTO            SETFUN32                ; set underflow flag

DARGOK32        MOVLW           D'24'                   ; initialize counter
                MOVWF           TEMPB1

DLOOP32         RLF             AARGB5,F                ; left shift
                RLF             AARGB4,F
                RLF             AARGB3,F
                RLF             AARGB2,F
                RLF             AARGB1,F
                RLF             AARGB0,F
                RLF             TEMP,F

                MOVF            BARGB2,W                ; subtract
                SUBWF           AARGB2,F
                MOVF            BARGB1,W
                BTFSS           _C
                INCFSZ          BARGB1,W
DS132           SUBWF           AARGB1,F

                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W
DS232           SUBWF           AARGB0,F

                RLF             BARGB0,W
                IORWF           TEMP,F

                BTFSS           TEMP,LSB                ; test for restore
                GOTO            DREST32

                BSF             AARGB5,LSB
                GOTO            DOK32

DREST32         MOVF            BARGB2,W                ; restore if necessary
                ADDWF           AARGB2,F
                MOVF            BARGB1,W
                BTFSC           _C
```

```
                INCFSZ          BARGB1,W
DAREST32        ADDWF           AARGB1,F

                MOVF            BARGB0,W
                BTFSC           _C
                INCF            BARGB0,W
                ADDWF           AARGB0,F

                BCF             AARGB5,LSB

DOK32           DECFSZ          TEMPB1,F
                GOTO            DLOOP32

DROUND32        BTFSC           FPFLAGS,RND
                BTFSS           AARGB5,LSB
                GOTO            DIV32OK
                BCF             _C
                RLF             AARGB2,F                ; compute next significant bit
                RLF             AARGB1,F                ; for rounding
                RLF             AARGB0,F
                RLF             TEMP,F

                MOVF            BARGB2,W                ; subtract
                SUBWF           AARGB2,F
                MOVF            BARGB1,W
                BTFSS           _C
                INCFSZ          BARGB1,W
                SUBWF           AARGB1,F

                MOVF            BARGB0,W
                BTFSS           _C
                INCFSZ          BARGB0,W
                SUBWF           AARGB0,F

                RLF             BARGB0,W
                IORWF           TEMP,W
                ANDLW           0x01

                ADDWF           AARGB5,F
                BTFSC           _C
                INCF            AARGB4,F
                BTFSC           _Z
                INCF            AARGB3,F

                BTFSS           _Z                      ; test if rounding caused carryout
                GOTO            DIV32OK
                RRF             AARGB3,F
                RRF             AARGB4,F
                RRF             AARGB5,F
                INCF            EXP,F
                BTFSC           _Z                      ; test for overflow
                GOTO            SETFOV32


DIV32OK         BTFSS           SIGN,MSB
                BCF             AARGB3,MSB              ; clear explicit MSB if positive

                MOVF            AARGB3,W
                MOVWF           AARGB0                  ; move result to AARG
                MOVF            AARGB4,W
                MOVWF           AARGB1
                MOVF            AARGB5,W
                MOVWF           AARGB2

                RETLW           0
```

```
SETFUN32        BSF             FPFLAGS,FUN             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                MOVLW           0x01                    ; saturate to smallest floating
                MOVWF           AEXP                    ; point number = 0x 01 00 00 00
                CLRF            AARGB0                  ; modulo the appropriate sign bit
                CLRF            AARGB1
                CLRF            AARGB2
                RLF             SIGN,F
                RRF             AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

SETFDZ32        BSF             FPFLAGS,FDZ             ; set divide by zero flag
                RETLW           0xFF

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Subtract

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL FPS32

;       Output: 32 bit floating point sum in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     2+251 = 253 clks                RND = 0
;                       2+265 = 267 clks                RND = 1, SAT = 0
;                       2+271 = 273 clks                RND = 1, SAT = 1

;       Min Timing:     2+12 = 14 clks

;       PM: 2+146 = 148                                 DM: 14

;-------------------------------------------------------------------------------------------

FPS32           MOVLW           0x80
                XORWF           BARGB0,F

;*******************************************************************************************

;       Floating Point Add

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL FPA32

;       Output: 32 bit floating point sum in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     31+41+6*7+6+41+90 = 251 clks    RND = 0
;                       31+41+6*7+6+55+90 = 265 clks    RND = 1, SAT = 0
;                       31+41+6*7+6+55+96 = 271 clks    RND = 1, SAT = 1

;       Min Timing:     8+4 = 12 clks

;       PM: 146                                         DM: 14

;-------------------------------------------------------------------------------------------
```

```
FPA32       MOVF        AARGB0,W            ; exclusive or of signs in TEMP
            XORWF       BARGB0,W
            MOVWF       TEMP

            CLRF        AARGB3              ; clear extended byte
            CLRF        BARGB3

            MOVF        AEXP,W              ; use AARG if AEXP >= BEXP
            SUBWF       BEXP,W
            BTFSS       _C
            GOTO        USEA32

            MOVF        BEXP,W              ; use BARG if AEXP < BEXP
            MOVWF       AARGB5              ; therefore, swap AARG and BARG
            MOVF        AEXP,W
            MOVWF       BEXP
            MOVF        AARGB5,W
            MOVWF       AEXP

            MOVF        BARGB0,W
            MOVWF       AARGB5
            MOVF        AARGB0,W
            MOVWF       BARGB0
            MOVF        AARGB5,W
            MOVWF       AARGB0

            MOVF        BARGB1,W
            MOVWF       AARGB5
            MOVF        AARGB1,W
            MOVWF       BARGB1
            MOVF        AARGB5,W
            MOVWF       AARGB1

            MOVF        BARGB2,W
            MOVWF       AARGB5
            MOVF        AARGB2,W
            MOVWF       BARGB2
            MOVF        AARGB5,W
            MOVWF       AARGB2

USEA32      MOVF        BEXP,W              ; return AARG if BARG = 0
            BTFSC       _Z
            RETLW       0x00

            MOVF        AARGB0,W
            MOVWF       SIGN                ; save sign in SIGN
            BSF         AARGB0,MSB          ; make MSB's explicit
            BSF         BARGB0,MSB

            MOVF        BEXP,W              ; compute shift count in BEXP
            SUBWF       AEXP,W
            MOVWF       BEXP
            BTFSC       _Z
            GOTO        ALIGNED32

            MOVLW       8
            SUBWF       BEXP,W
            BTFSS       _C                  ; if BEXP >= 8, do byte shift
            GOTO        ALIGNB32
            MOVWF       BEXP
            MOVF        BARGB2,W            ; keep for postnormalization
            MOVWF       BARGB3
            MOVF        BARGB1,W
            MOVWF       BARGB2
            MOVF        BARGB0,W
            MOVWF       BARGB1
```

```
            CLRF        BARGB0

            MOVLW       8
            SUBWF       BEXP,W
            BTFSS       _C                  ; if BEXP >= 8, do byte shift
            GOTO        ALIGNB32
            MOVWF       BEXP
            MOVF        BARGB2,W            ; keep for postnormalization
            MOVWF       BARGB3
            MOVF        BARGB1,W
            MOVWF       BARGB2
            CLRF        BARGB1

            MOVLW       8
            SUBWF       BEXP,W
            BTFSS       _C                  ; if BEXP >= 8, BARG = 0 relative to AARG
            GOTO        ALIGNB32
            MOVF        SIGN,W
            MOVWF       AARGB0
            RETLW       0x00

ALIGNB32    MOVF        BEXP,W              ; already aligned if BEXP = 0
            BTFSC       _Z
            GOTO        ALIGNED32

ALOOPB32    BCF         _C                  ; right shift by BEXP
            RRF         BARGB0,F
            RRF         BARGB1,F
            RRF         BARGB2,F
            RRF         BARGB3,F
            DECFSZ      BEXP,F
            GOTO        ALOOPB32

ALIGNED32   BTFSS       TEMP,MSB            ; negate if signs opposite
            GOTO        AOK32

            COMF        BARGB3,F
            COMF        BARGB2,F
            COMF        BARGB1,F
            COMF        BARGB0,F
            INCF        BARGB3,F
            BTFSC       _Z
            INCF        BARGB2,F
            BTFSC       _Z
            INCF        BARGB1,F
            BTFSC       _Z
            INCF        BARGB0,F

AOK32
            MOVF        BARGB3,W
            ADDWF       AARGB3,F
            MOVF        BARGB2,W
            BTFSC       _C
            INCFSZ      BARGB2,W
            ADDWF       AARGB2,F
            MOVF        BARGB1,W
            BTFSC       _C
            INCFSZ      BARGB1,W
            ADDWF       AARGB1,F
            MOVF        BARGB0,W
            BTFSC       _C
            INCFSZ      BARGB0,W
            ADDWF       AARGB0,F

            BTFSC       TEMP,MSB
            GOTO        ACOMP32
```

```
                BTFSS           _C
                GOTO            NRMRND4032

                RRF             AARGB0,F                ; shift right and increment EXP
                RRF             AARGB1,F
                RRF             AARGB2,F
                RRF             AARGB3,F
                INCFSZ          AEXP,F
                GOTO            NRMRND4032
                GOTO            SETFOV32

ACOMP32         BTFSC           _C
                GOTO            NRM4032                 ; normalize and fix sign

                COMF            AARGB3,F
                COMF            AARGB2,F                ; negate, toggle sign bit and
                COMF            AARGB1,F                ; then normalize
                COMF            AARGB0,F
                INCF            AARGB3,F
                BTFSC           _Z
                INCF            AARGB2,F
                BTFSC           _Z
                INCF            AARGB1,F
                BTFSC           _Z
                INCF            AARGB0,F

                MOVLW           0x80
                XORWF           SIGN,F
                GOTO            NRM32
```

Please check the Microchip BBS for the latest version of the source code. For BBS access information, see Section 6, Microchip Bulletin Board Service information, page 6-3.

APPENDIX F: PIC17CXXX 32-BIT FLOATING POINT LIBRARY

```
;       RCS Header $Id: fp32.a17 2.8 1996/12/21 20:59:37 F.J.Testa Exp $

;       $Revision: 2.8 $

;       PIC17 32-BIT FLOATING POINT LIBRARY
;
;       Unary operations: both input and output are in AEXP,AARG
;
;       Binary operations: input in AEXP,AARG and BEXP,BARG with output in AEXP,AARG
;
;       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
;       for an error condition specified in FPFLAGS.
;
;       Max timings are worst case cycle counts, while Min timings are non-exception
;       best case cycle counts.
;
;         Routine               Function
;
;       FLO2432   24 bit integer to 32 bit floating point conversion
;       FLO32
;
;               Timing:            RND
;                                0       1
;
;                       0       60      60
;                  SAT
;                       1       67      67
;
;       NRM3232   32 bit normalization of unnormalized 32 bit floating point numbers
;       NRM32
;
;               Timing:            RND
;                                0       1
;
;                       0       48      48
;                  SAT
;                       1       55      55
;
;
;       INT3224         32 bit floating point to 24 bit integer conversion
;       INT32
;
;               Timing:            RND
;                                0       1
;
;                       0       70      79
;                  SAT
;                       1       70      81
;
;
;       FLO3232   32 bit integer to 32 bit floating point conversion
;
;               Timing:            RND
;                                0       1
;
;                       0       75      91
;                  SAT
;                       1       75      97
;
```

```
;       NRM4032   32 bit normalization of unnormalized 40 bit floating point numbers
;
;               Timing:            RND
;                               0       1
;
;                     0       61      77
;                  SAT
;                     1       61      83
;
;
;       INT3232         32 bit floating point to 32 bit integer conversion
;
;               Timing:            RND
;                               0       1
;
;                     0       82      91
;                  SAT
;                     1       82      93
;
;       FPA32           32 bit floating point add
;
;               Timing:            RND
;                               0       1
;
;                     0       160     176
;                  SAT
;                     1       160     182
;
;       FPS32           32 bit floating point subtract
;
;               Timing:            RND
;                               0       1
;
;                     0       161     177
;                  SAT
;                     1       161     183
;
;       FPM32           32 bit floating point multiply
;
;               Timing:            RND
;                               0       1
;
;                     0       100     114
;                  SAT
;                     1       100     122
;
;       FPD32           32 bit floating point divide
;
;               Timing:            RND
;                               0       1
;
;                     0       323     337
;                  SAT
;                     1       323     345
;
;
;*****************************************************************************************
```

```
;*****************************************************************************************

;       32 bit floating point representation

;       EXPONENT        8 bit biased exponent

;                       It is important to note that the use of biased exponents produces
;                       a unique representation of a floating point 0, given by
;                       EXP = HIGHBYTE = MIDBYTE = LOWBYTE = 0x00, with 0 being
;                       the only number with EXP = 0.

;       HIGHBYTE        8 bit most significant byte of fraction in sign-magnitude representation,
;                       with SIGN = MSB, implicit MSB = 1 and radix point to the right of MSB

;       MIDBYTE         8 bit middle significant byte of sign-magnitude fraction

;       LOWBYTE         8 bit least significant byte of sign-magnitude fraction

;       EXPONENT        HIGHBYTE        MIDBYTE         LOWBYTE

;       xxxxxxxx        S.xxxxxxx       xxxxxxxx        xxxxxxxx

;                        |
;                      RADIX
;                      POINT


;*****************************************************************************************
;*****************************************************************************************

;       Integer to float conversion

;       Input:  24 bit 2's complement integer right justified in AARGB0,AARGB1, AARGB2

;       Use:    CALL    FLO2432    or    CALL    FLO32

;       Output: 32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     12+48 = 60 clks                 SAT = 0
;                       12+55 = 67 clks                 SAT = 1

;       Min Timing:     6+24 = 30 clks                  AARG = 0
;                       6+20 = 26 clks

;       PM: 12+115 = 127                                DM: 7

;-----------------------------------------------------------------------------------------

FLO2432
FLO32
                MOVLW           D'23'+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM3232
                CLRF            WREG,F                  ; if < 0, negate, set MSB in SIGN
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
```

```
;*****************************************************************************************

;       Normalization routine

;       Input:  32 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2 with sign in SIGN,MSB.

;       Use:    CALL    NRM3232 or      CALL    NRM32

;       Output: 32 bit normalized floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  NORMALIZE( AARG )

;       Max Timing:     21+19+8 = 48 clks               SAT = 0
;                       21+19+15 = 55 clks              SAT = 1

;       Min Timing:     4+7+7+6 = 24 clks               AARG = 0
;                       3+5+4+8 = 20 clks

;       PM: 115                                         DM: 7

;-----------------------------------------------------------------------------------------

NRM3232
NRM32
                CLRF            TEMP,W                  ; clear exponent decrement
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM3232A

TNIB3232        MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM3232
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1
                SWAPF           AARGB2,W
                ANDLW           0x0F
                ADDWF           AARGB1,F

                SWAPF           AARGB2,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB2

                BSF             TEMP,2                  ; increase decrement by 4

NORM3232        BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN3232
                RLCF            AARGB2,F                ; otherwise, shift left and
                RLCF            AARGB1,F                ; increment decrement
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3232
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
```

```
                GOTO            TNORMUN3232             ; 3 left shifts are required
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F

TNORMUN3232     MOVFP           TEMP,WREG               ; if EXP <= decrement in TEMP,
                CPFSGT          EXP                     ; floating point underflow has
                GOTO            SETFUN32                ; occured
                SUBWF           EXP,F                   ; otherwise, compute EXP

FIXSIGN32       BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive
                RETLW           0

NRM3232A        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                MOVFP           AARGB2,AARGB1
                CLRF            AARGB2,W
                BSF             TEMP,3                  ; increase decrement by 8
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM3232B

TNIB3232A       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM3232A
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1

                BSF             TEMP,2                  ; increase decrement by 4

NORM3232A       BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN3232
                RLCF            AARGB1,F                ; otherwise, shift left and
                RLCF            AARGB0,F                ; increment decrement
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3232
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN3232             ; 3 left shifts are required
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN3232

NRM3232B        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                CLRF            AARGB1,W
                BCF             TEMP,3                  ; increase decrement by 8
                BSF             TEMP,4
                CPFSGT          AARGB0                  ; if highbyte=0, result=0
                GOTO            RES032

TNIB3232B       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
```

```
                GOTO            NORM3232B
                SWAPF           AARGB0,F                ; if so, shift 4 bits

                BSF             TEMP,2                  ; increase decrement by 4

NORM3232B       BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN3232
                RLCF            AARGB0,F                ; otherwise, shift left and
                INCF            TEMP,F                  ; increment decrement
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN3232
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN3232             ; 3 left shifts are required
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN3232

RES032          CLRF            AARGB0,F                ; result equals zero
                CLRF            AARGB1,F
                CLRF            AARGB2,F
                CLRF            AARGB3,F                ; clear extended byte
                CLRF            EXP,F
                RETLW           0

;*******************************************************************************************
;*******************************************************************************************

;       Integer to float conversion

;       Input:  32 bit 2’s complement integer right justified in AARGB0,AARGB1,
;               AARGB2, AARGB3

;       Use:    CALL    FLO3232

;       Output: 32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  FLOAT( AARG )

;       Max Timing:     14+61 = 75 clks                 RND = 0
;                       14+77 = 91 clks                 RND = 1, SAT = 0
;                       14+83 = 97 clks                 RND = 1, SAT = 1

;       Min Timing:     6+32 = 38 clks                  AARG = 0
;                       6+26 = 32 clks

;       PM: 14+178 = 192                                DM: 8

;-------------------------------------------------------------------------------------------

FLO3232         MOVLW           D’31’+EXPBIAS           ; initialize exponent and add bias
                MOVWF           EXP
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BTFSS           AARGB0,MSB              ; test sign
                GOTO            NRM4032
                CLRF            WREG,F                  ; if < 0, negate, set MSB in SIGN
                COMF            AARGB3,F
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB3,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
```

```
;*****************************************************************************************

;       Normalization routine

;       Input:  40 bit unnormalized floating point number in AEXP, AARGB0, AARGB1,
;               AARGB2, AARGB3 with sign in SIGN,MSB.

;       Use:    CALL    NRM4032

;       Output: 32 bit normalized floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  NORMALIZE( AARG )


;       Max Timing:     27+22+8+4 = 61 clks             RND = 0
;                       27+22+24+4 = 77 clks            RND = 1, SAT = 0
;                       27+22+23+11 = 83 clks           RND = 1, SAT = 1

;       Min Timing:     4+8+8+6+6 = 32 clks             AARG = 0
;                       3+5+4+8+6 = 26 clks

;       PM: 178                                         DM: 8

;-----------------------------------------------------------------------------------------

NRM4032         CLRF            TEMP,W                  ; clear exponent decrement
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM4032A

TNIB4032        MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM4032
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1
                SWAPF           AARGB2,W
                ANDLW           0x0F
                ADDWF           AARGB1,F

                SWAPF           AARGB2,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB2
                SWAPF           AARGB3,W
                ANDLW           0x0F
                ADDWF           AARGB2,F

                SWAPF           AARGB3,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB3

                BSF             TEMP,2                  ; increase decrement by 4

NORM4032        BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN4032
                RLCF            AARGB3,F                ; otherwise, shift left and
                RLCF            AARGB2,F                ; increment decrement
                RLCF            AARGB1,F
```

```
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN4032
                RLCF            AARGB3,F
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN4032             ; 3 left shifts are required
                RLCF            AARGB3,F
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F

TNORMUN4032     MOVFP           TEMP,WREG               ; if EXP <= decrement in TEMP,
                CPFSGT          EXP                     ; floating point underflow has
                GOTO            SETFUN32                ; occured
                SUBWF           EXP,F                   ; otherwise, compute EXP

NRMRND4032
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB3,MSB              ; is NSB > 0x80?
                GOTO            FIXSIGN32
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB3                  ; if NSB = 0x80, select even
                RRCF            AARGB2,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                BTFSS           _C                      ; has rounding caused carryout?
                GOTO            FIXSIGN32
                RRCF            AARGB0,F                ; if so, right shift
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                INFSNZ          EXP,F                   ; test for floating point overflow
                GOTO            SETFOV32
                GOTO            FIXSIGN32

NRM4032A        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                MOVFP           AARGB2,AARGB1
                MOVFP           AARGB3,AARGB2
                CLRF            AARGB3,W
                BSF             TEMP,3                  ; increase decrement by 8
                CPFSGT          AARGB0                  ; test if highbyte=0
                GOTO            NRM4032B

TNIB4032A       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM4032A
                SWAPF           AARGB0,F                ; if so, shift 4 bits
                SWAPF           AARGB1,W
                ANDLW           0x0F
                ADDWF           AARGB0,F

                SWAPF           AARGB1,W
                ANDLW           0xF0
                MOVPF           WREG,AARGB1
                SWAPF           AARGB2,W
                ANDLW           0x0F
```

```
            ADDWF       AARGB1,F

            SWAPF       AARGB2,W
            ANDLW       0xF0
            MOVPF       WREG,AARGB2

            BSF         TEMP,2              ; increase decrement by 4

NORM4032A   BCF         _C                  ; clear carry bit

            BTFSC       AARGB0,MSB          ; if MSB=1, normalization done
            GOTO        TNORMUN4032
            RLCF        AARGB2,F            ; otherwise, shift left and
            RLCF        AARGB1,F            ; increment decrement
            RLCF        AARGB0,F
            INCF        TEMP,F
            BTFSC       AARGB0,MSB
            GOTO        TNORMUN4032
            RLCF        AARGB2,F
            RLCF        AARGB1,F
            RLCF        AARGB0,F
            INCF        TEMP,F
            BTFSC       AARGB0,MSB          ; since highnibble != 0, at most
            GOTO        TNORMUN4032         ; 3 left shifts are required
            RLCF        AARGB2,F
            RLCF        AARGB1,F
            RLCF        AARGB0,F
            INCF        TEMP,F
            GOTO        TNORMUN4032

NRM4032B    MOVFP       AARGB1,AARGB0       ; shift 8 bits by move
            MOVFP       AARGB2,AARGB1
            CLRF        AARGB2,W
            BCF         TEMP,3              ; increase decrement by 8
            BSF         TEMP,4
            CPFSGT      AARGB0              ; if highbyte=0, result=0
            GOTO        NRM4032C

TNIB4032B   MOVLW       0xF0                ; test if highnibble=0
            ANDWF       AARGB0,W
            TSTFSZ      WREG
            GOTO        NORM4032B

            SWAPF       AARGB0,F            ; if so, shift 4 bits
            SWAPF       AARGB1,W
            ANDLW       0x0F
            ADDWF       AARGB0,F

            SWAPF       AARGB1,W
            ANDLW       0xF0
            MOVPF       WREG,AARGB1

            BSF         TEMP,2              ; increase decrement by 4

NORM4032B   BCF         _C                  ; clear carry bit

            BTFSC       AARGB0,MSB          ; if MSB=1, normalization done
            GOTO        TNORMUN4032
            RLCF        AARGB1,F            ; otherwise, shift left and
            RLCF        AARGB0,F
            INCF        TEMP,F              ; increment decrement
            BTFSC       AARGB0,MSB
            GOTO        TNORMUN4032
            RLCF        AARGB1,F
            RLCF        AARGB0,F
            INCF        TEMP,F
```

```
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN4032             ; 3 left shifts are required
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN4032

NRM4032C        MOVFP           AARGB1,AARGB0           ; shift 8 bits by move
                CLRF            AARGB1,W
                BSF             TEMP,3                  ; increase decrement by 8
                CPFSGT          AARGB0                  ; if highbyte=0, result=0
                GOTO            RES032

TNIB4032C       MOVLW           0xF0                    ; test if highnibble=0
                ANDWF           AARGB0,W
                TSTFSZ          WREG
                GOTO            NORM4032C
                SWAPF           AARGB0,F                ; if so, shift 4 bits

                BSF             TEMP,2                  ; increase decrement by 4

NORM4032C       BCF             _C                      ; clear carry bit

                BTFSC           AARGB0,MSB              ; if MSB=1, normalization done
                GOTO            TNORMUN4032
                RLCF            AARGB0,F                ; otherwise, shift left and
                INCF            TEMP,F                  ; increment decrement
                BTFSC           AARGB0,MSB
                GOTO            TNORMUN4032
                RLCF            AARGB0,F
                INCF            TEMP,F
                BTFSC           AARGB0,MSB              ; since highnibble != 0, at most
                GOTO            TNORMUN4032             ; 3 left shifts are required
                RLCF            AARGB0,F
                INCF            TEMP,F
                GOTO            TNORMUN4032


;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Use:    CALL    INT3224         or      CALL    INT32

;       Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     10+45+15 = 70 clks              RND = 0
;                       10+45+24 = 79 clks              RND = 1, SAT = 0
;                       10+45+26 = 81 clks              RND = 1, SAT = 1

;       Min Timing:     4 clks

;       PM: 183                                         DM: 8

;-------------------------------------------------------------------------------------------

INT3224
INT32
                CLRF            AARGB3,W
                CPFSGT          EXP                     ; test for zero argument
                RETLW           0x00
```

```
            MOVPF       AARGB0,SIGN         ; save sign in SIGN
            BSF         AARGB0,MSB          ; make MSB explicit

            MOVLW       EXPBIAS+D'23'       ; remove bias+23 from EXP
            SUBWF       EXP,W

            BTFSS       WREG,MSB            ; if >= 23, integer overflow
            GOTO        SETIOV3224          ; will occur
            NEGW        EXP,F

            MOVLW       7                   ; do byte shift if EXP >= 8
            CPFSGT      EXP
            GOTO        SNIB3224
            SUBWF       EXP,F               ; EXP = EXP - 7
            MOVFP       AARGB2,AARGB3       ; save for rounding
            MOVFP       AARGB1,AARGB2
            MOVFP       AARGB0,AARGB1
            CLRF        AARGB0,F
            DCFSNZ      EXP,F               ; EXP = EXP - 1
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0

            CPFSGT      EXP                 ; do another byte shift if EXP >= 8
            GOTO        SNIB3224A
            SUBWF       EXP,F               ; EXP = EXP - 7
            MOVFP       AARGB2,AARGB3       ; save for rounding
            MOVFP       AARGB1,AARGB2
            CLRF        AARGB1,F
            DCFSNZ      EXP,F               ; EXP = EXP - 1
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0

            CPFSGT      EXP                 ; do another byte shift if EXP >= 8
            GOTO        SNIB3224B
            SUBWF       EXP,F               ; EXP = EXP - 7
            MOVFP       AARGB2,AARGB3       ; save for rounding
            CLRF        AARGB2,F
            DCFSNZ      EXP,F               ; EXP = EXP - 1
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0

SNIB3224C   MOVLW       3                   ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT3224C
            SWAPF       AARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB3
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0

SHIFT3224C  BCF         _C                  ; at most 3 right shifts are required
            RRCF        AARGB3,F            ; right shift by EXP
            DCFSNZ      EXP,F
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            DCFSNZ      EXP,F
            GOTO        SHIFT3224OK         ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            GOTO        SHIFT3224OK

SNIB3224B   MOVLW       3                   ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT3224B
            SUBWF       EXP,F               ; EXP = EXP - 3
            SWAPF       AARGB2,W
            MOVPF       WREG,AARGB3         ; save for rounding
            ANDLW       0x0F
            MOVPF       WREG,AARGB2
```

```
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0

SHIFT3224B      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB2,F                ; right shift by EXP
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                GOTO            SHIFT3224OK

SNIB3224A       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT3224A
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB2,W
                MOVPF           WREG,AARGB3             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0

SHIFT3224A      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB1,F                ; right shift by EXP
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                GOTO            SHIFT3224OK

SNIB3224        MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT3224
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB2,W
                MOVPF           WREG,AARGB3             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
```

```
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1

                SWAPF           AARGB0,W
                ANDLW           0xF0
                ADDWF           AARGB1,F

                SWAPF           AARGB0,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB0
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0

SHIFT3224       BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB0,F                ; right shift by EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3224OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F

SHIFT3224OK
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB3,MSB              ; is NSB > 0x80?
                GOTO            INT3224OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB3                  ; if NSB = 0x80, select even
                RRCF            AARGB2,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTFSC           AARGB0,MSB
                GOTO            SETIOV3224

INT3224OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB2,F
                CLRF            WREG,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                RETLW           0

SETIOV3224      BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG
```

```
                CLRF            AARGB0,F                ; saturate to largest two's
                BTFSS           SIGN,MSB                ; complement 24 bit integer
                SETF            AARGB0,F                ; SIGN = 0, 0x 7F FF FF
                MOVPF           AARGB0,AARGB1           ; SIGN = 1, 0x 80 00 00
                MOVPF           AARGB0,AARGB2
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Float to integer conversion

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2

;       Use:    CALL    INT3232

;       Output: 32 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2,
;               AARGB3

;       Result: AARG  <--  INT( AARG )

;       Max Timing:     11+54+17 = 82 clks              RND = 0
;                       11+54+26 = 91 clks              RND = 1, SAT = 0
;                       11+54+28 = 93 clks              RND = 1, SAT = 1

;       Min Timing:     4 clks

;       PM: 249                                         DM: 9

;-------------------------------------------------------------------------------------------

INT3232
                CLRF            AARGB3,W
                CPFSGT          EXP                     ; test for zero argument
                RETLW           0x00
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BSF             AARGB0,MSB              ; make MSB explicit

                CLRF            AARGB4,F

                MOVLW           EXPBIAS+D'31'           ; remove bias from EXP
                SUBWF           EXP,W

                BTFSS           WREG,MSB                ; if >= 31, integer overflow
                GOTO            SETIOV3232              ; will occur
                NEGW            EXP,F

                MOVLW           7                       ; do byte shift if EXP >= 8
                CPFSGT          EXP
                GOTO            SNIB3232
                SUBWF           EXP,F                   ; EXP = EXP - 7
                MOVFP           AARGB3,AARGB4           ; save for rounding
                MOVFP           AARGB2,AARGB3
                MOVFP           AARGB1,AARGB2
                MOVFP           AARGB0,AARGB1
                CLRF            AARGB0,F
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0

                CPFSGT          EXP                     ; do another byte shift if EXP >= 8
                GOTO            SNIB3232A
                SUBWF           EXP,F                   ; EXP = EXP - 7
                MOVFP           AARGB3,AARGB4           ; save for rounding
                MOVFP           AARGB2,AARGB3
```

```
            MOVFP       AARGB1,AARGB2
            CLRF        AARGB1,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0

            CPFSGT      EXP                     ; do another byte shift if EXP >= 8
            GOTO        SNIB3232B
            SUBWF       EXP,F                   ; EXP = EXP - 7
            MOVFP       AARGB3,AARGB4           ; save for rounding
            MOVFP       AARGB2,AARGB3
            CLRF        AARGB2,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0

            CPFSGT      EXP                     ; do another byte shift if EXP >= 8
            GOTO        SNIB3232C
            SUBWF       EXP,F                   ; EXP = EXP - 7
            MOVFP       AARGB3,AARGB4           ; save for rounding
            CLRF        AARGB3,F
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0

SNIB3232D   MOVLW       3                       ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT3232D
            SWAPF       AARGB4,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB4
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0

SHIFT3232D  BCF         _C                      ; at most 3 right shifts are required
            RRCF        AARGB4,F                ; right shift by EXP
            DCFSNZ      EXP,F
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB4,F
            DCFSNZ      EXP,F
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB4,F
            GOTO        SHIFT3232OK

SNIB3232C   MOVLW       3                       ; do nibble shift if EXP >= 4
            CPFSGT      EXP
            GOTO        SHIFT3232C
            SUBWF       EXP,F                   ; EXP = EXP - 3
            SWAPF       AARGB3,W
            MOVPF       WREG,AARGB4             ; save for rounding
            ANDLW       0x0F
            MOVPF       WREG,AARGB3
            DCFSNZ      EXP,F                   ; EXP = EXP - 1
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0

SHIFT3232C  BCF         _C                      ; at most 3 right shifts are required
            RRCF        AARGB3,F                ; right shift by EXP
            RRCF        AARGB4,F
            DCFSNZ      EXP,F
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            RRCF        AARGB4,F
            DCFSNZ      EXP,F
            GOTO        SHIFT3232OK             ; shift completed if EXP = 0
            BCF         _C
            RRCF        AARGB3,F
            RRCF        AARGB4,F
```

```
                GOTO            SHIFT3232OK

SNIB3232B       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT3232B
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB3,W
                MOVPF           WREG,AARGB4             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB3

                SWAPF           AARGB2,W
                ANDLW           0xF0
                ADDWF           AARGB3,F

                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0

SHIFT3232B      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB2,F                ; right shift by EXP
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                GOTO            SHIFT3232OK

SNIB3232A       MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT3232A
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB3,W
                MOVPF           WREG,AARGB4             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB3

                SWAPF           AARGB2,W
                ANDLW           0xF0
                ADDWF           AARGB3,F

                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
```

```
SHIFT3232A      BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB1,F                ; right shift by EXP
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                GOTO            SHIFT3232OK

SNIB3232        MOVLW           3                       ; do nibble shift if EXP >= 4
                CPFSGT          EXP
                GOTO            SHIFT3232
                SUBWF           EXP,F                   ; EXP = EXP - 3
                SWAPF           AARGB3,W
                MOVPF           WREG,AARGB4             ; save for rounding
                ANDLW           0x0F
                MOVPF           WREG,AARGB3

                SWAPF           AARGB2,W
                ANDLW           0xF0
                ADDWF           AARGB3,F

                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2

                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F

                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1

                SWAPF           AARGB0,W
                ANDLW           0xF0
                ADDWF           AARGB1,F

                SWAPF           AARGB0,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB0
                DCFSNZ          EXP,F                   ; EXP = EXP - 1
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0

SHIFT3232       BCF             _C                      ; at most 3 right shifts are required
                RRCF            AARGB0,F                ; right shift by EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
```

```
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F
                DCFSNZ          EXP,F
                GOTO            SHIFT3232OK             ; shift completed if EXP = 0
                BCF             _C
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                RRCF            AARGB4,F

SHIFT3232OK
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB4,MSB              ; is NSB > 0x80?
                GOTO            INT3232OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB4                  ; if NSB = 0x80, select even
                RRCF            AARGB3,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB3,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTFSC           AARGB0,MSB
                GOTO            SETIOV3232

INT3232OK       BTFSS           SIGN,MSB                ; if sign bit set, negate
                RETLW           0
                COMF            AARGB3,F
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB3,F
                CLRF            WREG,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                RETLW           0

SETIOV3232      BSF             FPFLAGS,IOV             ; set integer overflow flag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                CLRF            AARGB0,F                ; saturate to largest two’s
                BTFSS           SIGN,MSB                ; complement 32 bit integer
                SETF            AARGB0,F                ; SIGN = 0, 0x 7F FF FF FF
                MOVPF           AARGB0,AARGB1           ; SIGN = 1, 0x 80 00 00 00
                MOVPF           AARGB0,AARGB2
                MOVPF           AARGB0,AARGB3
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Multiply

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL    FPM32
```

```
;       Output: 32 bit floating point product in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG * BARG

;       Max Timing:     19+73+8 = 100 clks                RND = 0, SAT = 0
;                       19+73+22 = 114 clks               RND = 1, SAT = 0
;                       19+73+30 = 122 clks               RND = 1, SAT = 1

;       Min Timing:     5+6 = 11 clks                     AARG * BARG = 0

;       PM: 123                                           DM: 15


;-----------------------------------------------------------------------------------------

FPM32           CLRF            AARGB3,W                ; test for zero arguments
                CPFSEQ          BEXP
                CPFSGT          AEXP
                GOTO            RES032

M32BNE0         MOVFP           AARGB0,WREG
                XORWF           BARGB0,W
                MOVPF           WREG,SIGN               ; save sign in SIGN

                MOVFP           BEXP,WREG
                ADDWF           EXP,F
                MOVLW           EXPBIAS-1
                BTFSS           _C
                GOTO            MTUN32

                SUBWF           EXP,F                   ; remove bias and overflow test
                BTFSC           _C
                GOTO            SETFOV32
                GOTO            MOK32

MTUN32          SUBWF           EXP,F                   ; remove bias and underflow test
                BTFSS           _C
                GOTO            SETFUN32

MOK32           BSF             AARGB0,MSB              ; make argument MSB’s explicit
                BSF             BARGB0,MSB

                MOVPF           AARGB0,TEMPB0           ; multiply mantissas
                MOVPF           AARGB1,TEMPB1
                MOVPF           AARGB2,TEMPB2

                MOVFP           AARGB2,WREG
                MULWF           BARGB2
                MOVPF           PRODH,AARGB4

                MOVFP           AARGB1,WREG
                MULWF           BARGB1
                MOVPF           PRODH,AARGB2
                MOVPF           PRODL,AARGB3

                MULWF           BARGB2
                MOVPF           PRODL,WREG
                ADDWF           AARGB4,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB3,F
                CLRF            WREG,F
                ADDWFC          AARGB2,F

                MOVFP           TEMPB2,WREG
                MULWF           BARGB1
                MOVPF           PRODL,WREG
```

```
            ADDWF       AARGB4,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB3,F
            CLRF        WREG,F
            ADDWFC      AARGB2,F

            MOVFP       AARGB0,WREG
            MULWF       BARGB2
            MOVPF       PRODL,WREG
            ADDWF       AARGB3,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB2,F
            MOVFP       AARGB0,WREG
            MULWF       BARGB1
            CLRF        AARGB1,W
            ADDWFC      AARGB1,F
            MOVPF       PRODL,WREG
            ADDWF       AARGB2,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB1,F

            MOVFP       TEMPB2,WREG
            MULWF       BARGB0
            MOVPF       PRODL,WREG
            ADDWF       AARGB3,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB2,F
            CLRF        AARGB0,W
            ADDWFC      AARGB1,F
            ADDWFC      AARGB0,F

            MOVFP       TEMPB1,WREG
            MULWF       BARGB0
            MOVPF       PRODL,WREG
            ADDWF       AARGB2,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB1,F
            CLRF        WREG,F
            ADDWFC      AARGB0,F

            MOVFP       TEMPB0,WREG
            MULWF       BARGB0
            MOVPF       PRODL,WREG
            ADDWF       AARGB1,F
            MOVPF       PRODH,WREG
            ADDWFC      AARGB0,F

            BTFSC       AARGB0,MSB          ; check for postnormalization
            GOTO        MROUND32
            RLCF        AARGB3,F
            RLCF        AARGB2,F
            RLCF        AARGB1,F
            RLCF        AARGB0,F
            DECF        EXP,F
            BTFSC       _Z
            GOTO        SETFUN32

MROUND32
            BTFSC       FPFLAGS,RND         ; is rounding enabled?
            BTFSS       AARGB3,MSB          ; is NSB > 0x80?
            GOTO        MUL32OK
            BSF         _C                  ; set carry for rounding
            MOVLW       0x80
            CPFSGT      AARGB3              ; if NSB = 0x80, select even
            RRCF        AARGB2,W            ; using lsb in carry
            CLRF        WREG,F
```

```
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                BTFSS           _C                      ; has rounding caused carryout?
                GOTO            MUL32OK
                RRCF            AARGB0,F                ; if so, right shift
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                INFSNZ          EXP,F                   ; test for floating point overflow
                GOTO            SETFOV32

MUL32OK         BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive

                RETLW           0

SETFOV32        BSF             FPFLAGS,FOV             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                SETF            AEXP,F                  ; saturate to largest floating
                SETF            AARGB0,F                ; point number = 0x FF 7F FF FF
                SETF            AARGB1,F                ; modulo the appropriate sign bit
                SETF            AARGB2,F
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG


;*******************************************************************************************
;*******************************************************************************************

;       Floating Point Divide

;       Input:  32 bit floating point dividend in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point divisor in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL    FPD32

;       Output: 32 bit floating point quotient in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG / BARG

;       Max Timing:     10+34+69+80+111+11+8 = 323 clks  RND = 0
;                       10+34+69+80+111+11+22 = 337 clks RND = 1, SAT = 0
;                       10+34+69+80+111+11+30 = 345 clks RND = 1, SAT = 1

;       Min Timing:     6+6 = 12 clks                    AARG = 0

;       PM: 350+257 = 607                                DM: 20

;        In addition to those registers defined in MATH17.INC, this routine uses
;        TBLPTRL and TBLPTRH without saving and restoring.

;-------------------------------------------------------------------------------------------

FPD32SEED        macro

;       Timing:     34 clks

;        PM: 32+257 = 289

;        generation of F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 8 explicit bits of BARG as a pointer
;        and the remaining 7 explicit bits as the argument to linear interpolation.
```

```
                MOVLW               HIGH (IBTBL256I)        ; access table for F0
                MOVWF               TBLPTRH
                RLCF                BARGB1,W
                RLCF                BARGB0,W
                ADDLW               LOW (IBTBL256I)
                MOVWF               TBLPTRL
                BTFSC               _C
                INCF                TBLPTRH,F
                TABLRD              0,1,TEMPB0
                TLRD                1,TEMPB0
                TABLRD              0,0,TEMPB1
                TLRD                0,AARGB5

                MOVFP               AARGB5,WREG             ; calculate difference
                SUBWF               TEMPB1,W
                MOVWF               AARGB5

                BCF                 _C                      ; interpolate
                RLCF                BARGB2,W
                MULWF               AARGB5
                MOVPF               PRODH,TBLPTRH
                RLCF                BARGB1,W
                MULWF               AARGB5
                MOVPF               PRODL,WREG
                ADDWF               TBLPTRH,F
                BTFSC               _C
                INCF                PRODH,F

                CLRF                TEMPB2,F
                MOVFP               TBLPTRH,WREG
                SUBWF               TEMPB2,F
                MOVPF               PRODH,WREG
                SUBWFB              TEMPB1,F
                CLRF                WREG,F
                SUBWFB              TEMPB0,F                ; F0

                endm

;-----------------------------------------------------------------------------------------------

FPD32SEEDS1        macro

;       Timing:     51 clks

;        PM: 49+129 = 178

;        generation of F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 7 explicit bits of BARG as a pointer
;        and the remaining 16 explicit bits as the argument to linear interpolation.


                MOVLW               HIGH (IBTBL128I)       ; access table for F0
                MOVWF               TBLPTRH
                MOVFP               BARGB0,WREG
                ANDLW               0x7F
                ADDLW               LOW (IBTBL128I)
                MOVWF               TBLPTRL
                BTFSC               _C
                INCF                TBLPTRH,F
                TABLRD              0,1,TEMPB0
                TLRD                1,TEMPB0
                TABLRD              0,0,TEMPB1
                TLRD                1,AARGB4
                TLRD                0,AARGB5
```

```
                MOVFP               AARGB5,WREG             ; calculate difference
                SUBWF               TEMPB1,W
                MOVWF               AARGB5
                MOVFP               AARGB4,WREG
                SUBWFB              TEMPB0,W
                MOVWF               AARGB4

                MOVPF               AARGB5,TEMPB2

                MOVFP               AARGB5,WREG
                MULWF               BARGB2
                MOVPF               PRODH,TBLPTRH

                MOVFP               AARGB4,WREG
                MULWF               BARGB1
                MOVPF               PRODH,AARGB4
                MOVPF               PRODL,AARGB5

                MULWF               BARGB2
                MOVPF               PRODL,WREG
                ADDWF               TBLPTRH,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB5,F
                CLRF                WREG,F
                ADDWFC              AARGB4,F

                MOVFP               TEMPB2,WREG
                MULWF               BARGB1
                MOVPF               PRODL,WREG
                ADDWF               TBLPTRH,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB5,F
                CLRF                WREG,F
                ADDWFC              AARGB4,F

                CLRF                TEMPB2,F
                MOVFP               TBLPTRH,WREG
                SUBWF               TEMPB2,F
                MOVFP               AARGB5,WREG
                SUBWFB              TEMPB1,F
                MOVFP               AARGB4,WREG
                SUBWFB              TEMPB0,F                ; F0

                endm

;-----------------------------------------------------------------------------------------------

FPD32SEEDS        macro

;       Timing:     75 clks

;        PM: 73+65 = 138

;        generation of F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 6 explicit bits of BARG as a pointer
;        and the remaining 17 explicit bits as the argument to linear interpolation.


                MOVLW               HIGH (IBTBL64I)         ; access table for F0
                MOVWF               TBLPTRH
                RRCF                BARGB0,W
                ANDLW               0x3F
                ADDLW               LOW (IBTBL64I)
                MOVWF               TBLPTRL
                BTFSC               _C
```

```
INCF            TBLPTRH,F
TABLRD          0,1,TEMPB0
TLRD            1,TEMPB0
TABLRD          0,0,TEMPB1
TLRD            1,AARGB4
TLRD            0,AARGB5

MOVFP           AARGB5,WREG             ; calculate difference
SUBWF           TEMPB1,W
MOVWF           AARGB5
MOVFP           AARGB4,WREG
SUBWFB          TEMPB0,W
MOVWF           AARGB4

MOVPF           AARGB4,TBLPTRH
MOVPF           AARGB5,TBLPTRL

MOVFP           BARGB2,WREG
MULWF           AARGB5
MOVPF           PRODH,TMR0H
MOVPF           PRODL,TMR0L

MOVFP           BARGB1,WREG
MULWF           AARGB4
MOVPF           PRODH,AARGB5
MOVPF           PRODL,TEMPB2

MULWF           TBLPTRL
MOVPF           PRODL,WREG
ADDWF           TMR0H,F
MOVPF           PRODH,WREG
ADDWFC          TEMPB2,F
CLRF            WREG,F
ADDWFC          AARGB5,F

MOVFP           BARGB2,WREG
MULWF           TBLPTRH
MOVPF           PRODL,WREG
ADDWF           TMR0H,F
MOVPF           PRODH,WREG
ADDWFC          TEMPB2,F
CLRF            WREG,F
ADDWFC          AARGB5,F

MOVFP           BARGB0,WREG
ANDLW           0x01
MULWF           TBLPTRL
MOVPF           PRODL,WREG
ADDWF           TEMPB2,F
MOVPF           PRODH,WREG
ADDWFC          AARGB5,F
MOVFP           BARGB0,WREG
ANDLW           0x01
MULWF           TBLPTRH
CLRF            AARGB4,W
ADDWFC          AARGB4,F
MOVPF           PRODL,WREG
ADDWF           AARGB5,F
MOVPF           PRODH,WREG
ADDWFC          AARGB4,F

BCF             _C
RRCF            AARGB4,F
RRCF            AARGB5,F
RRCF            TEMPB2,F
RRCF            TMR0H,F
```

```
                MOVFP               TEMPB2,TBLPTRH
                CLRF                TEMPB2,F
                MOVFP               TMR0H,WREG
                SUBWF               TEMPB2,F
                MOVFP               TBLPTRH,WREG
                SUBWFB              TEMPB1,F
                MOVFP               AARGB5,WREG
                SUBWFB              TEMPB0,F                ; F0

                endm

;-----------------------------------------------------------------------------------------------

FPD32           CLRF                TEMPB3,W                ; clear exponent modification
                CPFSGT              BEXP                    ; test for divide by zero
                GOTO                SETFDZ32

                CPFSGT              AEXP
                GOTO                RES032

D32BNE0         MOVFP               AARGB0,WREG
                XORWF               BARGB0,W
                MOVPF               WREG,SIGN               ; save sign in SIGN

                BSF                 AARGB0,MSB              ; make argument MSB's explicit
                BSF                 BARGB0,MSB

FPD32SEED                                                   ; generation of F0

                MOVPF               AARGB0,AARGB5           ; A0 = F0 * A
                MOVPF               AARGB1,TBLPTRH
                MOVPF               AARGB2,TBLPTRL

                MOVFP               AARGB2,WREG
                MULWF               TEMPB2
                MOVPF               PRODH,AARGB4

                MOVFP               AARGB1,WREG
                MULWF               TEMPB1
                MOVPF               PRODH,AARGB2
                MOVPF               PRODL,AARGB3

                MULWF               TEMPB2
                MOVPF               PRODL,WREG
                ADDWF               AARGB4,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB3,F
                CLRF                WREG,F
                ADDWFC              AARGB2,F

                MOVFP               TBLPTRL,WREG
                MULWF               TEMPB1
                MOVPF               PRODL,WREG
                ADDWF               AARGB4,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB3,F
                CLRF                WREG,F
                ADDWFC              AARGB2,F

                MOVFP               AARGB0,WREG
                MULWF               TEMPB2
                MOVPF               PRODL,WREG
                ADDWF               AARGB3,F
                MOVPF               PRODH,WREG
                ADDWFC              AARGB2,F
```

```
                MOVFP           AARGB0,WREG
                MULWF           TEMPB1
                CLRF            AARGB1,W
                ADDWFC          AARGB1,F
                MOVPF           PRODL,WREG
                ADDWF           AARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB1,F

                MOVFP           TBLPTRL,WREG
                MULWF           TEMPB0
                MOVPF           PRODL,WREG
                ADDWF           AARGB3,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB2,F
                CLRF            AARGB0,W
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                MOVFP           TBLPTRH,WREG
                MULWF           TEMPB0
                MOVPF           PRODL,WREG
                ADDWF           AARGB2,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB1,F
                CLRF            WREG,F
                ADDWFC          AARGB0,F

                MOVFP           AARGB5,WREG
                MULWF           TEMPB0
                MOVPF           PRODL,WREG
                ADDWF           AARGB1,F
                MOVPF           PRODH,WREG
                ADDWFC          AARGB0,F

                BTFSC           AARGB0,MSB
                GOTO            DAOK32
                RLCF            AARGB3,F
                RLCF            AARGB2,F
                RLCF            AARGB1,F
                RLCF            AARGB0,F
                DECF            TEMPB3,F

DAOK32
                MOVFP           BARGB0,AARGB5           ; B0 = F0 * B
                MOVFP           BARGB1,TBLPTRH
                MOVFP           BARGB2,TBLPTRL

                MOVFP           BARGB2,WREG
                MULWF           TEMPB2
                MOVPF           PRODH,AARGB4

                MOVFP           BARGB1,WREG
                MULWF           TEMPB1
                MOVPF           PRODH,BARGB2
                MOVPF           PRODL,BARGB3

                MULWF           TEMPB2
                MOVPF           PRODL,WREG
                ADDWF           AARGB4,F
                MOVPF           PRODH,WREG
                ADDWFC          BARGB3,F
                CLRF            WREG,F
                ADDWFC          BARGB2,F

                MOVFP           TBLPTRL,WREG
```

```
            MULWF           TEMPB1
            MOVPF           PRODL,WREG
            ADDWF           AARGB4,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB3,F
            CLRF            WREG,F
            ADDWFC          BARGB2,F

            MOVFP           BARGB0,WREG
            MULWF           TEMPB2
            MOVPF           PRODL,WREG
            ADDWF           BARGB3,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB2,F
            MOVFP           BARGB0,WREG
            MULWF           TEMPB1
            CLRF            BARGB1,W
            ADDWFC          BARGB1,F
            MOVPF           PRODL,WREG
            ADDWF           BARGB2,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB1,F

            MOVFP           TBLPTRL,WREG
            MULWF           TEMPB0
            MOVPF           PRODL,WREG
            ADDWF           BARGB3,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB2,F
            CLRF            BARGB0,W
            ADDWFC          BARGB1,F
            ADDWFC          BARGB0,F

            MOVFP           TBLPTRH,WREG
            MULWF           TEMPB0
            MOVPF           PRODL,WREG
            ADDWF           BARGB2,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB1,F
            CLRF            WREG,F
            ADDWFC          BARGB0,F

            MOVFP           AARGB5,WREG
            MULWF           TEMPB0
            MOVPF           PRODL,WREG
            ADDWF           BARGB1,F
            MOVPF           PRODH,WREG
            ADDWFC          BARGB0,F

            BTFSS           BARGB0,MSB
            BTFSC           BARGB0,MSB-1
            GOTO            DBOK32
            RLCF            AARGB4,F
            RLCF            BARGB3,F
            RLCF            BARGB2,F
            RLCF            BARGB1,F
            RLCF            BARGB0,F
            INCF            TEMPB3,F

DBOK32
            COMF            BARGB3,F
            COMF            BARGB2,F
            COMF            BARGB1,F            ; F1 = 2 - B0
            COMF            BARGB0,F
            INCF            BARGB3,F
            CLRF            WREG,F
```

```
ADDWFC          BARGB2,F
ADDWFC          BARGB1,F
ADDWFC          BARGB0,F

MOVPF           AARGB0,TEMPB0           ; A1 = F1 * A0
MOVPF           AARGB1,TEMPB1
MOVPF           AARGB2,TEMPB2
MOVPF           AARGB3,TBLPTRL

MOVFP           AARGB2,WREG
MULWF           BARGB2
MOVPF           PRODH,AARGB4

MOVFP           AARGB1,WREG
MULWF           BARGB3
MOVPF           PRODH,WREG
ADDWF           AARGB4,F
MOVFP           AARGB1,WREG
MULWF           BARGB2
CLRF            AARGB3,W
ADDWFC          AARGB3,F
MOVPF           PRODL,WREG
ADDWF           AARGB4,F
MOVPF           PRODH,WREG
ADDWFC          AARGB3,F

MOVFP           TBLPTRL,WREG
MULWF           BARGB1
MOVPF           PRODH,WREG
ADDWF           AARGB4,F
CLRF            AARGB2,W
ADDWFC          AARGB3,F
ADDWFC          AARGB2,F

MOVFP           TEMPB2,WREG
MULWF           BARGB1
MOVPF           PRODL,WREG
ADDWF           AARGB4,F
MOVPF           PRODH,WREG
ADDWFC          AARGB3,F
CLRF            WREG,F
ADDWFC          AARGB2,F

MOVFP           TEMPB1,WREG
MULWF           BARGB1
MOVPF           PRODL,WREG
ADDWF           AARGB3,F
MOVPF           PRODH,WREG
ADDWFC          AARGB2,F

MOVFP           AARGB0,WREG
MULWF           BARGB2
MOVPF           PRODL,WREG
ADDWF           AARGB3,F
MOVPF           PRODH,WREG
ADDWFC          AARGB2,F
MOVFP           AARGB0,WREG
MULWF           BARGB1
CLRF            AARGB1,W
ADDWFC          AARGB1,F
MOVPF           PRODL,WREG
ADDWF           AARGB2,F
MOVPF           PRODH,WREG
ADDWFC          AARGB1,F

MOVFP           TEMPB0,WREG
```

```
            MULWF           BARGB3
            MOVPF           PRODL,WREG
            ADDWF           AARGB4,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB3,F
            CLRF            WREG,F
            ADDWFC          AARGB2,F
            ADDWFC          AARGB1,F

            MOVFP           TEMPB0,WREG
            MULWF           BARGB0
            MOVPF           PRODH,AARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB1,F
            CLRF            WREG,F
            ADDWFC          AARGB0,F

            MOVFP           TBLPTRL,WREG
            MULWF           BARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB4,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB3,F
            CLRF            WREG,F
            ADDWFC          AARGB2,F
            ADDWFC          AARGB1,F
            ADDWFC          AARGB0,F

            MOVFP           TEMPB2,WREG
            MULWF           BARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB3,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB2,F
            CLRF            WREG,F
            ADDWFC          AARGB1,F
            ADDWFC          AARGB0,F

            MOVFP           TEMPB1,WREG
            MULWF           BARGB0
            MOVPF           PRODL,WREG
            ADDWF           AARGB2,F
            MOVPF           PRODH,WREG
            ADDWFC          AARGB1,F
            CLRF            WREG,F
            ADDWFC          AARGB0,F

            BTFSC           AARGB0,MSB
            GOTO            DEXP32
            RLCF            AARGB3,F
            RLCF            AARGB2,F
            RLCF            AARGB1,F
            RLCF            AARGB0,F
            DECF            TEMPB3,F

            BTFSC           AARGB0,MSB
            GOTO            DEXP32
            RLCF            AARGB3,F
            RLCF            AARGB2,F
            RLCF            AARGB1,F
            RLCF            AARGB0,F
            DECF            TEMPB3,F

DEXP32      MOVFP           BEXP,WREG           ; compute AEXP - BEXP
            SUBWF           EXP,F
            MOVLW           EXPBIAS+1           ; add bias + 1 for scaling of F0
```

```
                BTFSS           _C
                GOTO            ALTB32

AGEB32          ADDWF           TEMPB3,W                ; if AEXP > BEXP, test for overflow
                ADDWF           EXP,F
                BTFSC           _C
                GOTO            SETFOV32
                GOTO            DROUND32

ALTB32          ADDWF           TEMPB3,W                ; if AEXP < BEXP, test for underflow
                ADDWF           EXP,F
                BTFSS           _C
                GOTO            SETFUN32

DROUND32
                BTFSC           FPFLAGS,RND             ; is rounding enabled?
                BTFSS           AARGB3,MSB              ; is NSB > 0x80?
                GOTO            DIV32OK
                BSF             _C                      ; set carry for rounding
                MOVLW           0x80
                CPFSGT          AARGB3                  ; if NSB = 0x80, select even
                RRCF            AARGB2,W                ; using lsb in carry
                CLRF            WREG,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F

                BTFSS           _C                      ; test if rounding caused carryout
                GOTO            DIV32OK
                RRCF            AARGB0,F
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                INFSNZ          EXP,F                   ; test for overflow
                GOTO            SETFOV32

DIV32OK         BTFSS           SIGN,MSB
                BCF             AARGB0,MSB              ; clear explicit MSB if positive

                RETLW           0

SETFUN32        BSF             FPFLAGS,FUN             ; set floating point underflag
                BTFSS           FPFLAGS,SAT             ; test for saturation
                RETLW           0xFF                    ; return error code in WREG

                MOVLW           0x01                    ; saturate to smallest floating
                MOVPF           WREG,AEXP               ; point number = 0x 01 00 00 00
                CLRF            AARGB0,F                ; modulo the appropriate sign bit
                CLRF            AARGB1,F
                CLRF            AARGB2,F
                RLCF            SIGN,F
                RRCF            AARGB0,F
                RETLW           0xFF                    ; return error code in WREG

SETFDZ32        BSF             FPFLAGS,FDZ             ; set floating point divide by zero
                RETLW           0xFF                    ; flag and return error code in WREG

;-----------------------------------------------------------------------------------------------

;        table for interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 6 explicit bits of BARG as a pointer
;        and the remaining 17 explicit bits as the argument to linear interpolation.

IBTBL64I
                DATA        0xFFFF
                DATA        0xFC10
                DATA        0xF83E
```

```
DATA        0xF48A
DATA        0xF0F1
DATA        0xED73
DATA        0xEA0F
DATA        0xE6C3
DATA        0xE38E
DATA        0xE070
DATA        0xDD68
DATA        0xDA74
DATA        0xD794
DATA        0xD4C7
DATA        0xD20D
DATA        0xCF64
DATA        0xCCCD
DATA        0xCA46
DATA        0xC7CE
DATA        0xC566
DATA        0xC30C
DATA        0xC0C1
DATA        0xBE83
DATA        0xBC52
DATA        0xBA2F
DATA        0xB817
DATA        0xB60B
DATA        0xB40B
DATA        0xB216
DATA        0xB02C
DATA        0xAE4C
DATA        0xAC77
DATA        0xAAAB
DATA        0xA8E8
DATA        0xA72F
DATA        0xA57F
DATA        0xA3D7
DATA        0xA238
DATA        0xA0A1
DATA        0x9F11
DATA        0x9D8A
DATA        0x9C0A
DATA        0x9A91
DATA        0x991F
DATA        0x97B4
DATA        0x9650
DATA        0x94F2
DATA        0x939B
DATA        0x9249
DATA        0x90FE
DATA        0x8FB8
DATA        0x8E78
DATA        0x8D3E
DATA        0x8C09
DATA        0x8AD9
DATA        0x89AE
DATA        0x8889
DATA        0x8768
DATA        0x864C
DATA        0x8534
DATA        0x8421
DATA        0x8312
DATA        0x8208
DATA        0x8102
DATA        0x8001
```

```
;-----------------------------------------------------------------------------------------------

;        generation of F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 7 explicit bits of BARG as a pointer
;        and the remaining 16 explicit bits as the argument to linear interpolation.

IBTBL128I
                DATA        0xFFFF
                DATA        0xFE04
                DATA        0xFC10
                DATA        0xFA23
                DATA        0xF83E
                DATA        0xF660
                DATA        0xF48A
                DATA        0xF2BA
                DATA        0xF0F1
                DATA        0xEF2F
                DATA        0xED73
                DATA        0xEBBE
                DATA        0xEA0F
                DATA        0xE866
                DATA        0xE6C3
                DATA        0xE526
                DATA        0xE38E
                DATA        0xE1FC
                DATA        0xE070
                DATA        0xDEE9
                DATA        0xDD68
                DATA        0xDBEB
                DATA        0xDA74
                DATA        0xD902
                DATA        0xD794
                DATA        0xD62C
                DATA        0xD4C7
                DATA        0xD368
                DATA        0xD20D
                DATA        0xD0B7
                DATA        0xCF64
                DATA        0xCE17
                DATA        0xCCCD
                DATA        0xCB87
                DATA        0xCA46
                DATA        0xC908
                DATA        0xC7CE
                DATA        0xC698
                DATA        0xC566
                DATA        0xC437
                DATA        0xC30C
                DATA        0xC1E5
                DATA        0xC0C1
                DATA        0xBFA0
                DATA        0xBE83
                DATA        0xBD69
                DATA        0xBC52
                DATA        0xBB3F
                DATA        0xBA2F
                DATA        0xB921
                DATA        0xB817
                DATA        0xB710
                DATA        0xB60B
                DATA        0xB50A
                DATA        0xB40B
                DATA        0xB30F
                DATA        0xB216
                DATA        0xB120
                DATA        0xB02C
```

```
DATA        0xAF3B
DATA        0xAE4C
DATA        0xAD60
DATA        0xAC77
DATA        0xAB8F
DATA        0xAAAB
DATA        0xA9C8
DATA        0xA8E8
DATA        0xA80B
DATA        0xA72F
DATA        0xA656
DATA        0xA57F
DATA        0xA4AA
DATA        0xA3D7
DATA        0xA306
DATA        0xA238
DATA        0xA16B
DATA        0xA0A1
DATA        0x9FD8
DATA        0x9F11
DATA        0x9E4D
DATA        0x9D8A
DATA        0x9CC9
DATA        0x9C0A
DATA        0x9B4C
DATA        0x9A91
DATA        0x99D7
DATA        0x991F
DATA        0x9869
DATA        0x97B4
DATA        0x9701
DATA        0x9650
DATA        0x95A0
DATA        0x94F2
DATA        0x9446
DATA        0x939B
DATA        0x92F1
DATA        0x9249
DATA        0x91A3
DATA        0x90FE
DATA        0x905A
DATA        0x8FB8
DATA        0x8F17
DATA        0x8E78
DATA        0x8DDA
DATA        0x8D3E
DATA        0x8CA3
DATA        0x8C09
DATA        0x8B70
DATA        0x8AD9
DATA        0x8A43
DATA        0x89AE
DATA        0x891B
DATA        0x8889
DATA        0x87F8
DATA        0x8768
DATA        0x86D9
DATA        0x864C
DATA        0x85BF
DATA        0x8534
DATA        0x84AA
DATA        0x8421
DATA        0x8399
DATA        0x8312
DATA        0x828D
DATA        0x8208
```

```
                DATA        0x8185
                DATA        0x8102
                DATA        0x8081
                DATA        0x8001

;-----------------------------------------------------------------------------------------------

;        table for F0 by interpolating between consecutive 16 bit approximations
;        to the reciprocal of BARG, with the top 8 explicit bits of BARG as a pointer
;        and the remaining 7 explicit bits as the argument to linear interpolation.

IBTBL256I
                DATA        0xFFFF
                DATA        0xFF01
                DATA        0xFE04
                DATA        0xFD09
                DATA        0xFC10
                DATA        0xFB19
                DATA        0xFA23
                DATA        0xF930
                DATA        0xF83E
                DATA        0xF74E
                DATA        0xF660
                DATA        0xF574
                DATA        0xF48A
                DATA        0xF3A1
                DATA        0xF2BA
                DATA        0xF1D5
                DATA        0xF0F1
                DATA        0xF00F
                DATA        0xEF2F
                DATA        0xEE50
                DATA        0xED73
                DATA        0xEC98
                DATA        0xEBBE
                DATA        0xEAE5
                DATA        0xEA0F
                DATA        0xE939
                DATA        0xE866
                DATA        0xE793
                DATA        0xE6C3
                DATA        0xE5F3
                DATA        0xE526
                DATA        0xE459
                DATA        0xE38E
                DATA        0xE2C5
                DATA        0xE1FC
                DATA        0xE136
                DATA        0xE070
                DATA        0xDFAC
                DATA        0xDEE9
                DATA        0xDE28
                DATA        0xDD68
                DATA        0xDCA9
                DATA        0xDBEB
                DATA        0xDB2F
                DATA        0xDA74
                DATA        0xD9BA
                DATA        0xD902
                DATA        0xD84A
                DATA        0xD794
                DATA        0xD6DF
                DATA        0xD62C
                DATA        0xD579
                DATA        0xD4C7
                DATA        0xD417
```

```
DATA        0xD368
DATA        0xD2BA
DATA        0xD20D
DATA        0xD161
DATA        0xD0B7
DATA        0xD00D
DATA        0xCF64
DATA        0xCEBD
DATA        0xCE17
DATA        0xCD71
DATA        0xCCCD
DATA        0xCC29
DATA        0xCB87
DATA        0xCAE6
DATA        0xCA46
DATA        0xC9A6
DATA        0xC908
DATA        0xC86A
DATA        0xC7CE
DATA        0xC733
DATA        0xC698
DATA        0xC5FE
DATA        0xC566
DATA        0xC4CE
DATA        0xC437
DATA        0xC3A1
DATA        0xC30C
DATA        0xC278
DATA        0xC1E5
DATA        0xC152
DATA        0xC0C1
DATA        0xC030
DATA        0xBFA0
DATA        0xBF11
DATA        0xBE83
DATA        0xBDF6
DATA        0xBD69
DATA        0xBCDD
DATA        0xBC52
DATA        0xBBC8
DATA        0xBB3F
DATA        0xBAB6
DATA        0xBA2F
DATA        0xB9A8
DATA        0xB921
DATA        0xB89C
DATA        0xB817
DATA        0xB793
DATA        0xB710
DATA        0xB68D
DATA        0xB60B
DATA        0xB58A
DATA        0xB50A
DATA        0xB48A
DATA        0xB40B
DATA        0xB38D
DATA        0xB30F
DATA        0xB292
DATA        0xB216
DATA        0xB19B
DATA        0xB120
DATA        0xB0A6
DATA        0xB02C
DATA        0xAFB3
DATA        0xAF3B
DATA        0xAEC3
```

```
DATA        0xAE4C
DATA        0xADD6
DATA        0xAD60
DATA        0xACEB
DATA        0xAC77
DATA        0xAC03
DATA        0xAB8F
DATA        0xAB1D
DATA        0xAAAB
DATA        0xAA39
DATA        0xA9C8
DATA        0xA958
DATA        0xA8E8
DATA        0xA879
DATA        0xA80B
DATA        0xA79C
DATA        0xA72F
DATA        0xA6C2
DATA        0xA656
DATA        0xA5EA
DATA        0xA57F
DATA        0xA514
DATA        0xA4AA
DATA        0xA440
DATA        0xA3D7
DATA        0xA36E
DATA        0xA306
DATA        0xA29F
DATA        0xA238
DATA        0xA1D1
DATA        0xA16B
DATA        0xA106
DATA        0xA0A1
DATA        0xA03C
DATA        0x9FD8
DATA        0x9F74
DATA        0x9F11
DATA        0x9EAF
DATA        0x9E4D
DATA        0x9DEB
DATA        0x9D8A
DATA        0x9D29
DATA        0x9CC9
DATA        0x9C69
DATA        0x9C0A
DATA        0x9BAB
DATA        0x9B4C
DATA        0x9AEE
DATA        0x9A91
DATA        0x9A34
DATA        0x99D7
DATA        0x997B
DATA        0x991F
DATA        0x98C4
DATA        0x9869
DATA        0x980E
DATA        0x97B4
DATA        0x975A
DATA        0x9701
DATA        0x96A8
DATA        0x9650
DATA        0x95F8
DATA        0x95A0
DATA        0x9549
DATA        0x94F2
DATA        0x949C
```

```
DATA        0x9446
DATA        0x93F0
DATA        0x939B
DATA        0x9346
DATA        0x92F1
DATA        0x929D
DATA        0x9249
DATA        0x91F6
DATA        0x91A3
DATA        0x9150
DATA        0x90FE
DATA        0x90AC
DATA        0x905A
DATA        0x9009
DATA        0x8FB8
DATA        0x8F68
DATA        0x8F17
DATA        0x8EC8
DATA        0x8E78
DATA        0x8E29
DATA        0x8DDA
DATA        0x8D8C
DATA        0x8D3E
DATA        0x8CF0
DATA        0x8CA3
DATA        0x8C56
DATA        0x8C09
DATA        0x8BBC
DATA        0x8B70
DATA        0x8B24
DATA        0x8AD9
DATA        0x8A8E
DATA        0x8A43
DATA        0x89F8
DATA        0x89AE
DATA        0x8964
DATA        0x891B
DATA        0x88D2
DATA        0x8889
DATA        0x8840
DATA        0x87F8
DATA        0x87AF
DATA        0x8768
DATA        0x8720
DATA        0x86D9
DATA        0x8692
DATA        0x864C
DATA        0x8605
DATA        0x85BF
DATA        0x8579
DATA        0x8534
DATA        0x84EF
DATA        0x84AA
DATA        0x8465
DATA        0x8421
DATA        0x83DD
DATA        0x8399
DATA        0x8356
DATA        0x8312
DATA        0x82CF
DATA        0x828D
DATA        0x824A
DATA        0x8208
DATA        0x81C6
DATA        0x8185
DATA        0x8143
```

```
                DATA        0x8102
                DATA        0x80C1
                DATA        0x8081
                DATA        0x8040
                DATA        0x8001

;*****************************************************************************************
;*****************************************************************************************

;       Floating Point Subtract

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL FPS32

;       Output: 32 bit floating point difference in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     1+160 = 161 clks                RND = 0
;                       1+176 = 177 clks                RND = 1, SAT = 0
;                       1+182 = 183 clks                RND = 1, SAT = 1

;       Min Timing:     1+20 = 21 clks

;       PM: 425                                         DM: 14

;-----------------------------------------------------------------------------------------

FPS32           BTG             BARGB0,MSB              ; toggle sign bit for subtraction

;*****************************************************************************************

;       Floating Point Add

;       Input:  32 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2
;               32 bit floating point number in BEXP, BARGB0, BARGB1, BARGB2

;       Use:    CALL FPA32

;       Output: 32 bit floating point sum in AEXP, AARGB0, AARGB1, AARGB2

;       Result: AARG  <--  AARG - BARG

;       Max Timing:     7+70+22+61 = 160 clks           RND = 0
;                       7+70+22+77 = 176 clks           RND = 1, SAT = 0
;                       7+70+22+83 = 182 clks           RND = 1, SAT = 1

;       Min Timing:     6+14 = 20 clks

;       PM: 424                                         DM: 14

;-----------------------------------------------------------------------------------------

FPA32           MOVFP           AARGB0,WREG             ; exclusive or of signs in TEMPB0
                XORWF           BARGB0,W
                MOVPF           WREG,TEMPB0

                CLRF            AARGB3,F

                MOVFP           AEXP,WREG               ; use AARG if AEXP >= BEXP
                CPFSGT          BEXP
                GOTO            USEA32

USEB32          MOVFP           BARGB0,WREG             ; use BARG if AEXP < BEXP
```

```
            MOVPF       WREG,SIGN           ; save sign in SIGN
            BSF         BARGB0,MSB          ; make MSB's explicit
            BSF         AARGB0,MSB

            MOVFP       AEXP,WREG           ; compute shift count in BEXP
            MOVPF       WREG,TEMPB1
            MOVFP       BEXP,WREG
            MOVPF       WREG,AEXP

            CLRF        WREG,F
            CPFSGT      TEMPB1              ; return BARG if AARG = 0
            GOTO        BRETURN32
            MOVFP       TEMPB1,WREG
            SUBWF       BEXP,F
            BTFSC       _Z
            GOTO        BLIGNED32

            MOVLW       7
            CPFSGT      BEXP                ; do byte shift if BEXP >= 8
            GOTO        BNIB32

            SUBWF       BEXP,F              ; BEXP = BEXP - 7
            MOVFP       AARGB2,AARGB3       ; keep for postnormalization
            MOVFP       AARGB1,AARGB2
            MOVFP       AARGB0,AARGB1
            CLRF        AARGB0,F
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED32

            CPFSGT      BEXP                ; do another byte shift if BEXP >= 8
            GOTO        BNIB32A
            SUBWF       BEXP,F              ; BEXP = BEXP - 7
            MOVFP       AARGB2,AARGB3       ; keep for postnormalization
            MOVFP       AARGB1,AARGB2
            CLRF        AARGB1,F
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED32

            CPFSGT      BEXP                ; do another byte shift if BEXP >= 8
            GOTO        BNIB32B
            SUBWF       BEXP,F              ; BEXP = BEXP - 7
            MOVFP       AARGB2,AARGB3       ; keep for postnormalization
            CLRF        AARGB2,F
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED32

            CPFSGT      BEXP                ; if BEXP still >= 8, then
            GOTO        BNIB32C             ; AARG = 0 relative to BARG

BRETURN32   MOVFP       SIGN,AARGB0         ; return BARG
            MOVFP       BARGB1,AARGB1
            MOVFP       BARGB2,AARGB2
            CLRF        AARGB3,F
            RETLW       0x00

BNIB32C     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        BLOOP32C
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       AARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB3
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED32           ; aligned if BEXP = 0

BLOOP32C    BCF         _C                  ; right shift by BEXP
```

```
                RRCF            AARGB3,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED32               ; aligned if BEXP = 0
                BCF             _C
                RRCF            AARGB3,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED32               ; aligned if BEXP = 0
                BCF             _C                      ; at most 3 right shifts are
                RRCF            AARGB3,F                ; possible
                GOTO            BLIGNED32

BNIB32B         MOVLW           3                       ; do nibbleshift if BEXP >= 4
                CPFSGT          BEXP
                GOTO            BLOOP32B
                SUBWF           BEXP,F                  ; BEXP = BEXP -3
                SWAPF           AARGB3,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB3
                SWAPF           AARGB2,W
                ANDLW           0xF0
                ADDWF           AARGB3,F
                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            BLIGNED32               ; aligned if BEXP = 0

BLOOP32B        BCF             _C                      ; right shift by BEXP
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED32               ; aligned if BEXP = 0
                BCF             _C
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                DCFSNZ          BEXP,F
                GOTO            BLIGNED32               ; aligned if BEXP = 0
                BCF             _C                      ; at most 3 right shifts are
                RRCF            AARGB2,F                ; possible
                RRCF            AARGB3,F
                GOTO            BLIGNED32

BNIB32A         MOVLW           3                       ; do nibbleshift if BEXP >= 4
                CPFSGT          BEXP
                GOTO            BLOOP32A
                SUBWF           BEXP,F                  ; BEXP = BEXP -3
                SWAPF           AARGB3,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB3
                SWAPF           AARGB2,W
                ANDLW           0xF0
                ADDWF           AARGB3,F
                SWAPF           AARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB2
                SWAPF           AARGB1,W
                ANDLW           0xF0
                ADDWF           AARGB2,F
                SWAPF           AARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,AARGB1
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            BLIGNED32               ; aligned if BEXP = 0

BLOOP32A        BCF             _C                      ; right shift by BEXP
                RRCF            AARGB1,F
```

```
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED32           ; aligned if BEXP = 0
            BCF         _C
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED32           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        AARGB1,F            ; possible
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            GOTO        BLIGNED32

BNIB32      MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        BLOOP32
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       AARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB3
            SWAPF       AARGB2,W
            ANDLW       0xF0
            ADDWF       AARGB3,F
            SWAPF       AARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB2
            SWAPF       AARGB1,W
            ANDLW       0xF0
            ADDWF       AARGB2,F
            SWAPF       AARGB1,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB1
            SWAPF       AARGB0,W
            ANDLW       0xF0
            ADDWF       AARGB1,F
            SWAPF       AARGB0,W
            ANDLW       0x0F
            MOVPF       WREG,AARGB0
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        BLIGNED32           ; aligned if BEXP = 0

BLOOP32     BCF         _C                  ; right shift by BEXP
            RRCF        AARGB0,F
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED32           ; aligned if BEXP = 0
            BCF         _C
            RRCF        AARGB0,F
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            RRCF        AARGB3,F
            DCFSNZ      BEXP,F
            GOTO        BLIGNED32           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        AARGB0,F            ; possible
            RRCF        AARGB1,F
            RRCF        AARGB2,F
            RRCF        AARGB3,F

BLIGNED32   CLRF        BARGB3,W
            BTFSS       TEMPB0,MSB          ; negate if signs opposite
```

```
                GOTO            AOK32
                COMF            AARGB3,F
                COMF            AARGB2,F
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB3,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                GOTO            AOK32

USEA32          TSTFSZ          BEXP                    ; return AARG if BARG = 0
                GOTO            BNE032
                RETLW           0x00

BNE032          CLRF            BARGB3,F
                MOVPF           AARGB0,SIGN             ; save sign in SIGN
                BSF             AARGB0,MSB              ; make MSB's explicit
                BSF             BARGB0,MSB

                MOVFP           BEXP,WREG               ; compute shift count in BEXP
                SUBWF           AEXP,W
                MOVPF           WREG,BEXP
                BTFSC           _Z
                GOTO            ALIGNED32

                MOVLW           7
                CPFSGT          BEXP                    ; do byte shift if BEXP >= 8
                GOTO            ANIB32
                SUBWF           BEXP,F                  ; BEXP = BEXP - 7
                MOVFP           BARGB2,WREG             ; keep for postnormalization
                MOVPF           WREG,BARGB3
                MOVFP           BARGB1,WREG
                MOVPF           WREG,BARGB2
                MOVFP           BARGB0,WREG
                MOVPF           WREG,BARGB1
                CLRF            BARGB0,F
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            ALIGNED32

                MOVLW           7
                CPFSGT          BEXP                    ; do another byte shift if BEXP >= 8
                GOTO            ANIB32A
                SUBWF           BEXP,F                  ; BEXP = BEXP - 7
                MOVFP           BARGB2,WREG
                MOVPF           WREG,BARGB3
                MOVFP           BARGB1,WREG
                MOVPF           WREG,BARGB2
                CLRF            BARGB1,F
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            ALIGNED32

                MOVLW           7
                CPFSGT          BEXP                    ; do another byte shift if BEXP >= 8
                GOTO            ANIB32B
                SUBWF           BEXP,F                  ; BEXP = BEXP - 7
                MOVFP           BARGB2,WREG
                MOVPF           WREG,BARGB3
                CLRF            BARGB2,F
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            ALIGNED32

                MOVLW           7
                CPFSGT          BEXP                    ; if BEXP still >= 8, then
                GOTO            ANIB32C                 ; BARG = 0 relative to AARG
```

```
            MOVFP       SIGN,AARGB0         ; return AARG
            RETLW       0x00

ANIB32C     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP32C
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB3
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED32           ; aligned if BEXP = 0

ALOOP32C    BCF         _C                  ; right shift by BEXP
            RRCF        BARGB3,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED32           ; aligned if BEXP = 0
            BCF         _C
            RRCF        BARGB3,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED32           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        BARGB3,F            ; possible
            GOTO        ALIGNED32

ANIB32B     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP32B
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB3
            SWAPF       BARGB2,W
            ANDLW       0xF0
            ADDWF       BARGB3,F
            SWAPF       BARGB2,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB2
            DCFSNZ      BEXP,F              ; BEXP = BEXP - 1
            GOTO        ALIGNED32           ; aligned if BEXP = 0

ALOOP32B    BCF         _C                  ; right shift by BEXP
            RRCF        BARGB2,F
            RRCF        BARGB3,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED32           ; aligned if BEXP = 0
            BCF         _C
            RRCF        BARGB2,F
            RRCF        BARGB3,F
            DCFSNZ      BEXP,F
            GOTO        ALIGNED32           ; aligned if BEXP = 0
            BCF         _C                  ; at most 3 right shifts are
            RRCF        BARGB2,F            ; possible
            RRCF        BARGB3,F
            GOTO        ALIGNED32

ANIB32A     MOVLW       3                   ; do nibbleshift if BEXP >= 4
            CPFSGT      BEXP
            GOTO        ALOOP32A
            SUBWF       BEXP,F              ; BEXP = BEXP -3
            SWAPF       BARGB3,W
            ANDLW       0x0F
            MOVPF       WREG,BARGB3
            SWAPF       BARGB2,W
            ANDLW       0xF0
            ADDWF       BARGB3,F
```

```
                SWAPF           BARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB2
                SWAPF           BARGB1,W
                ANDLW           0xF0
                ADDWF           BARGB2,F
                SWAPF           BARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB1
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            ALIGNED32               ; aligned if BEXP = 0

ALOOP32A        BCF             _C                      ; right shift by BEXP
                RRCF            BARGB1,F
                RRCF            BARGB2,F
                RRCF            BARGB3,F
                DCFSNZ          BEXP,F
                GOTO            ALIGNED32               ; aligned if BEXP = 0
                BCF             _C
                RRCF            BARGB1,F
                RRCF            BARGB2,F
                RRCF            BARGB3,F
                DCFSNZ          BEXP,F
                GOTO            ALIGNED32               ; aligned if BEXP = 0
                BCF             _C                      ; at most 3 right shifts are
                RRCF            BARGB1,F                ; possible
                RRCF            BARGB2,F
                RRCF            BARGB3,F
                GOTO            ALIGNED32

ANIB32          MOVLW           3                       ; do nibbleshift if BEXP >= 4
                CPFSGT          BEXP
                GOTO            ALOOP32
                SUBWF           BEXP,F                  ; BEXP = BEXP -3
                SWAPF           BARGB3,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB3
                SWAPF           BARGB2,W
                ANDLW           0xF0
                ADDWF           BARGB3,F
                SWAPF           BARGB2,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB2
                SWAPF           BARGB1,W
                ANDLW           0xF0
                ADDWF           BARGB2,F
                SWAPF           BARGB1,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB1
                SWAPF           BARGB0,W
                ANDLW           0xF0
                ADDWF           BARGB1,F
                SWAPF           BARGB0,W
                ANDLW           0x0F
                MOVPF           WREG,BARGB0
                DCFSNZ          BEXP,F                  ; BEXP = BEXP - 1
                GOTO            ALIGNED32               ; aligned if BEXP = 0

ALOOP32         BCF             _C                      ; right shift by BEXP
                RRCF            BARGB0,F
                RRCF            BARGB1,F
                RRCF            BARGB2,F
                RRCF            BARGB3,F
                DCFSNZ          BEXP,F
                GOTO            ALIGNED32               ; aligned if BEXP = 0
                BCF             _C
```

```
                RRCF            BARGB0,F
                RRCF            BARGB1,F
                RRCF            BARGB2,F
                RRCF            BARGB3,F
                DCFSNZ          BEXP,F
                GOTO            ALIGNED32               ; aligned if BEXP = 0
                BCF             _C                      ; at most 3 right shifts are
                RRCF            BARGB0,F                ; possible
                RRCF            BARGB1,F
                RRCF            BARGB2,F
                RRCF            BARGB3,F

ALIGNED32       CLRF            AARGB3,W
                BTFSS           TEMPB0,MSB              ; negate if signs opposite
                GOTO            AOK32
                COMF            BARGB3,F
                COMF            BARGB2,F
                COMF            BARGB1,F
                COMF            BARGB0,F
                INCF            BARGB3,F
                ADDWFC          BARGB2,F
                ADDWFC          BARGB1,F
                ADDWFC          BARGB0,F

AOK32           MOVFP           BARGB3,WREG
                ADDWF           AARGB3,F
                MOVFP           BARGB2,WREG             ; add
                ADDWFC          AARGB2,F
                MOVFP           BARGB1,WREG
                ADDWFC          AARGB1,F
                MOVFP           BARGB0,WREG
                ADDWFC          AARGB0,F

                BTFSC           TEMPB0,MSB
                GOTO            ACOMP32
                BTFSS           _C
                GOTO            NRMRND4032

                RRCF            AARGB0,F                ; shift right and increment EXP
                RRCF            AARGB1,F
                RRCF            AARGB2,F
                RRCF            AARGB3,F
                INCFSZ          AEXP,F
                GOTO            NRMRND4032
                GOTO            SETFOV32                ; set floating point overflow flag

ACOMP32         BTFSC           _C
                GOTO            NRM4032                 ; normalize and fix sign
                CLRF            WREG,F
                COMF            AARGB3,F                ; negate, toggle sign bit and
                COMF            AARGB2,F                ; then normalize
                COMF            AARGB1,F
                COMF            AARGB0,F
                INCF            AARGB3,F
                ADDWFC          AARGB2,F
                ADDWFC          AARGB1,F
                ADDWFC          AARGB0,F
                BTG             SIGN,MSB
                GOTO            NRM4032
```

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Trademarks

Printed on recycled paper.

DNV Certification, Inc. USA
ANSI • RAB QMS ACCREDITED
DNV MSC The Netherlands Accredited by the RvA
DNV
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02