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,..., 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 e 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

 $-126 \le e \le 128$.

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
А	< 0x80	round to A
А	= 0x80	if A,LSb = 0, round to A
		if A,LSb = 1, round to A+1
Α	> 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	
А	0	round to A	
А	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	fI	f2
IEEE754 32-bit	sxxx xxxx	<i>y</i> · 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, $\cdot = radix$ 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	6/17_to_IEEE		
RLCF	AARGB0,F	RLCF	AARGB0,F		
RLCF	AEXP,F	RRCF	AEXP,F		
RRCF	AARGBO F	RRCF	AARGRO 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	7FFFF	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

$$A = 16\pi = 50.2654824574 \approx \widehat{A} = 0x84490$$
FDB,

implying a biased exponent eb = 0x84, and the fraction or mantissa f = 0x490 FDB. To obtain the base 2 exponent e, we subtract the bias 0x7F, yielding

$$e = eb - bias = 0x84 - 0x7F = 0x05.$$

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

$$f = 1.100 \ 1001 \ 0000 \ 1111 \ 1101 \ 1011$$

The decimal equivalent of f can then be computed by adding the respective powers of two corresponding to nonzero bits.

$$f = 2^{0} + 2^{-1} + 2^{-4} + 2^{-7} + 2^{-12} + 2^{-13} + 2^{-14} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-19} + 2^{-20} + 2^{-22} + 2^{-23} = 1.5707963705,$$

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

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

24-bit Format

It is important to note that the difference between this evaluation of \widehat{A} and the number 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 A is given by

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

leading to the fraction f

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

and the decimal equivalent of A

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

with a correspondingly larger truncation error as expected. It is coincidence that both of these representations overestimate 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:

A = 0.15625 (Decimal Number)

(see algorithm A.3)

Find Exponent:

$$2^z = 0.15625$$

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

$$e = int(z) = -3$$

Find fractional part:

$$x = \frac{0.15625}{2^{-3}} = 1.25$$
 (x will always be ≥ 1)

$$.25 \ge 2^0$$
 ?,

$$a(0) = 1$$
:

$$k = 0$$
 1.25 $\ge 2^0$?, yes $a(0) = 1$; $x = 1.25 - 1 = 0.25$

$$.25 \ge 2^{-1}$$
 ?, no

$$a(1) = 0 \quad ; \quad x$$

$$(1) = 0$$
 , $(x = 1)$

$$0.25 \ge 2^{-2}$$
 ?,

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

Therefore,

 $f = 1.25 \text{ decimal} = 1.010 \quad 0000 \quad 0000 \quad 0000 \quad 0000 \quad \text{binary}$

$$\mathbf{A} = (-1)^{\mathbf{S}} \mathbf{f} \cdot 2^{\mathbf{e}}$$

$$f = x$$
,

$$s = 0$$
 (sign bit)

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

Now, convert 0.15625 to Microchip Float Format

eb = Biased Exponent

$$eb = e + 7Fh$$

$$eb = -3 + 7Fh$$

Microchip Float Format:

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
                          LSB to MSB
BARGB3
BARGB2
BARGB1
BARGB0
                          BARG fraction
BEXE
                          BARG exponent
TEMPB3
TEMPB2
TEMPB1
TEMPBO = 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	SATuı	rate	enabl	le bi	t			
RND	RouNI	Ding	enab]	le bi	t			
DOM	DOMa	in er	ror e	excep	tion	flag		
NAN	Not-A-Number exception flag							
FDZ	Float	ing	point	Div	ide k	oy Ze	ro	
FUN	Float	ing	point	Und	erflo	ow Fl	ag	
FOV	Float	ing	point	0ve	rflov	w Fla	g	
VOI	Integ	ger O	verfl	Low F	lag			

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

- Cavanagh, J.J.F., "Digital Computer Arithmetic," McGraw-Hill, 1984.
- Hwang, K., "Computer Arithmetic," John Wiley & Sons, 1979.
- Scott, N.R., "Computer Number Systems & Arithmetic," Prentice Hall, 1985.
- IEEE Standards Board, "IEEE Standard for Floating-Point Arithmetic," ANSI/IEEE Std 754-1985, IEEE, 1985.
- Knuth, D.E., "The Art of Computer Programming, Volume 2," Addison-Wesley, 1981.
- 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 <u>Integer conversion algorithm[3]:</u>

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 <u>Fractional conversion algorithm[3]:</u>

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 <u>Decimal to binary conversion algorithm:</u>

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 \boldsymbol{A} then has the floating point representation

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

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 \ge 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: ");
```

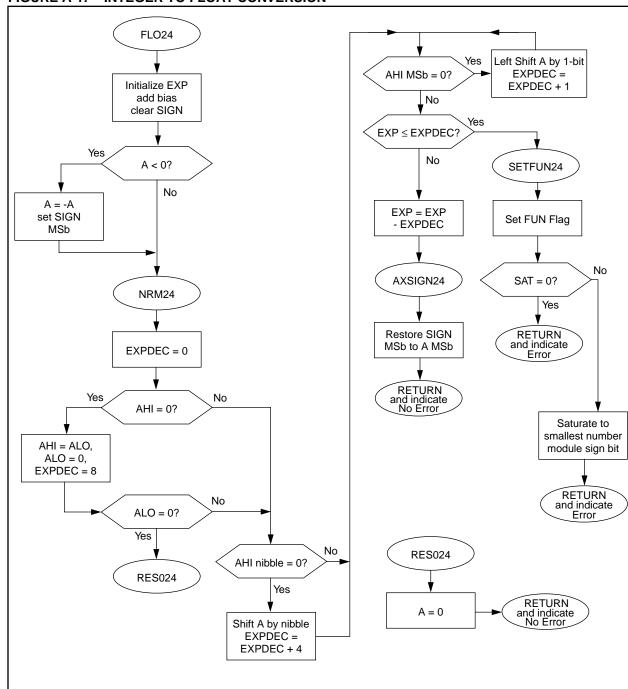
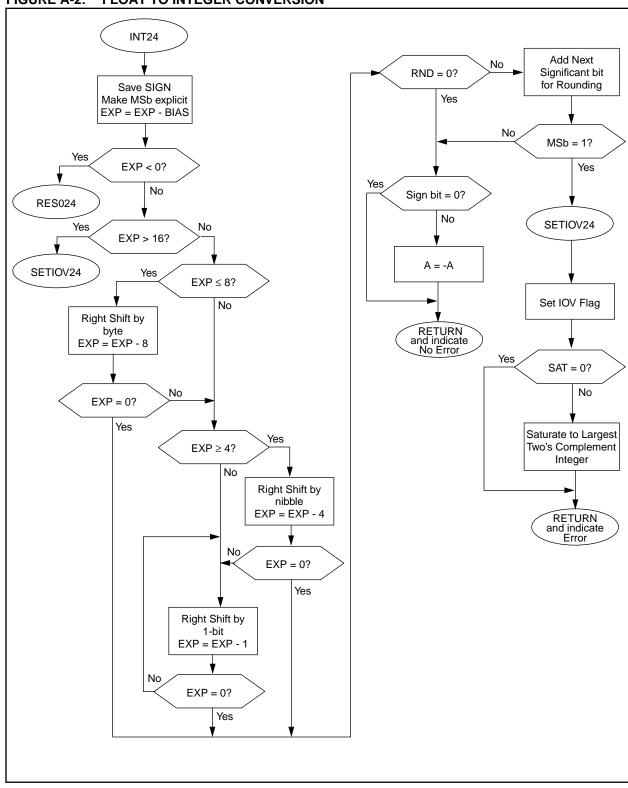
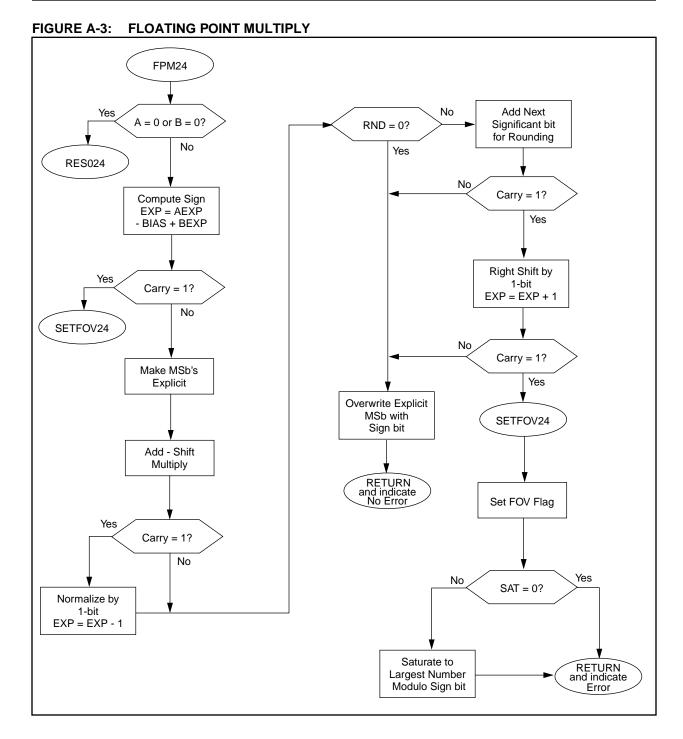
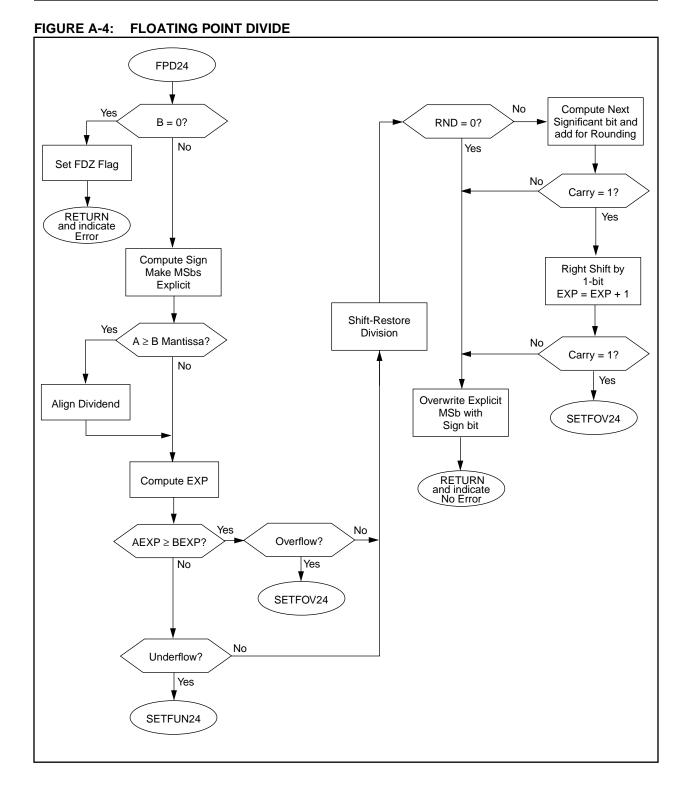


FIGURE A-1: INTEGER TO FLOAT CONVERSION







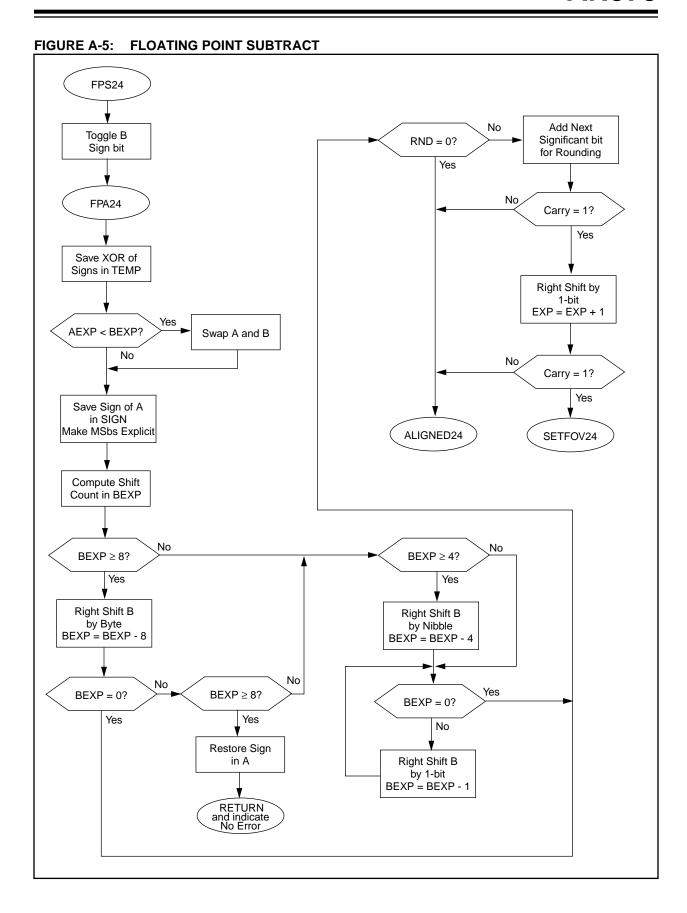


FIGURE A-6: NORMALIZATION

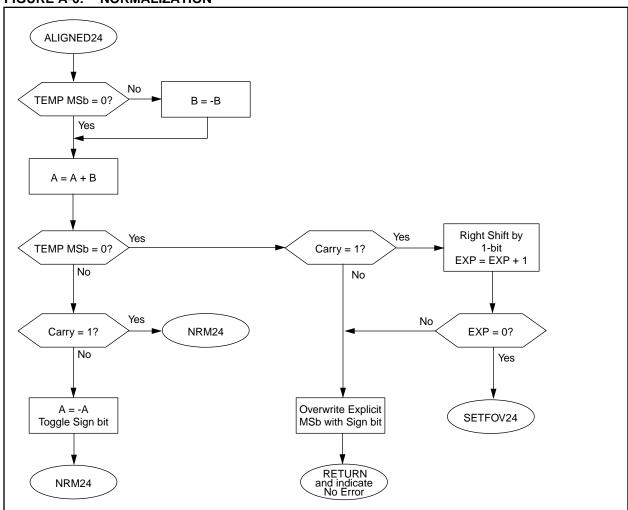


TABLE A-1: PIC17CXXX FLOATING POINT PERFORMANCE DATA

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 <u>Device Family Include File</u>

```
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
;*****
                      1.00.01
                                    03/21/97
;*****
TRUE
                  EOU 1
FALSE
                  EQU 0
P16C5X
         SET
                FALSE
                             ; If P16C5X, use INHX8M file format.
               FALSE
                             ; If P16CXX, use INHX8M file format.
P16CXX
          SET
P17CXX
           SET
                FALSE
                             ; If P17CXX, the INHX32 file format is required
                                         in the LIST directive
RESET_V
           SET
                0x0000
                             ; Default Reset Vector address of Oh
                             ; (16Cxx and 17Cxx devices)
P16_MAP1
           SET
                FALSE
                             ; FOR 16C60/61/70/71/710/711/715/84 Memory Map
P16_MAP2
                FALSE
                             ; For all other 16Cxx Memory Maps
          SET
;*****
          16CXX *******
           __14000
   IFDEF
P16CXX
           SET
                 TRUE
                             ; If P14000, use INHX8M file format.
P16_MAP2
                  TRUE
           SET
   ENDIF
   IFDEF
           __16C554
                             ; If P16C554, use INHX8M file format.
P16CXX
           SET
                TRUE
P16_MAP2
           SET
                 TRUE
   ENDIF
           __16C556
   IFDEF
                             ; If P16C556, use INHX8M file format.
P16CXX
           SET
                 TRUE
P16_MAP2
           SET
                 TRUE
   ENDIF
   IFDEF
           __16C558
P16CXX
                             ; If P16C558, use INHX8M file format.
                 TRUE
P16_MAP2
           SET
                 TRUE
   ENDIF
   IFDEF
            _16C61
P16CXX
                  TRUE
                             ; If P16C61, use INHX8M file format.
P16_MAP1
           SET
                  TRUE
   ENDIF
```

```
IFDEF __16C62
               TRUE
P16CXX
          SET
                           ; If P16C62, use INHX8M file format.
P16_MAP2
          SET
                 TRUE
   ENDIF
   чясчт
            _16C62A
P16CXX
                           ; If P16C62A, use INHX8M file format.
          SET
               TRUE
P16_MAP2
          SET
               TRUE
   ENDIF
           __16C63
   IFDEF
          SET
                            ; If P16C63, use INHX8M file format.
P16CXX
               TRUE
P16_MAP2
          SET
                 TRUE
   ENDIF
   IFDEF
           __16C64
         SET TRUE
P16CXX
                           ; If P16C64, use INHX8M file format.
P16_MAP2
          SET
                 TRUE
  ENDIF
   IFDEF
           __16C64A
P16CXX
          SET TRUE
                           ; If P16C64A, use INHX8M file format.
               TRUE
P16_MAP2
          SET
  ENDIF
           __16C65
   IFDEF
P16CXX
          SET TRUE
                            ; If P16C65, use INHX8M file format.
P16_MAP2
          SET
               TRIJE
   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
                            ; If P16C621, use INHX8M file format.
P16CXX
          SET TRUE
P16_MAP2
          SET
                 TRUE
  ENDIF
   IFDEF ___16C622
P16CXX
          SET TRUE
                           ; If P16C622, use INHX8M file format.
          SET
               TRIJE
P16_MAP2
   ENDIF
   IFDEF
           __16C642
                           ; If P16C642, use INHX8M file format.
P16CXX
          SET TRUE
               TRUE
P16_MAP2
          SET
  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
               TRUE
          SET
   ENDIF
   IFDEF __16C71
```

```
P16CXX
               TRUE
          SET
                           ; 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
          __16C72
   TROTE
P16CXX
          SET TRUE
                           ; If P16C72, use INHX8M file format.
                TRUE
P16_MAP2
          SET
   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
           _16C74A
   IFDEF
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
           __16F84
   IFDEF
P16CXX
          SET TRUE
                           ; If P16F84, use INHX8M file format.
               TRUE
P16_MAP1
          SET
  ENDIF
          __16F83
  IFDEF
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
                            ; If P16C923, use INHX8M file format.
P16CXX
          SET TRUE
               TRUE
P16_MAP2
          SET
   ENDIF
  IFDEF
          __16C924
P16CXX
          SET TRUE
                            ; If P16C924, use INHX8M file format.
```

```
P16_MAP2
                          SET
                                       TRUE
       ENDIF
        IFDEF __16CXX
                                                                     ; Generic Processor Type
                                                                ; If P16CXX, use INHX8M file format.
P16CXX
                          SET TRUE
P16_MAP2
                                       TRUE
                           SET
      ENDIF
;
;
                         17CXX ********
         IFDEF __17C42
                         SET TRUE
                                                                       ; If P17C42, the INHX32 file format is required
P17CXX
                                                                                                  in the LIST directive
;
        ENDIF
                            __17C43
       IFDEF
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
                       SET TRUE ; If P16C54, use INHX8M file format.

SET 0x01FF ; Reset Vector at end of 512 words
P16C5X
RESET_V
       ENDIF
FIDUDA SET TRUE ; If P16C54A, use INHX8M file format.

RESET_V SET 0x01FF ; Reset Vector of cold set o
      ENDIF
       IFDEF ___16C55
P16C5X SET TRUE
                                                                    ; If P16C55, use INHX8M file format.
                                       0x01FF
RESET_V
                          SET
                                                                      ; Reset Vector at end of 512 words
        ENDIF
        IFDEF
                            __16C56
                        SET TRUE ; If P16C56, use INHX8M file format.
SET 0x03FF ; Reset Vector at end of 1K words
P16C5X
RESET_V
      ENDIF
                          SET TRUE ; If P16C57, use INHX8M file format.
SET 0x07FF ; Reset Vector at 1000
      IFDEF ___16C57
P16C5X
                        SET TRUE
RESET_V
        ENDIF
        IFDEF
                               _16C58A
P16C5X
                        SET TRUE
                                                                    ; If P16C58A, use INHX8M file format.
RESET_V SET 0x07FF
                                                                    ; Reset Vector at end of 2K words
         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
вО
                 equ
                           0
В1
                 equ
                           1
                           2
B2
                 equ
                           3
В3
                 equ
В4
                 equ
                 equ
                           6
B6
                 equ
В7
                 equ
MSB
                           7
                 equ
LSB
                 equ
                           0
     define commonly used bits
     STATUS bit definitions
              _C
#define
                               STATUS, 0
#define
              _Z
                               STATUS, 2
       general register variables
   IF ( P16_MAP1 )
ACCB7
               equ
                      0x0C
                      0x0D
ACCB6
               equ
ACCB5
                      0x0E
               equ
ACCB4
               equ
                      0x0F
ACCB3
                      0x10
                      0x11
ACCB2
               equ
                      0x12
ACCB1
               equ
                      0x13
ACCB0
               equ
ACC
                      0x13
                              ; most significant byte of contiguous 8 byte accumulator
               equ
SIGN
                      0x15
                              ; save location for sign in MSB
               equ
TEMPB3
               equ
                      0x1C
TEMPB2
                      0x1D
               equ
TEMPB1
               equ
                      0x1E
TEMPB()
               equ
                      0x1F
TEMP
                              ; temporary storage
                      0x1F
               equ
```

```
;
       binary operation arguments
AARGB7
                      0x0C
              equ
AARGB6
              equ
                      0x0D
AARGB5
              equ
                      0 \times 0 E
AARGB4
                      0x0F
              eau
AARGB3
              equ
                      0x10
AARGB2
              equ
                      0x11
AARGB1
              equ
                      0x12
AARGR0
                      0x13
              equ
AARG
                      0x13
              equ
                             ; most significant byte of argument A
BARGB3
              equ
                      0x17
BARGB2
              equ
                      0x18
                      0x19
BARGB1
              equ
BARGB0
                      0x1A
              equ
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
                      0x0C
              equ
                      0x0D
REMB2
              equ
REMB1
              equ
                      0x0E
REMB0
                      0x0F
                             ; most significant byte of remainder
              equ
LOOPCOUNT
              equ
                      0x20
                             ; loop counter
       *************************
       FLOATING POINT SPECIFIC DEFINITIONS
;
       literal constants
;
                   D'127'
EXPBIAS
            equ
;
;
       biased exponents
;
EXP
                         ; 8 bit biased exponent
             equ
                    0 \times 14
                         ; 8 bit biased exponent for argument A
AEXP
             eau
                    0x14
BEXP
                    0x1B
                         ; 8 bit biased exponent for argument B
             equ
;
       floating point library exception flags
;
;
FPFLAGS
             equ
                    0x16
                           ; floating point library exception flags
VOI
             equ
                    0
                           ; bit0 = integer overflow flag
                           ; bit1 = floating point overflow flag
FOV
             equ
                    1
                           ; bit2 = floating point underflow flag
FUN
                    2.
             equ
FDZ
                    3
                           ; bit3 = floating point divide by zero flag
             equ
NAN
                    4
                           ; bit4 = not-a-number exception flag
             equ
DOM
             equ
                    5
                           ; bit5 = domain error exception flag
                    6
                           ; bit6 = floating point rounding flag, 0 = truncation
RND
             equ
                           ; 1 = unbiased rounding to nearest LSB
                           ; bit7 = floating point saturate flag, 0 = terminate on
SAT
             equ
                            ; exception without saturation, 1 = terminate on
                            ; exception with saturation to appropriate value
   ENDIF
   IF ( P16_MAP2 )
```

```
ACCB7
              equ
                      0x20
ACCB6
                      0x21
              equ
ACCB5
              equ
                      0x22
ACCB4
              equ
                      0x23
                      0x24
ACCB3
              equ
                      0x25
ACCB2
              equ
ACCB1
              equ
                      0x26
ACCB0
              equ
                      0x27
                      0x27
                             ; most significant byte of contiguous 8 byte accumulator
ACC
              equ
;
                      0x29
                             ; save location for sign in MSB
SIGN
              equ
TEMPB3
              equ
                      0x30
TEMPB2
                      0 \times 31
              equ
TEMPB1
                      0x32
              equ
TEMPB0
              equ
                      0x33
TEMP
              equ
                      0x33
                             ; temporary storage
       binary operation arguments
AARGB7
              equ
                      0x20
AARGB6
              equ
                      0x21
                      0x22
AARGB5
              equ
                      0x23
AARGB4
              equ
AARGB3
              equ
                      0x24
AARGB2
                      0x25
              equ
AARGB1
              equ
                      0x26
AARGB0
              equ
                      0x27
AARG
                      0x27
                             ; most significant byte of argument A
              equ
BARGB3
              equ
                      0x2B
BARGB2
              equ
                      0x2C
BARGB1
                      0 \times 2D
              equ
BARGB0
                      0x2E
              equ
BARG
                             ; most significant byte of argument B
              equ
                      0x2E
       Note that AARG and ACC reference the same storage location
FIXED POINT SPECIFIC DEFINITIONS
       remainder storage
REMB3
              equ
                      0x20
REMB2
              equ
                      0x21
REMB1
                      0x22
              equ
REMB0
                      0x23
                             ; most significant byte of remainder
              equ
LOOPCOUNT
              equ
                      0x34
                             ; loop counter
                  ************************
       FLOATING POINT SPECIFIC DEFINITIONS
       literal constants
EXPBIAS
                     D'127'
              equ
       biased exponents
;
                      0x28
                           ; 8 bit biased exponent
EXP
              equ
                      0x28
                             ; 8 bit biased exponent for argument A
AEXP
              equ
BEXP
              equ
                      0x2F ; 8 bit biased exponent for argument B
;
```

```
floating point library exception flags
;
FPFLAGS
                            ; floating point library exception flags
              equ
                            ; bit0 = integer overflow flag
IOV
              equ
FOV
                            ; bit1 = floating point overflow flag
              equ
                     1
FUN
                            ; bit2 = floating point underflow flag
              eau
FDZ
                            ; bit3 = floating point divide by zero flag
              equ
NAN
              equ
                            ; bit4 = not-a-number exception flag
                           ; bit5 = domain error exception flag
DOM
              equ
                     5
RND
                            ; bit6 = floating point rounding flag, 0 = truncation
              equ
                             ; 1 = unbiased rounding to nearest LSb
                             ; bit7 = floating point saturate flag, 0 = terminate on
SAT
              equ
                             ; exception without saturation, 1 = terminate on
                             ; exception with saturation to appropriate value
ELEMENTARY FUNCTION MEMORY
CEXP
              equ
                        0x35
CARGB0
                        0x36
              eau
CARGB1
              equ
                        0x37
CARGB2
              equ
                        0x38
CARGB3
                        0x39
              equ
DEXP
                        0x3A
              equ
DARGB0
                        0x3B
              equ
DARGB1
                        0x3C
              equ
DARGB2
              equ
                        0x3D
DARGB3
                        0x3E
              equ
EEXP
              equ
                        0x3F
                        0 \times 40
EARGRO.
              equ
EARGB1
                        0x41
              eau
EARGB2
              equ
                        0x42
EARGB3
                        0x43
              equ
ZARGB0
                        0x44
              equ
ZARGB1
                        0x45
              eau
ZARGB2
                        0x46
              equ
ZARGB3
              equ
                        0x47
RANDB0
              equ
                        0x48
RANDB1
                        0x49
              equ
RANDB2
                        0x4A
              equ
RANDB3
              equ
                        0x4B
24-BIT FLOATING POINT CONSTANTS
       Machine precision
MACHEP24EXP
              equ
                        0x6F
                                          i = 1.52587890625e-5 = 2**-16
MACHEP24B0
              equ
                        0x00
MACHEP24B1
                        0x00
              eau
        Maximum argument to EXP24
MAXLOG24EXP
                        0x85
                                          ; 88.7228391117 = log(2**128)
              equ
MAXLOG24B0
              equ
                        0x31
MAXLOG24B1
                        0x72
              equ
        Minimum argument to EXP24
```

```
MINLOG24EXP
                          0x85
                                             i - 87.3365447506 = log(2**-126)
               equ
MINLOG24B0
                          0xAE
               equ
MINLOG24B1
                          0xAC
               equ
        Maximum argument to EXP1024
MAXLOG1024EXP
                         0x84
                                            ; 38.531839445 = log10(2**128)
               equ
MAXLOG1024B0
               equ
                         0x1A
MAXLOG1024B1
                         0x21
               equ
        Minimum argument to EXP1024
                                            ; -37.9297794537 = log10(2**-126)
MINLOG1024EXP
               equ
                          0x84
MINLOG1024B0
               equ
                         0 \times 97
MINLOG1024B1
                         0xB8
               equ
        Maximum representable number before overflow
MAXNIIM24EXP
                          OxFF
                                            ; 6.80554349248E38 = (2**128) * (2 - 2**-15)
               equ
MAXNUM24B0
                          0x7F
               equ
MAXNUM24B1
                          0xFF
               equ
        Minimum representable number before underflow
MINNUM24EXP
                         0x01
                                            ; 1.17549435082E-38 = (2**-126) * 1
               eau
MINNUM24B0
                          0x00
               equ
MINNUM24B1
                         0x00
               equ
       Loss threshold for argument to SIN24 and COS24
LOSSTHR24EXP
                         0x8B
                                             ; 4096 = sqrt(2**24)
               equ
LOSSTHR24B0
               equ
                          0x00
                         0x00
LOSSTHR24B1
               equ
32-BIT FLOATING POINT CONSTANTS
        Machine precision
MACHEP32EXP
               equ
                          0x67
                                             ; 5.96046447754E-8 = 2**-24
                          0x00
MACHEP32B0
               equ
MACHEP32B1
                          0x00
               equ
MACHEP32B2
                          0x00
               equ
       Maximum argument to EXP32
                         0x85
                                             ; 88.7228391117 = log(2**128)
MAXIOG32EXP
               equ
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
                          OxAC
               equ
MINLOG32B2
                          0x50
               equ
        Maximum argument to EXP1032
MAXLOG1032EXP
               equ
                         0x84
                                             ; 38.531839445 = log10(2**128)
MAXLOG1032B0
                          0x1A
               equ
MAXLOG1032B1
                          0x20
               equ
MAXLOG1032B2
               equ
                          0x9B
        Minimum argument to EXP1032
```

```
MINLOG1032EXP
                           0x84
                                               ; -37.9297794537 = log10(2**-126)
                equ
MINLOG1032B0
                equ
                           0x97
MINLOG1032B1
                           0xB8
                equ
MINLOG1032B2
                           0x18
                equ
         Maximum representable number before overflow
                                               ; 6.80564774407E38 = (2**128) * (2 - 2**-23)
MAXNUM32EXP
                           0xFF
                equ
MAXNUM32B0
                           0x7F
                equ
MAXNUM32B1
                           0xFF
                equ
MAXNUM32B2
                equ
                           0xFF
         Minimum representable number before underflow
MINNUM32EXP
                equ
                           0x01
                                               ; 1.17549435082E-38 = (2**-126) * 1
MINNUM32B0
                equ
                           0x00
                           0x00
MINNUM32B1
                equ
MINNUM32B2
                equ
                           0x00
         Loss threshold for argument to SIN32 and COS32
LOSSTHR32EXP
                           0x8B
                                               ; 4096 = sqrt(2**24)
                           0x00
LOSSTHR32B0
                equ
LOSSTHR32B1
                           0x00
                equ
LOSSTHR32B2
                           0x00
                equ
```

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
       {\tt P} type registers must remain so since {\tt 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
в0
                equ
                        0
                        1
В1
                equ
В2
                        2
                equ
в3
                equ
                equ
                        5
B5
                equ
В6
                equ
                        6
В7
                equ
                        7
MSB
                equ
                        0
LSB
                equ
     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
                       0x19
ACCB6
               equ
ACCB5
                       0x1A
               equ
ACCB4
               equ
                       0x1B
ACCB3
               equ
                       0x1C
                       0x1D
ACCB2
               equ
                       0x1E
ACCB1
               equ
ACCB0
                       0x1F
               equ
ACC
                       0x1F
                               ; most significant byte of contiguous 8 byte accumulator
               equ
                       0x21
                              ; save location for sign in MSB
SIGN
               equ
TEMPB3
               equ
                       0x28
TEMPB2
                       0x29
               equ
TEMPB1
               equ
                       0x2A
TEMPB()
               equ
                       0x2B
TEMP
                       0x2B
                               ; temporary storage
               equ
```

```
binary operation arguments
AARGB7
                      0x18
              equ
AARGB6
              equ
                      0x19
AARGB5
              equ
                      0x1A
AARGB4
                      0x1B
              eau
AARGB3
              equ
                      0x1C
AARGB2
              equ
                      0x1D
AARGB1
              equ
                      0x1E
AARGRO
                      0x1F
              equ
AARG
                      0x1F
                             ; most significant byte of argument A
              equ
BARGB3
              equ
                      0x23
BARGB2
              equ
                      0x24
                      0 \times 25
BARGB1
              equ
BARGB0
                      0x26
              eau
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
                      0x18
REMB3
              equ
REMB2
              equ
                      0x19
REMB1
                      0x1A
              equ
REMB0
                      0x1B
                             ; most significant byte of remainder
              equ
FLOATING POINT SPECIFIC DEFINITIONS
       literal constants
EXPBIAS
                     D'127'
              eau
       biased exponents
EXP
                      0 \times 20
                             ; 8 bit biased exponent
              eau
AEXP
                      0x20
                             ; 8 bit biased exponent for argument A
              eau
BEXP
              equ
                      0x27
                             ; 8 bit biased exponent for argument B
       floating point library exception flags
FPFLAGS
                      0x22
                             ; floating point library exception flags
              eau
IOV
              equ
                             ; bit0 = integer overflow flag
FOV
              equ
                      1
                             ; bit1 = floating point overflow flag
                            ; bit2 = floating point underflow flag
FUN
              equ
                      2
                            ; bit3 = floating point divide by zero flag
FDZ
                      3
              equ
                            ; bit4 = not-a-number exception flag
NAN
              equ
DOM
                            ; bit5 = domain error flag
              equ
RND
              equ
                      6
                             ; bit6 = floating point rounding flag, 0 = truncation
                             ; 1 = unbiased rounding to nearest LSB
SAT
                      7
                             ; bit7 = floating point saturate flag, 0 = terminate on
              eau
                             ; 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
                        0x31
                equ
CARGB3
                        0x30
                equ
                        0x39
DEXP
                equ
DARGB0
                        0x38
                equ
DARGB1
                        0x37
                equ
DARGB2
                        0x36
                equ
DARGB3
                equ
                        0x35
                        0x3E
EEXP
                equ
EARGB0
                equ
                        0x3D
EARGB1
                equ
                        0x3C
EARGB2
                equ
                        0x3B
EARGB3
                        0x3A
                equ
FEXP
                equ
                        0x43
FARGB0
                equ
                        0x42
FARGB1
                equ
                        0x41
FARGB2
                        0x40
                equ
FARGB3
                        0x3F
                equ
GEXP
                equ
                        0x48
                        0 \times 47
GARGB0
                equ
GARGB1
                equ
                        0x46
                        0x45
GARGB2
                equ
GARGB3
                        0 \times 44
                equ
                        0x2F
ZARGB0
                equ
ZARGB1
                        0x2E
                equ
ZARGB2
                        0x2D
                equ
ZARGB3
                        0x2C
                equ
RANDB0
                equ
                        0x4C
RANDB1
                        0x4B
                equ
RANDB2
                        0x4A
                equ
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
                        0x72
                equ
        Minimum argument to EXP24
MINLOG24EXP
                        0x85
                                         ; -87.3365447506 = log(2**-126)
                equ
MINLOG24B0
                equ
                        0xAE
MINLOG24B1
                        0xAC
                equ
```

```
Maximum argument to EXP1024
MAXLOG1024EXPe qu
                       0x84
                                     ; 38.531839445 = log10(2**128)
MAXLOG1024B0
               equ
                       0 \times 1 A
MAXLOG1024B1
               equ
                       0x21
      Minimum argument to EXP1024
MINLOG1024EXP
                                     ; -37.9297794537 = log10(2**-126)
               equ
                       0x84
MINLOG1024B0
                       0x97
              equ
MINLOG1024B1
                       0xB8
               eau
       Maximum representable number before overflow
                                     ; 6.80554349248E38 = (2**128) * (2 - 2**-15)
MAXNIIM24EXP
                       OxFF
               equ
MAXNUM24B0
                       0x7F
              eau
MAXNUM24B1
               equ
                       0xFF
       Minimum representable number before underflow
                       0x01
                                     ; 1.17549435082E-38 = (2**-126) * 1
MINNUM24EXP
               equ
MINNUM24B0
               equ
                       0 \times 00
MINNUM24B1
               equ
                       0x00
      Loss threshold for argument to SIN24 and COS24
LOSSTHR24EXP
                       0x8A
                                     ; LOSSTHR = sqrt(2**24)*PI/4
               equ
LOSSTHR24B0
                       0x49
               equ
LOSSTHR24B1
               equ
                       0x10
32-BIT FLOATING POINT CONSTANTS
     Machine precision
MACHEP32EXP
                       0x67
                                     ; 5.96046447754E-8 = 2**-24
               equ
MACHEP32B0
                       0x00
              equ
MACHEP32B1
                       0x00
               eau
MACHEP32B2
                       0 \times 00
               equ
      Maximum argument to EXP32
MAXLOG32EXP
                       0x85
                                     ; 88.7228391117 = log(2**128)
               equ
MAXLOG32B0
                       0x31
               equ
MAXLOG32B1
               equ
                       0x72
MAXLOG32B2
               equ
                       0x18
     Minimum argument to EXP32
MINLOG32EXP
               equ
                       0x85
                                      ; -87.3365447506 = log(2**-126)
MINLOG32B0
                       OXAE
               equ
MINLOG32B1
                       0xAC
               equ
MINLOG32B2
                       0x50
               equ
       Maximum argument to EXP1032
MAXLOG1032EXP
                       0x84
                                     ; 38.531839445 = log10(2**128)
               eau
MAXLOG1032B0
               equ
                       0x1A
MAXLOG1032B1
               equ
                       0x20
MAXLOG1032B2
               equ
                       0 \times 9 B
```

AN575

```
Minimum argument to EXP1032
MINLOG1032EXP
                equ
                         0x84
                                         i - 37.9297794537 = log10(2**-126)
MINLOG1032B0
                         0x97
                equ
MINLOG1032B1
                         0xB8
                 equ
MINLOG1032B2
                         0x18
                equ
        Maximum representable number before overflow
                                         ; 6.80564774407E38 = (2**128) * (2 - 2**-23)
MAXNUM32EXP
                         0xFF
                equ
MAXNUM32B0
                         0x7F
                equ
MAXNUM32B1
                         0xFF
                equ
MAXNUM32B2
                equ
                         0xFF
        Minimum representable number before underflow
                                         ; 1.17549435082E-38 = (2**-126) * 1
MINNUM32EXP
                equ
                         0x01
MINNUM32B0
                equ
                         0x00
MINNUM32B1
                         0x00
                equ
MINNUM32B2
                         0 \times 00
                equ
        Loss threshold for argument to SIN32 and COS32
LOSSTHR32EXP
                         0x8A
                                         ; LOSSTHR = sqrt(2**24)*PI/4
                 equ
LOSSTHR32B0
                         0x49
                equ
LOSSTHR32B1
                         0x0F
                equ
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
        Timing:
                           RND
                                 1
                        83
                                 83
           SAT
                        88
                                 88
NRM2424
                24 bit normalization of unnormalized 24 bit floating point numbers
NRM24
        Timing:
                            RND
                        0
                                 1
                        72
                                 72
           SAT
                        77
                                 77
INT2416
                24 bit floating point to 16 bit integer conversion
TNT24
        Timing:
                           RND
                                 1
                        83
                                 89
           SAT
                        83
                                 92
                1
FL02424
                24 bit integer to 24 bit floating point conversion
        Timing:
                           RND
                        0
                                 1
                        108
                                 117
           SAT
                1
                        108
                                 123
```

```
NRM3224
                 32 bit normalization of unnormalized 24 bit floating point numbers
        Timing:
                           RND
                        0
                        94
                                 103
           SAT
                        94
                                 109
INT2424
                24 bit floating point to 24 bit integer conversion
        Timing:
                           RND
                        0
                                  1
                        105
                                  113
           SAT
                        105
                                  115
FPA24
                24 bit floating point add
        Timing:
                           RND
                                 1
                                 208
                        197
           SAT
                        197
                                 213
FPS24
                24 bit floating point subtract
        Timing:
                           RND
                                 1
                        199
                                 240
           SAT
                        199
                                 215
FPM24
                24 bit floating point multiply
                           RND
        Timing:
                        298
                                 309
           SAT
                        298
                                 313
FPD24
                24 bit floating point divide
        Timing:
                           RND
                         472
                                 494
           SAT
                        472
                                 498
                1
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
      XXXXXXX
                    S.xxxxxxx
                                 xxxxxxx
                   RADIX
                   POINT
Integer to float conversion
      Input: 16 bit 2's complement integer right justified in AARGB0, AARGB1
      Use: CALL FL01624 or CALL FL024
      Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
      Result: AARG <-- FLOAT( AARG )
                                        SAT = 0
;
      Max Timing:
                   11+72 = 83 \text{ clks}
                    11+77 = 88 \text{ clks}
                                        SAT = 1
      Min Timing:
                   7+14 = 21 \text{ clks}
                                        AARG = 0
                    7+18 = 25 \text{ clks}
      PM: 11+26 = 37
                                        DM: 6
FL01624
FLO24
             MOVLW
                          D'15'+EXPBIAS
                                              ; initialize exponent and add bias
             MOVWF
                           EXP
             MOVF
                          AARGB0,W
             MOVWF
                          SIGN
                          AARGB0,MSB
             BTFSS
                                              ; test sign
             GOTO
                          NRM2424
             COMF
                          AARGB1,F
                                              ; if < 0, negate and set MSB in SIGN
                          AARGB0,F
             COME
             INCF
                           AARGB1,F
                           _Z
             BTFSC
             INCF
                           AARGB0,F
      Normalization routine
      Input: 24 bit unnormalized floating point number in AEXP, AARGBO, 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, AARGBO, AARGB1
      Result: AARG <-- NORMALIZE( AARG )
```

```
10+6+7*7+7 = 72 \text{ clks}
                                                     SAT = 0
       Max Timing:
                      10+6+7*7+1+11 = 77 clks
                                                     SAT = 1
       Min Timing:
                                                     AARG = 0
                      14 clks
                       5+9+4 = 18 \text{ clks}
      PM: 26
                                                     DM: 6
NRM2424
NRM24
               CLRF
                              TEMP
                                               ; clear exponent decrement
               MOVF
                              AARGB0,W
                                               ; test if highbyte=0
               BTFSS
                              _{\rm Z}
               GOTO
                              NORM2424
               MOVF
                              AARGB1,W
                                               ; if so, shift 8 bits by move
               MOVWF
                              AARGB0
                                               ; if highbyte=0, result=0
               BTFSC
                              _{\rm Z}
               GOTO
                              RES024
               CLRF
                              AARGB1
               BSF
                              TEMP,3
NORM2424
               MOVF
                              TEMP,W
               SUBWF
                              EXP,F
                              _Z
               BTFSS
               BTFSS
                              _C
                              SETFUN24
               GOTO
               BCF
                                               ; clear carry bit
                              _C
NORM2424A
               BTFSC
                              AARGB0,MSB
                                                ; if MSB=1, normalization done
               COTO
                              FIXSIGN24
               RLF
                                                ; otherwise, shift left and
                              AARGB1,F
               RLF
                              AARGB0,F
                                                ; decrement EXP
               DECFSZ
                              EXP,F
               GOTO
                              NORM2424A
                                                ; underflow if EXP=0
               GOTO
                              SETFUN24
FIXSIGN24
               BTFSS
                              SIGN, MSB
               BCF
                              AARGB0,MSB
                                                ; clear explicit MSB if positive
               RETLW
RES024
               CLRF
                                                ; result equals zero
                              AARGB0
               CLRF
                              AARGB1
               CLRF
                              AARGB2
                                                ; clear extended byte
               CLRF
                              EXP
               RETLW
Integer to float conversion
       Input: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2
             CALL
       Use:
                    FL02424
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- FLOAT( AARG )
       Max Timing:
                      14+94 = 108 \text{ clks}
                                                     RND = 0
                      14+103 = 117 \text{ clks}
                                                     RND = 1, SAT = 0
                      14+109 = 123 \text{ clks}
                                                     RND = 1, SAT = 1
```

```
6+28 = 34 \text{ clks}
        Min Timing:
                                                            AARG = 0
                         6+22 = 28 \text{ clks}
                                                            DM: 7
        PM: 14+51 = 65
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
                                 AARGB1,F
                 COME
                 COMF
                                  AARGB0,F
                                 AARGB2,F
                 INCF
                 BTFSC
                                  _{\rm Z}
                                  AARGB1,F
                 INCF
                 BTFSC
                                  _{\rm Z}
                                  AARGB0,F
                 INCF
                 BSF
                                  SIGN, MSB
        Normalization routine
        Input: 32 bit unnormalized floating point number in AEXP, AARGBO, AARGB1,
;
                 AARGB2, with sign in SIGN, MSB
                CALL
                         NRM3224
        Output: 24 bit normalized floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- NORMALIZE( AARG )
                         21+6+7*8+7+4 = 94 \text{ clks} \text{ RND} = 0
;
        Max Timing:
                         21+6+7*8+20+4 = 103 \text{ clks} RND = 1, SAT = 0
;
                         21+6+7*8+19+11 = 109 \text{ clks}
                                                           RND = 1, SAT = 1
;
        Min Timing:
                         22+6 = 28 \text{ clks}
                                                            AARG = 0
                         5+9+4+4 = 22 \text{ clks}
        PM: 51
                                                            DM: 7
                                  TEMP
NRM3224
                 CLRF
                                                          ; clear exponent decrement
                 MOVF
                                  AARGB0,W
                                                          ; test if highbyte=0
                 BTFSS
                                  _{\rm Z}
                 GOTO
                                  NORM3224
                                                          ; if so, shift 8 bits by move
                 MOVF
                                  AARGB1,W
                 MOVWF
                                  AARGB0
                 MOVF
                                  AARGB2,W
                 MOVWF
                                  AARGB1
                 CLRF
                                  AARGB2
                 BSF
                                  TEMP, 3
                                                          ; increase decrement by 8
                                  AARGB0,W
                                                         ; test if highbyte=0
                 MOVF
                 BTFSS
                                  _{\rm Z}
                                  NORM3224
                 COTO
                 MOVF
                                  AARGB1,W
                                                          ; if so, shift 8 bits by move
                 MOVWF
                                  AARGB0
                 CLRF
                                  AARGB1
                 BCF
                                  TEMP,3
                                                          ; increase decrement by 8
                                  TEMP,4
                 BSF
```

```
AARGB0,W
              MOVF
                                                 ; if highbyte=0, result=0
              BTFSC
                            _{\rm Z}
                            RES024
              GOTO
NORM3224
              MOVF
                            TEMP, W
              SUBWF
                            EXP,F
              BTFSS
                            _{\rm Z}
              BTFSS
                            _C
                            SETFUN24
              COTO
              BCF
                                                 ; clear carry bit
                            _C
NORM3224A
              BTFSC
                            AARGB0,MSB
                                                 ; if MSB=1, normalization done
              COTO
                            NRMRND3224
              RLF
                            AARGB2,F
                                                 ; otherwise, shift left and
                                                 ; decrement EXP
              RLF
                            AARGB1,F
              RLF
                            AARGB0,F
              DECFSZ
                            EXP,F
              GOTO
                            NORM3224A
                                                 ; underflow if EXP=0
              GOTO
                            SETFUN24
NRMRND3224
              BTFSC
                            FPFLAGS, RND
                            AARGB1,LSB
              BTFSS
              GOTO
                            FIXSIGN24
              BTFSS
                                                 ; round if next bit is set
                            AARGB2,MSB
              GOTO
                            FIXSIGN24
                            AARGB1,F
              TNCF
              BTFSC
                            _{\rm Z}
              INCF
                            AARGB0,F
              BTFSS
                            _{\rm Z}
                                                 ; has rounding caused carryout?
                            FIXSIGN24
              COTO
                            AARGB0,F
                                                 ; if so, right shift
              RRF
              RRF
                            AARGB1,F
              INCF
                            EXP,F
                                                 ; check for overflow
              BTFSC
                            _{\rm Z}
                            SETFOV24
              GOTO
              COTO
                            FIXSIGN24
Float to integer conversion
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                   INT2416
                                   or
       Use:
              CALL
                                         CALL
                                                INT24
      Output: 16 bit 2's complement integer right justified in AARGBO, AARGB1
      Result: AARG <-- INT( AARG )
                     29+6*6+5+13 = 83 clks
       Max Timing:
                                                 RND = 0
                     29+6*6+5+19 = 89 \text{ clks}
                                                  RND = 1, SAT = 0
                     29+6*6+5+22 = 92 \text{ clks}
                                                  RND = 1, SAT = 1
      Min Timing:
                     18+5+7 = 30 \text{ clks}
      PM: 63
                                                  DM: 6
INT2416
INT24
                            EXP,W
              MOVF
                                                  ; 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	
			1 1
	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
	MOVE	AARGB0,W AARGB1	
	MOVWF CLRF	AARGB1 AARGB0	
	CLRF	AARGBU	
	MOVLW	8	; do byte shift if EXP >= 8
	SUBWF	EXP,W	/ do byte shift if EM /- 0
	BTFSS	_C	
	GOTO	TSHIFT2416	
	MOVWF	EXP	
	RLF	AARGB1,F	; rotate next bit for rounding
	CLRF	AARGB1	, rocace none are rer reamaing
	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	INT24160K	
	BTFSS	_C	; round if next bit is set
	GOTO	INT2416OK	
	INCF	AARGB1,F	
	BTFSC	_Z	
	INCF	AARGB0,F	
	DTECC	AARGB0,MSB	; test for overflow
	BTFSC	SETIOV16	/ test for overflow
	GOTO	SEITOATO	
INT24160K	BTFSS	SIGN, MSB	; if sign bit set, negate
111121100N	RETLW	0	, it bight bit bee, hegate
	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
                                             ; complement 16 bit integer
             BTFSS
                         SIGN, MSB
             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, AARGBO, AARGB1
          CALL INT2424
      Use:
      Output: 24 bit 2's complement integer right justified in AARGBO, AARGB1, AARGB2
      Result: AARG <-- INT( AARG )
                   41+6*7+6+16 = 105 \text{ clks}
                                                  RND = 0
      Max Timing:
                   41+6*7+6+24 = 113 \text{ clks}
                                                  RND = 1, SAT = 0
                   41+6*7+6+26 = 115 \text{ clks}
                                                   RND = 1, SAT = 1
      Min Timing: 5 clks
      PM: 82
                                                   DM: 6
;------
INT2424
             CLRF
                          AARGB2
             MOVF
                          EXP,W
                                             ; test for zero argument
             BTFSC
                          _{\rm Z}
             RETLW
                          0x00
             MOVF
                          AARGB0,W
                                             ; save sign in SIGN
             MOVWF
                          SIGN
                          AARGB0,MSB
                                             ; make MSB explicit
             BSF
             MOVLW
                          EXPBIAS+D'23'
                                             ; remove bias from EXP
             SUBWF
                          EXP,F
             BTFSS
                          EXP,MSB
                          SETIOV24
             GOTO
             COMF
                          EXP,F
             INCF
                          EXP,F
                                              ; do byte shift if EXP >= 8
             MOVLW
                          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
                          AARGB1
             MOVWF
```

	CLRF	AARGB0	
	MOVLW SUBWF	8 EXP,W	; do another byte shift if EXP >= 8
	BTFSS	_C	
	GOTO MOVWF	TSHIFT2424 EXP	
	RLF	AARGB2,F	; rotate next bit for rounding
	MOVF	AARGB1,W	
	MOVWF CLRF	AARGB2 AARGB1	
	Chitr	AARODI	
	MOVLW	8	; do another byte shift if EXP >= 8
	SUBWF	EXP,W	
	BTFSS GOTO	_C TSHIFT2424	
	MOVWF	EXP	
	RLF	AARGB2,F	; rotate next bit for rounding
	CLRF MOVF	AARGB2 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 RRF	AARGB1,F AARGB2,F	
	DECFSZ	EXP,F	
	GOTO	SHIFT2424	
SHIFT2424OK	BTFSC	FPFLAGS,RND	
	BTFSS	AARGB2,LSB	
	GOTO	INT2424OK	
	BTFSS GOTO	_C INT2424OK	
	INCF	AARGB2,F	
	BTFSC	_Z	
	INCF	AARGB1,F	
	BTFSC INCF	_Z AARGB0,F	
	BTFSC	AARGBO,MSB	; test for overflow
	GOTO	SETIOV24	
INT24240K	BTFSS	SIGN, MSB	; if sign bit set, negate
	RETLW	0	
	COME	AARGBO,F	
	COMF COMF	AARGB1,F AARGB2,F	
	INCF	AARGB2,F	
	BTFSC	_Z	
	INCF BTFSC	AARGB1,F _Z	
	INCF	AARGB0,F	
	RETLW	0	
IRES024	CLRF	AARGB0	; integer result equals zero
	CLRF	AARGB1	3
	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
              MOVT W
                            OxFF
                                                  ; SIGN = 0, 0x 7F FF FF
              MOVWF
                            AARGB0
              MOVWF
                            AARGB1
                                                  ; SIGN = 1, 0x 80 00 00
              MOVWF
                            AARGB2
                            SIGN, F
              RLF
              RRF
                            AARGB0,F
              RETLW
                                                  ; return error code in WREG
                            0xFF
Floating Point Multiply
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
              CALL
                     FPM24
       Use:
       Output: 24 bit floating point product in AEXP, AARGBO, AARGB1
      Result: AARG <-- AARG * BARG
                     25+15*16+15+18 = 298 clks
                                                 RND = 0
       Max Timing:
                     25+15*16+15+29 = 309 \text{ clks}
                                                  RND = 1, SAT = 0
                     25+15*16+15+33 = 313 \text{ clks}
                                                  RND = 1, SAT = 1
                     6+5 = 11 \text{ clks}
                                                  AARG * BARG = 0
       Min Timing:
                     24+15*11+14+15 = 218 clks
      PM: 80
                                                  DM: 11
FPM24
              MOVF
                            AEXP,W
                                                 ; test for zero arguments
              BTFSS
                            _{\rm Z}
                            BEXP,W
              MOVF
              BTFSC
                            _{\rm Z}
                            RES024
              GOTO
M24BNE0
              MOVF
                            AARGB0,W
              XORWE
                            BARGB0,W
              MOVWF
                            SIGN
                                                  ; save sign in SIGN
              MOVF
                            BEXP, W
                            EXP,F
              ADDWF
                            EXPBIAS-1
              MOVLW
              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
	110 1 112	12111	, initializa dedirect
MLOOP24	BTFSS	AARGB3,LSB	; test next bit
MLOOPZ4			/ test next bit
	GOTO	MNOADD24	
1/30004		D1D6D1	
MADD24	MOVF	BARGB1,W	
	ADDWF	AARGB1,F	
	MOVF	BARGB0,W	
	BTFSC	_C	
	INCFSZ	BARGB0,W	
	ADDWF	AARGBO,F	
	ADDWI	AARODO , I	
MNOADD24	DDE	AADGDO E	
MINOADD24	RRF	AARGBO,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	, check for postnormalization
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	EXP,F	
MROUND24	BTFSC	FPFLAGS, RND	
	BTFSS	AARGB1,LSB	
	GOTO	MUL24OK	
			; round if next bit is set
	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	MUL240K	
	RRF	AARGBO,F	; if so, right shift
			, ii so, light shilt
	RRF	AARGB1,F	
	INCF	EXP,F	
	BTFSC	_Z	; check for overflow
	GOTO	SETFOV24	
MUL240K	BTFSS	SIGN, MSB	
	BCF	AARGB0,MSB	; clear explicit MSB if positive
	RETLW	0	-
	••		
SETFOV24	BSF	FPFLAGS, FOV	; set floating point underflag
O V Z I			; test for saturation
	BTFSS	FPFLAGS, SAT	
	RETLW	0xFF	; return error code in WREG
		0 ==	
	MOVLW	0xFF	
	MOVWF	AEXP	; saturate to largest floating
	MOVWF	AARGB0	; point number = 0x FF 7F FF
	MOVWF	AARGB1	; modulo the appropriate sign bit
			-

```
SIGN, F
                RLF
                RRF
                                AARGB0,F
                RETLW
                                0xFF
                                                         ; return error code in WREG
        Floating Point Divide
        Input: 24 bit floating point dividend in AEXP, AARGBO, AARGB1
                24 bit floating point divisor in BEXP, BARGBO, BARGB1
        Use:
                CALL
                        FPD24
        Output: 24 bit floating point quotient in AEXP, AARGBO, AARGB1
        Result: AARG <-- AARG / BARG
                        32+13+15*26+25+12 = 472 clks
                                                       RND = 0
        Max Timing:
                        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 \text{ clks}
       PM: 120
                                                         DM: 11
FPD24
                MOVF
                                BEXP.W
                                                        ; test for divide by zero
                                _Z
                BTFSC
                GOTO
                                SETFDZ24
                MOVF
                                AEXP,W
                BTFSC
                                _{\rm Z}
                                RES024
                GOTO
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
                                BARGB0,W
                INCFSZ
                SUBWF
                                AARGB2, f
                CLRF
                                AARGB2
                CLRF
                                AARGB3
                BTFSS
                                _C
                GOTO
                                DALIGN240K
                BCF
                                 _C
                                                         ; align if necessary
                                AARGB0,F
                RRF
                                AARGB1,F
                RRF
                                AARGB2,F
                                0x01
                MOVLW
```

	MOVWF	TEMP	; save align increment
DALIGN240K	MOVF	BEXP,W	; compare AEXP and BEXP
	SUBWF	EXP,F	
	BTFSS	_C	
	GOTO	ALTB24	
3 CED 0 4	MOLITE	EVEDING 1	
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
	G010	SEIFONZT	/ Set underliow riag
DARGOK24	MOVLW	D'16'	; initialize counter
DIMOGRA I	MOVWF	TEMPB1	, inicializa comical
	110 V W1	THEFT	
DLOOP24	RLF	AARGB3,F	; left shift
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	RLF	TEMP, F	
	KHI	IEMF, P	
	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	
	G010	DORZ4	
DREST24	MOVF	BARGB1,W	; restore if necessary
	ADDWF	AARGB1,F	•
	MOVF	BARGB0,W	
	BTFSC	_C	
	INCF	BARGB0,W	
	ADDWF	AARGB0,F	
	BCF	AARGB3,LSB	
	201	11110000 / 200	
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
                            _C
              BTFSC
              INCF
                            AARGB2,F
              BTFSS
                            _{\rm Z}
                                                  ; test if rounding caused carryout
                            DIV240K
              COTO
              RRF
                            AARGB2,F
              RRF
                            AARGB3,F
              INCF
                            EXP,F
              BTFSC
                                                  ; test for overflow
                            _{\rm Z}
              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
SETFUN24
              BSF
                            FPFLAGS, FUN
                                                  ; set floating point underflag
                                                  ; test for saturation
              BTFSS
                            FPFLAGS, SAT
              RETLW
                                                  ; return error code in WREG
                            TTx0
              MOVLW
                            0x01
                                                  ; saturate to smallest floating
              MOVWF
                            AEXP
                                                  ; point number = 0x 01 00 00
              CLRF
                            AARGB0
                                                  ; modulo the appropriate sign bit
              CLRF
                            AARGB1
                            SIGN, F
              RLF
              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, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
       Use:
              CALL FPS24
       Output: 24 bit floating point sum in AEXP, AARGBO, AARGB1
       Result: AARG <-- AARG - BARG
       Max Timing:
                     2+197 = 199 \text{ clks}
                                                  RND = 0
                     2+208 = 210 \text{ clks}
                                                  RND = 1, SAT = 0
                     2+213 = 215 \text{ clks}
                                                  RND = 1, SAT = 1
      Min Timing:
                    2+12 = 14 \text{ clks}
```

```
PM: 2+112 = 114
                                                 DM: 11
FPS24
              MOVLW
                            0x80
              XORWE
                            BARGB0,F
Floating Point Add
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
;
              24 bit floating point number in BEXP, BARGBO, BARGB1
       Use:
             CALL FPA24
;
       Output: 24 bit floating point sum in AEXP, AARGBO, AARGB1
      Result: AARG <-- AARG - BARG
;
                     25+28+6*6+5+31+72 = 197 clks
;
       Max Timing:
                                                 RND = 0
                     25+28+6*6+5+42+72 = 208 \text{ clks}
                                                 RND = 1, SAT = 0
;
                     25+28+6*6+5+42+77 = 213 \text{ clks}
                                                 RND = 1, SAT = 1
      Min Timing:
                    8+4 = 12 \text{ clks}
       PM: 112
                                                 DM: 11
;------
FPA24
              MOVF
                            AARGB0,W
                                                 ; exclusive or of signs in TEMP
              XORWF
                            BARGB0,W
                            TEMP
              MOVWF
              CLRF
                            AARGB2
                                                 ; clear extended byte
              CLRF
                            BARGB2
              MOVF
                            AEXP,W
                                                 ; use AARG if AEXP >= BEXP
              SUBWE
                            BEXP,W
              BTFSS
                            _C
              GOTO
                            USEA24
              MOVF
                            BEXP,W
                                                 ; use BARG if AEXP < BEXP
              MOVWF
                            AARGB4
                                                 ; therefore, swap AARG and BARG
                            AEXP,W
              MOVF
              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
                            _{\rm Z}
                            0x00
              RETLW
```

	MOVF	AARGB0,W	
	MOVWF	SIGN	; save sign in SIGN
	BSF	AARGB0,MSB	; make MSB's explicit
	BSF	BARGB0,MSB	, mane hob b dipilote
	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	/ II blim >= 0, do byte shire
	MOVWF	BEXP	
	MOVF	BARGB1,W	; keep for postnormalization
	MOVWF	BARGB2	· ····································
	MOVF	BARGB0,W	
	MOVWF	BARGB1	
	CLRF	BARGB0	
	MOVLW	8	
	SUBWF	BEXP,W	10
	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
112001 22 1	RRF	BARGB0,F	r right bhile by bhil
	RRF	BARGB1,F	
	RRF	BARGB2,F	
	DECFSZ	BEXP,F	
	GOTO	ALOOPB24	
A T T CAMED 0 4	DEFICA	TEMP MAD	
ALIGNED24	BTFSS	TEMP, MSB	; negate if signs opposite
	GOTO	AOK24	
	COMF COMF	BARGB2,F BARGB1,F	
	COMF	BARGBO,F	
	INCF	BARGB2,F	
	BTFSC	_Z	
	INCF	BARGB1,F	
	BTFSC	_Z	
	INCF	BARGB0,F	
AOK24	MOTTE	DADGDO	
	MOVF	BARGB2,W	
	ADDWF	AARGB2,F	
	MOVF	BARGB1,W	
	BTFSC	_C BARCR1 W	
	INCFSZ	BARGB1,W	
	ADDWF MOVF	AARGB1,F BARGB0,W	
	BTFSC	_C	
	INCFSZ	_C BARGB0,W	
	ADDWF	AARGB0, F	
		•	
	BTFSC	TEMP,MSB	

	GOTO BTFSS GOTO	ACOMP24 _C NRMRND3224	
	RRF RRF RRF INCFSZ GOTO GOTO	AARGB0,F AARGB1,F AARGB2,F AEXP,F NRMRND3224 SETFOV24	; shift right and increment EXP
ACOMP24	BTFSC GOTO	_C NRM3224	; normalize and fix sign
	COMF COMF COMF INCF BTFSC INCF BTFSC INCF	AARGB2,F AARGB1,F AARGB0,F AARGB2,F _Z AARGB1,F _Z AARGB1,F	; negate, toggle sign bit and ; then normalize
	MOVLW XORWF GOTO	0x80 SIGN,F 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
FL01624
                16 bit integer to 24 bit floating point conversion
FLO24
Max Timing:
                            RND
                                 1
                         48
                                 48
           SAT
                         55
                                 55
NRM2424 24 bit normalization of unnormalized 24 bit floating point numbers
NRM24
Max Timing:
                           RND
                         39
                                 39
           SAT
                1
                         46
                                 46
                24 bit floating point to 16 bit integer conversion
INT2416
TNT24
Max Timing:
                            RND
                         0
                                 1
                         58
                                 66
           SAT
                         58
                                 70
FL02424
                24 bit integer to 24 bit floating point conversion
Max Timing:
                            RND
                                 77
                0
                         64
           SAT
                                 83
```

```
NRM3224 24 bit normalization of unnormalized 32 bit floating point numbers
Max Timing:
                            RND
                        52
                                 65
           SAT
                        52
                                 71
INT2424
                24 bit floating point to 24 bit integer conversion
Max Timing:
                            RND
                        0
                                 1
           SAT
                        71
                                 82
                24 bit floating point add
FPA24
Max Timing:
                            RND
                                 1
                        133
                                 146
           SAT
                        133
                                 152
FPS24
                24 bit floating point subtract
Max Timing:
                            RND
                                 1
                        134
                                 147
           SAT
                        134
                                 153
                24 bit floating point multiply
FPM24
Max Timing:
                            RND
                0
                        60
                                 72
           SAT
                                 79
FPD24
                24 bit floating point divide
Max Timing:
                            RND
                        0
                                 1
                                 185
                        176
           SAT
                        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
                   FL01624
             CALL
                  FLO24
;
      Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
      Result: AARG <-- FLOAT( AARG )
                  9+39 = 48 \text{ clks}
                                       SAT = 0
      Max Timing:
                   9+45 = 54 \text{ clks}
                                       SAT = 1
                   6+15 = 21 \text{ clks}
                                       AARG = 0
      Min Timing:
                   6+20 = 26 \text{ clks}
      PM: 9+68 = 77
                                       DM: 6
FI-01624
FLO24
             MOVLW
                         D'15'+EXPBIAS ; initialize exponent and add bias
             MOVWF
             MOVPF
                         AARGBO,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, AARGBO, AARGB1,
              with sign in SIGN, MSB.
       Use:
              CALL
                      NRM2424
              CALL
                      NRM24
       Output: 24 bit normalized floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- NORMALIZE( AARG )
                                                    SAT = 0
                     3+12+16+8 = 39 clks
       Max Timing:
                      3+12+16+15 = 46 \text{ clks}
                                                    SAT = 1
       Min Timing:
                     10+5 = 15 \text{ clks}
                                                    AARG = 0
                      3+5+4+8 = 20 \text{ clks}
       PM: 68
                                                    DM: 6
;------
NRM2424
NRM24
              CLRF
                             TEMP,W
                                                    ; clear exponent decrement
                             AARGB0
                                                    ; test if highbyte=0
              CPESGT
              GOTO
                             NRM2424A
TNIB2424
              MOVLW
                             0xF0
                                                    ; test if highnibble=0
              ANDWF
                             AARGB0,W
              TSTFSZ
                             WREG
                             NORM2424
              COTO
                                                    ; if so, shift 4 bits
              SWAPF
                             AARGB0,F
              SWAPF
                             AARGB1,W
              ANDLW
                             0x0F
              ADDWF
                             AARGB0,F
              SWAPF
                             AARGB1,W
              ANDLW
                             0xF0
              MOVPF
                             WREG, AARGB1
                             TEMP, 2
              BSF
                                                    ; increase decrement by 4
NORM2424
              BCF
                             _C
                                                    ; clear carry bit
              BTFSC
                             AARGB0,MSB
                                                    ; if MSB=1, normalization done
              COTO
                             TNORMUN2424
              RLCF
                             AARGB1,F
                                                    ; otherwise, shift left and
              RLCF
                             AARGB0,F
                                                    ; increment decrement
              INCF
                             TEMP,F
                             AARGB0,MSB
              BTFSC
              GOTO
                             TNORMUN2424
              RLCF
                             AARGB1,F
              RLCF
                             AARGB0,F
              INCF
                             TEMP,F
                                                   ; since highnibble != 0, at most
              BTFSC
                             AARGB0,MSB
              GOTO
                             TNORMUN2424
                                                    ; 3 left shifts are required
              RLCF
                             AARGB1,F
              RLCF
                             AARGB0,F
              INCF
                             TEMP,F
                             TEMP, WREG
TNORMUN2424
              MOVFP
                                                   ; if EXP <= decrement in TEMP,
                                                   ; floating point underflow has
              CPFSGT
                             EXP
              GOTO
                             SETFUN24
                                                   ; occured
              SUBWF
                             EXP,F
                                                    ; otherwise, compute EXP
```

```
FIXSIGN24
               BTFSS
                               SIGN, MSB
               BCF
                               AARGB0,MSB
                                                      ; clear explicit MSB if positive
               RETLW
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
                               RES024
               COTO
               MOVLW
                               0xF0
                                                      ; test if highnibble=0
               ANDWF
                               AARGB0,W
               TSTFSZ
                               WREG
                               NORM2424A
               COTO
               SWAPF
                               AARGB0,F
                                                      ; if so, shift 4 bits
               BSF
                               TEMP, 2
                                                      ; increase decrement by 4
NORM2424A
                               _C
               BCF
                                                      ; clear carry bit
               BTFSC
                               AARGB0,MSB
                                                      ; if MSB=1, normalization done
               GOTO
                               TNORMUN2424
                               AARGB0,F
                                                      ; otherwise, shift left and
               RLCF
               INCF
                               TEMP,F
                                                      ; increment decrement
               BTFSC
                               AARGB0,MSB
               GOTO
                               TNORMUN2424
                               AARGB0,F
               RLCF
               INCF
                               TEMP,F
               BTFSC
                               AARGB0,MSB
                                                      ; since highnibble != 0, at most
               GOTO
                               TNORMUN2424
                                                      ; 3 left shifts are required
               RLCF
                               AARGB0,F
                               TEMP,F
               TNCF
                               TNORMUN2424
               COTO
RES024
               CLRF
                               AARGB0,F
                                                     ; result equals zero
               CLRF
                               AARGB1,F
               CLRF
                               AARGB2,F
                                                      ; clear extended byte
               CLRF
                               EXP,F
               RETLW
Integer to float conversion
       Input: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2
       Use:
               CALL
                     FL02424
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- FLOAT( AARG )
;
       Max Timing:
                       12+52 = 64 \text{ clks}
                                                      RND = 0
                       12+65 = 77 \text{ clks}
                                                      RND = 1, SAT = 0
                       12+71 = 83 \text{ clks}
                                                      RND = 1, SAT = 1
;
                       6+24 = 30 \text{ clks}
;
       Min Timing:
                                                      AARG = 0
                       6+24 = 30 \text{ clks}
       PM: 12+121 = 133
                                                      DM: 7
```

```
FL02424
               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
                              AARGB2,F
               COMF
               COMF
                              AARGB1,F
               COMF
                              AARGB0,F
               INCF
                              AARGB2,F
               ADDWFC
                              AARGB1,F
                              AARGB0,F
               ADDWFC
Normalization routine
       Input: 32 bit unnormalized floating point number in AEXP, AARGBO, AARGB1,
               AARGB2 with sign in SIGN, MSB.
       Use:
               CALL
                      NRM3224
       Output: 24 bit normalized floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- NORMALIZE( AARG )
       Max Timing:
                       21+19+12 = 52 \text{ clks}
                                                      RND = 0
                       21+19+25 = 65 \text{ clks}
                                                      RND = 1, SAT = 0
                                                      RND = 1, SAT = 1
                       21+19+31 = 71 \text{ clks}
       Min Timing:
                       4+7+7+5 = 24 \text{ clks}
                                                      AARG = 0
                       3+5+4+8+4 = 24 \text{ 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
               ANDI W
                               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
                               TEMP,2
               BSF
                                                      ; increase decrement by 4
NORM3224
               BCF
                                                      ; clear carry bit
                               _C
                               AARGB0,MSB
                                                      ; if MSB=1, normalization done
               BTFSC
```

	GOTO	TNORMUN3224	
	RLCF	AARGB2,F	; otherwise, shift left and
	RLCF	AARGB1,F	; increment decrement
	RLCF	AARGBO,F	
	INCF	TEMP,F	
	BTFSC	AARGBO,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
		AARGB2,F	/ 3 left shifts are required
	RLCF	•	
	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	7 15 1.65 7 011001
		-	. got garry for rounding
	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	AARGBO,F	
	BTFSS	_C	; has rounding caused carryout?
	GOTO	FIXSIGN24	•
	RRCF	AARGBO,F	; if so, right shift
	RRCF		/ II 50, IIghe Shile
		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	3
TNIB3224A	MOVLW	0xF0	; test if highnibble=0
TNIB3224A	MOVLW ANDWF	0xF0 AARGBO.W	; test if highnibble=0
TNIB3224A	ANDWF	AARGB0,W	; test if highnibble=0
TNIB3224A	ANDWF TSTFSZ	AARGB0,W WREG	; test if highnibble=0
TNIB3224A	ANDWF TSTFSZ GOTO	AARGB0,W WREG NORM3224A	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF	AARGB0,W WREG NORM3224A AARGB0,F	<pre>; test if highnibble=0 ; if so, shift 4 bits</pre>
TNIB3224A	ANDWF TSTFSZ GOTO	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF	AARGB0,W WREG NORM3224A AARGB0,F	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF SWAPF ANDLW	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F AARGB1,W 0xF0	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F	
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF SWAPF ANDLW MOVPF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F AARGB1,W 0xF0 WREG,AARGB1	; if so, shift 4 bits
TNIB3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF SWAPF ANDLW	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F AARGB1,W 0xF0	
	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF SWAPF ANDLW MOVPF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F AARGB1,W 0xF0 WREG,AARGB1	; if so, shift 4 bits ; increase decrement by 4
TNIB3224A NORM3224A	ANDWF TSTFSZ GOTO SWAPF SWAPF ANDLW ADDWF SWAPF ANDLW MOVPF	AARGB0,W WREG NORM3224A AARGB0,F AARGB1,W 0x0F AARGB0,F AARGB1,W 0xF0 WREG,AARGB1	; if so, shift 4 bits

```
BTFSC
                            AARGB0,MSB
                                                  ; if MSB=1, normalization done
              GOTO
                             TNORMUN3224
                                                  ; otherwise, shift left and
              RLCF
                            AARGB1,F
              RLCF
                            AARGB0,F
                                                  ; increment decrement
              TNCF
                            TEMP.F
              BTFSC
                            AARGB0, MSB
              GOTO
                            TNORMUN3224
              RLCF
                            AARGB1,F
              RLCF
                            AARGB0,F
              INCF
                            TEMP,F
                            AARGB0,MSB
                                                  ; since highnibble != 0, at most
              BTFSC
              GOTO
                            TNORMUN3224
                                                  ; 3 left shifts are required
              RLCF
                            AARGB1,F
                            AARGB0,F
              RLCF
              INCF
                             TEMP,F
                             TNORMUN3224
              GOTO
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
                            RES024
              GOTO
              MOVLW
                                                  ; test if highnibble=0
TNIB3224B
                             0xF0
              ANDWF
                            AARGB0,W
              TSTFS7
                            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
              TNCF
                            TEMP,F
              BTFSC
                            AARGB0,MSB
                                                  ; since highnibble != 0, at most
                                                  ; 3 left shifts are required
              GOTO
                            TNORMUN3224
              RLCF
                            AARGB0,F
              INCF
                            TEMP,F
                            TNORMUN3224
              GOTO
Float to integer conversion
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                     INT2416
       Use:
              CALL
              CALL
                     INT24
       Output: 16 bit 2's complement integer right justified in AARGBO, AARGBO
       Result: AARG <-- INT( AARG )
       Max Timing:
                     10+36+12 = 58 \text{ clks}
                                                  RND = 0
                     10+36+20 = 66 \text{ clks}
                                                  RND = 1, SAT = 0
                     10+36+24 = 70 \text{ 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
                                                         ; if >= 15, integer overflow
                BTFSS
                                WREG, MSB
                                                         ; will occur
                                SETIOV2416
                GOTO
                NEGW
                                EXP,F
                MOVLW
                                                         ; do byte shift if EXP >= 8
                                EXP
                CPFSGT
                GOTO
                                SNIB2416
                                EXP,F
                                                         ; EXP = EXP - 7
                SUBWF
                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
                                SNIB2416A
                COTO
                                                         ; EXP = EXP - 7
                SUBWF
                                EXP,F
                MOVFP
                                AARGB1,AARGB2
                                                         ; save for rounding
                CLRF
                                AARGB1,F
                                EXP,F
                                                         ; EXP = EXP - 1
                DCFSNZ
                COTO
                                SHIFT2416OK
                                                         ; shift completed if EXP = 0
SNIB2416B
                MOVLW
                                                         ; do nibble shift if EXP >= 4
                CPFSGT
                                EXP
                COTO
                                SHIFT2416B
                SWAPF
                                AARGB2,W
                                0x0F
                ANDLW
                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
                                                         ; shift completed if EXP = 0
                                SHIFT2416OK
                GOTO
                BCF
                                 C
                RRCF
                                AARGB2,F
                DCFSNZ
                                EXP,F
                GOTO
                                SHIFT2416OK
                                                         ; shift completed if EXP = 0
                BCF
                                 _C
                RRCF
                                AARGB2,F
                                SHIFT2416OK
                GOTO
SNTB2416A
                MOVLW
                                 3
                                                         ; do nibble shift if EXP >= 4
                CPFSGT
                                EXP
                GOTO
                                SHIFT2416A
                                EXP,F
                                                         ; EXP = EXP - 3
                SUBWF
                SWAPF
                                AARGB1,W
                                                         ; save for rounding
                MOVFP
                                WREG, AARGB2
```

	ANDLW	0x0F	
	MOVPF	WREG, AARGB1	
	DCFSNZ	EXP, F	; EXP = EXP - 1
	GOTO	SHIFT2416OK	; shift completed if EXP = 0
	0010	BIIII 12 11 00K	/ Shire completed if EMI - 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
SNIBZ4IO	CPFSGT	EXP	/ do hippie shirt it EAP >- 4
	GOTO	SHIFT2416	
	SUBWF	EXP,F	; EXP = EXP - 3
	SWAPF	AARGB1,W	/ EAF - EAF - 5
	MOVFP	WREG, AARGB2	; save for rounding
	ANDLW	0x0F	, pare for rounding
	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	5
	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	INT24160K	
	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	
	ADDWEC	AARGB1,F	
	ADDWFC	AARGBO,F	
	BTFSC	AARGB0,MSB	

```
GOTO
                                SETIOV2416
INT2416OK
                BTFSS
                                SIGN, MSB
                                                         ; if sign bit set, negate
                RETLW
                COME
                                AARGB1,F
                COME
                                AARGRO.F
                                AARGB1,F
                INFSNZ
                INCF
                                AARGB0,F
                RETLW
SETIOV2416
                BSF
                                FPFLAGS, IOV
                                                        ; set integer overflow flag
                BTFSS
                                                         ; test for saturation
                                FPFLAGS, SAT
                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, 0 \times 7 F FF
                MOVPF
                                AARGB0, AARGB1
                                                         ; SIGN = 1, 0x 80 00
                RLCF
                                SIGN, F
                                AARGB0,F
                RRCF
                RETLW
                                0xFF
                                                         ; return error code in WREG
        Float to integer conversion
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              CALL
                      INT2424
;
        Use:
        Output: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2
        Result: AARG <-- INT( AARG )
                        11+45+15 = 71 \text{ clks}
                                                       RND = 0
        Max Timing:
                        11+45+23 = 79 \text{ clks}
                                                         RND = 1, SAT = 0
                        11+45+26 = 82 \text{ clks}
                                                         RND = 1, SAT = 1
;
       Min Timing:
                       4 clks
                                                         DM: 7
        PM: 185
INT2424
                CLRF
                                AARGB2,W
                CPFSGT
                                EXP
                                                         ; test for zero argument
                RETLW
                                0 \times 00
                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
                                7
                MOVLW
                                                         ; do byte shift if EXP >= 8
                                EXP
                CPFSGT
                                SNIB2424
                GOTO
                SUBWF
                                EXP,F
                                                         ; EXP = EXP - 7
                                                         ; save for rounding
                MOVFP
                                AARGB2,AARGB3
```

	MOVFP MOVFP CLRF	AARGB1,AARGB2 AARGB0,AARGB1 AARGB0,F	
	DCFSNZ	EXP,F	; EXP = EXP - 1
	GOTO	SHIFT2424OK	
	GOTO	SH1F124240K	; 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
	G010	SHIF IZ IZ IOK	/ SHITE COMPLETED II EAF - 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
	GOTO	SHIF 124240K	, 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	
			· which roundated if HVD 0
	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	3
	GOTO	SHIFT2424OK	; shift completed if EXP = 0
			/ SHILL COMPLETED IT 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
GIVIDE 12 1D	CPFSGT		, do hibbit bhile il bhi , = 1
		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 $EXP = 0$
OUT 1000 40 45	DGE		
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	abift completed if EVD 0
	GOTO	SHIFT2424OK	; shift completed if EXP = 0
	BCF	_C	
	RRCF	AARGB2,F	

	RRCF	AARGB3,F	
	GOTO	SHIFT2424OK	
C)77770 40 47		2	. 1 1111 115, 15 777
SNIB2424A	MOVLW	3	; do nibble shift if EXP >= 4
	CPFSGT	EXP	
	GOTO	SHIFT2424A	; EXP = EXP - 3
	SUBWF	EXP,F	/ EAP = EAP - 3
	SWAPF	AARGB2,W	· gave for rounding
	MOVPF ANDLW	WREG,AARGB3 0x0F	; save for rounding
	MOVPF	WREG, AARGB2	
	MOVPF	WRLG, AARGDZ	
	SWAPF	AARGB1,W	
	ANDLW	0xF0	
	ADDWF	AARGB2,F	
	11111111	1111(052,1	
	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	
	11111111	1111(052,1	
	SWAPF	AARGB1,W	
	ANDLW	0x0F	
	MOVPF	WREG, AARGB1	
		•	
	SWAPF	AARGB0,W	
	ANDLW	0xF0	
	ADDWF	AARGB1,F	
		•	
	SWAPF	AARGB0,W	
	ANDLW	0x0F	
	MOVPF	WREG, AARGBO	
	DCFSNZ	EXP,F	; EXP = EXP - 1
	GOTO	SHIFT2424OK	; shift completed if EXP = 0

```
_C
SHIFT2424
               BCF
                                                      ; 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
                              AARGB2,F
               RRCF
               RRCF
                              AARGB3,F
               DCFSNZ
                              EXP,F
                                                     ; shift completed if EXP = 0
               COTO
                              SHIFT2424OK
               BCF
                              C
               RRCF
                              AARGB0,F
               RRCF
                              AARGB1,F
               RRCF
                              AARGB2,F
               RRCF
                              AARGB3,F
SHIFT2424OK
               BTFSC
                              FPFLAGS, RND
                                                     ; is rounding enabled?
                                                     ; is NSB > 0x80?
                              AARGB3,MSB
               BTFSS
               GOTO
                              INT2424OK
               BSF
                              _C
                                                     ; set carry for rounding
               MOVLW
                              0x80
                              AARGB3
                                                      ; if NSB = 0x80, select even
               CPFSGT
               RRCF
                              AARGB2,W
                                                     ; using lsb in carry
               CLRF
                              WREG, F
               ADDWFC
                              AARGB2,F
               ADDWFC
                              AARGB1,F
                              AARGB0,F
               ADDWFC
               BTFSC
                              AARGB0,MSB
               GOTO
                              SETIOV2424
INT24240K
                                                     ; if sign bit set, negate
               BTFSS
                              SIGN, MSB
               RETLW
               COMF
                              AARGB2,F
               COMF
                              AARGB1,F
               COMF
                              AARGB0,F
               INCF
                              AARGB2,F
               CLRF
                              WREG, F
               ADDWFC
                              AARGB1,F
                              AARGB0,F
               ADDWFC
               RETLW
SETIOV2424
                              FPFLAGS, IOV
                                                     ; set integer overflow flag
               BSF
               BTFSS
                              FPFLAGS, SAT
                                                     ; test for saturation
               RETLW
                              0xFF
                                                     ; return error code in WREG
                                                     ; saturate to largest two's
               CLRF
                              AARGB0,F
               BTFSS
                                                     ; complement 24 bit integer
                              SIGN, MSB
                                                     ; SIGN = 0, 0 \times 7 F FF FF
               SETF
                              AARGB0,F
               MOVPF
                              AARGB0, AARGB1
                                                     ; SIGN = 1, 0x 80 00 00
               MOVPF
                              AARGB0, AARGB2
                              SIGN, F
               RLCF
               RRCF
                              AARGB0,F
               RETLW
                              0xFF
                                                     ; return error code in WREG
Floating Point Multiply
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
               24 bit floating point number in BEXP, BARGBO, BARGB1
```

```
Use:
                CALL
                         FPM24
        Output: 24 bit floating point product in AEXP, AARGBO, AARGB1
;
        Result: AARG <-- AARG * BARG
        Max Timing:
                        19+33+8 = 60 \text{ clks}
                                                           RND = 0
                         19+33+20 = 72 \text{ clks}
                                                           RND = 1, SAT = 0
;
                         19+33+27 = 79 clks
                                                           RND = 1, SAT = 1
        Min Timing:
                         5+5 = 10 \text{ clks}
                                                           AARG * BARG = 0
                         13+5+23+11 = 52 \text{ clks}
        PM: 80
                                                           DM: 11
FPM24
                 CLRF
                                 AARGB2,W
                                                          ; test for zero arguments
                 CPFSEQ
                                 BEXP
                 CPFSGT
                                  AEXP
                 GOTO
                                  RES024
                                  AARGBO, WREG
M24BNE0
                 MOVFP
                 XORWF
                                  BARGB0,W
                 MOVPF
                                  WREG, SIGN
                                                           ; save sign in SIGN
                                  BEXP, WREG
                 MOVFP
                 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
                                                           ; multiply mantissas
                 MOVFP
                                  AARGB1, WREG
                 MULWF
                                  BARGB1
                 MOVPF
                                  PRODH, AARGB2
                 MOVFP
                                  AARGB0, WREG
                 MULWF
                                  BARGB0
                 MOVPF
                                  PRODH, AARGB0
                 MOVPF
                                  PRODL, AARGB1
                 MULWF
                                  BARGB1
                 MOVPF
                                  PRODL, WREG
                                  AARGB2,F
                 ADDWF
                 MOVPF
                                  PRODH, WREG
                 ADDWFC
                                  AARGB1,F
                 CLRF
                                  WREG, F
                 ADDWFC
                                  AARGB0,F
                 MOVFP
                                  TEMPB1, WREG
                                  BARGB0
                 MULWF
```

```
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
                              _{\rm Z}
                              SETFUN24
               COTO
MROUND24
               BTFSC
                              FPFLAGS, RND
                                                     ; is rounding enabled?
                                                     ; is NSB > 0x80?
               BTFSS
                              AARGB2,MSB
               GOTO
                              MUL240K
               BSF
                                                     ; set carry for rounding
                              C
               MOVLW
                              0x80
               CPFSGT
                              AARGB2
                                                      ; if NSB = 0x80, select even
                              AARGB1,W
                                                      ; using lsb in carry
               RRCF
               CLRF
                              WREG, F
               ADDWFC
                              AARGB1,F
               ADDWFC
                              AARGB0,F
                              _C
               BTFSS
                                                      ; has rounding caused carryout?
               GOTO
                              MUL240K
               RRCF
                              AARGB0,F
                                                      ; if so, right shift
               RRCF
                              AARGB1,F
                              EXP,F
                                                      ; test for floating point overflow
               INFSNZ
                              SETFOV24
               COTO
MUL240K
               BTFSS
                              SIGN, MSB
                                                     ; clear explicit MSB if positive
               BCF
                              AARGB0,MSB
               RETLW
SETFOV24
               BSF
                              FPFLAGS, FOV
                                                     ; set floating point underflag
               BTFSS
                              FPFLAGS, SAT
                                                     ; test for saturation
               RETIM
                                                     ; return error code in WREG
                              0xFF
               SETF
                                                     ; saturate to largest floating
                              AEXP,F
               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, AARGBO, AARGB1
               24 bit floating point divisor in BEXP, BARGBO, BARGB1
       Use:
               CALL
                       FPD24
       Output: 24 bit floating point quotient in AEXP, AARGBO, AARGB1
       Result: AARG <-- AARG / BARG
```

```
10+12+30+38+67+19 = 176 \text{ clks}
                                                          RND = 0
;
        Max Timing:
                         10+12+30+38+67+28 = 185 clks
                                                          RND = 1, SAT = 0
                         10+12+30+38+67+35 = 192 \text{ clks}
                                                          RND = 1, SAT = 1
        Min Timing:
                        6+5 = 11 \text{ clks}
                                                          AARG = 0
        PM: 201+256 = 457
                                                          DM: 13
         In addition to those registers defined in MATH17.INC, this routine uses
         {\tt TBLPTRL} and {\tt 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)
                                     TBLPTRL
                MOVWF
                                     _C
                BTFSC
                INCF
                                     TBLPTRH, F
                TABLRD
                                     0,1,TEMPB0
                TLRD
                                     1,TEMPB0
                                     0,TEMPB1
                TLRD
                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
                SWAPE
                                     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
                                     AARGB5
                MOVWF
```

MOVFP

	MOVIF	AARGD4, WREG	
	SUBWFB	TEMPB0,W	
	MOVWF	AARGB4	
	MOVPF	AARGB5,TBLPTRL	
	MOVFP	BARGB0,WREG	
	ANDLW	0x07	
	MOVPF	WREG, TEMPB2	
	MOMED	AADODE MDEG	
	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	
		,	
	MOMED	מסום המסומה	
	MOVFP	TBLPTRL, WREG	
	SUBWF	TEMPB1,F	
	MOVPF	AARGB5,WREG	
	SUBWFB	TEMPB0,F	; F0
		•	
	endm		
	EHAIII		
;			
FPD24	CLRF	TEMPB3,W	; clear exponent modification
	CPFSGT	BEXP	; test for divide by zero
			. cest for arvide by zero
	GOTO	SETFDZ24	
	CPFSGT	AEXP	
	GOTO	RES024	
DЗИБИ⊡О			
D24BNE0	MULLED	A A DC B A MID INC	
	MOVFP	AARGB0,WREG	
	MOVFP XORWF	AARGB0,WREG BARGB0,W	

AARGB4,WREG

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 ADDWFC	WREG,F AARGBO,F	
BTFSC	AARGB0,MSB	
GOTO	DAOK24	
RLCF	AARGB2,F	
RLCF	AARGB1,F	
RLCF	AARGB0,F	
DECF	TEMPB3,F	
MOVFP	BARGB1,AARGB4	; B0 = F0 * B
MOVFP	BARGB1,WREG	
MULWF	TEMPB1	
MOVPF	PRODH,BARGB2	
MOVPF	PRODL, BARGB3	
MOVFP	BARGBO, 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	

DAOK24

	ADDWF MOVPF ADDWFC CLRF ADDWFC BTFSS	BARGB2,F PRODH,WREG BARGB1,F WREG,F BARGB0,F	
	BTFSC	BARGB0,MSB-1	
	GOTO	DBOK24	
	RLCF	BARGB3, F	
	RLCF	BARGB2,F	
	RLCF	BARGB1,F	
	RLCF	BARGBO, F	
	INCF	TEMPB3,F	
DBOK24			
2201121	COMF	BARGB2,F	; F1 = 2 - B0
	COMF	BARGB1,F	
	COMF	BARGBO,F	
	INCF	BARGB2,F	
	ADDWFC	BARGB1,F	
	ADDWFC	BARGB0,F	
	MOVPF	AARGB0,TEMPB0	i 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	
	MOTTED	MEMDD 2 NDEG	
	MOVFP	TEMPB2,WREG	
	MULWF MOVPF	BARGB1 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	AARGBO, WREG	
	MULWF	BARGB1	
	CLRF	AARGB1,W	
	ADDWFC	AARGB1,F	
	MOVPF	PRODL, WREG AARGB2, F	
	ADDWF MOVPF	PRODH, WREG	
	ADDWFC	AARGB1,F	
	·· ·	- ,	
	MOVFP	TEMPB2,WREG	
	MULWF	BARGB0	
	MOVPF	PRODL, WREG	
	ADDWF	AARGB3,F	
	MOVPF	PRODH, WREG	
	ADDWFC	AARGB2,F	

	CL DE	AADGDO M	
	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	
	ADDWIC	AARODO , F	
	BTFSC	AARGB0,MSB	; postnormalization
	GOTO	DNORM24	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
DNORM24	DECF	TEMPB3,F	
	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		, II ABAI > BBAI, COSC IOI OVCIIIOW
		_C	
	GOTO	SETFOV24	
	GOTO	DROUND24	
ALTB24	ADDWF	TEMPB3,W	
	ADDWF	EXP,F	; if AEXP < BEXP, test for underflow
	BTFSS	_C	,
	GOTO	SETFUN24	
	9010	SEIFONZŦ	
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		; using lsb in carry
		AARGB1,W	, abing ibb 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
SETFUN24
                BSF
                                 FPFLAGS, FUN
                                                          ; set floating point underflag
                BTFSS
                                 FPFLAGS, SAT
                                                          ; test for saturation
                RETLW
                                                          ; return error code in WREG
                                 0xFF
                MOVLW
                                 0x01
                                                          ; saturate to smallest floating
                                                          ; point number = 0x 01 00 00
                MOVPF
                                 WREG, AEXP
                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
                             0xC30C
                DATA
                DATA
                             0xBA2F
                DATA
                             0xB216
                DATA
                             0xAAAB
                DATA
                             0xA3D7
                             0×9D8A
                DATA
                DATA
                             0x97B4
                             0x9249
                DATA
                DATA
                             0x8D3E
                DATA
                             0x8889
                DATA
                             0 \times 8421
                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
                             0xF6D8
                DATA
                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 DATA	0xC5B2 0xC51A 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 0xBB84
DATA DATA	0xBB64 0xBAFB
DATA	0xBAFB
DATA	0xBA72 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 DATA	0xB0E3 0xB069
DATA	0xB069 0xAFF0
DATA	0xAFF7
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 0xA8B1
DATA DATA	0xA6B1
DATA	0xA042 0xA7D3
DATA	0xA7D3
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 DATA	0x9E1C 0x9DBA
DATA	0x9D5A
DATA	0x9CF9
DATA	0x9CF9
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 DATA	0x9574 0x951D
DATA	0x931D
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 DATA	0x8EA0 0x8E50
DATA	0x8E50
DATA	0x8DB3
DATA	0x8D65
DATA	0x8D17
DATA	0x8CC9

```
DATA
                        0x8C7C
             DATA
                        0x8C2F
             DATA
                        0x8BE2
             DATA
                        0x8B96
             DATA
                        0x8B4A
             DATA
                        Ox8AFE
                        0x8AB3
             DATA
             DATA
                        0x8A68
             DATA
                        0x8A1E
             DATA
                        0x89D3
             DATA
                        0x8989
                        0x893F
             DATA
             DATA
                        0x88F6
             DATA
                        0x88AD
             DATA
                        0 \times 8864
             DATA
                        0x881B
                        0x87D3
             DATA
             DATA
                        0x878B
                        0x8744
             DATA
             DATA
                        0x86FC
                        0x86B5
             DATA
             DATA
                        0x866F
             DATA
                        0x8628
                        0x85E2
             DATA
                        0x859C
             DATA
                        0x8557
             DATA
             DATA
                        0x8511
             DATA
                        0x84CC
             DATA
                        0x8487
             DATA
                        0x8443
             DATA
                        0x83FF
             DATA
                        0x83BB
             DATA
                        0x8377
             DATA
                        0 \times 8334
             DATA
                        0x82F1
             DATA
                        0x82AE
             DATA
                        0x826B
                        0x8229
             DATA
             DATA
                        0x81E7
             DATA
                        0x81A5
             DATA
                        0x8164
             DATA
                        0x8122
                        0x80E1
             DATA
                        0x80A1
             DATA
                        0x8060
             DATA
             DATA
                        0x8020
Floating Point Subtract
      Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
             24 bit floating point number in BEXP, BARGBO, BARGB1
             CALL FPS24
      Output: 24 bit floating point difference in AEXP, AARGBO, AARGB1
      Result: AARG <-- AARG - BARG
                                                RND = 0
      Max Timing:
                   1+133 = 134 \text{ clks}
                    1+146 = 147 \text{ clks}
                                                RND = 1, SAT = 0
                    1+152 = 153 \text{ clks}
                                                RND = 1, SAT = 1
      Min Timing:
                    1+10 = 11 \text{ clks}
```

Use:

```
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, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
      Use:
           CALL FPA24
;
       Output: 24 bit floating point sum in AEXP, AARGBO, AARGB1
      Result: AARG <-- AARG - BARG
;
                    81+52 = 133 \text{ clks}
;
      Max Timing:
                                              RND = 0
                     81+65 = 146 \text{ clks}
                                              RND = 1, SAT = 0
;
                    81+71 = 152 \text{ 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
                            WREG, TEMP
              MOVPF
              CLRF
                            AARGB2,F
                                             ; clear extended byte
              MOVFP
                            AEXP, WREG
                                              ; use AARG if AEXP >= BEXP
              CPFSGT
                            BEXP
              COTO
                            USEA24
USEB24
              MOVFP
                            BARGB0, WREG
                                              ; use BARG if AEXP < BEXP
              MOVPF
                            WREG, SIGN
                                              ; save sign in SIGN
                            BARGB0,MSB
                                              ; make MSB's explicit
              BSF
                            AARGB0,MSB
              BSF
              MOVFP
                            AEXP, WREG
                                              ; compute shift count in BEXP
              MOVPF
                            WREG, TEMPB1
              MOVED
                            BEXP, WREG
              MOVPF
                            WREG, AEXP
              CLRF
                            WREG, F
                                              ; return BARG if AARG = 0
              CPFSGT
                            TEMPB1
              GOTO
                            BRETURN24
              MOVFP
                            TEMPB1, WREG
              SUBWF
                            BEXP,F
              BTFSC
                            _Z
              GOTO
                            BLIGNED24
                            7
              MOVLW
              CPFSGT
                            BEXP
                                              ; do byte shift if BEXP >= 8
              GOTO
                            BNIB24
              SUBWF
                            BEXP.F
                                              ; BEXP = BEXP - 7
                                              ; keep for postnormalization
                            AARGB1,AARGB2
              MOVFP
                            AARGB0, AARGB1
              MOVFP
              CLRF
                            AARGB0,F
                            BEXP,F
                                              ; BEXP = BEXP -1
              DCFSNZ
```

	GOTO	BLIGNED24	
	CPFSGT GOTO	BEXP BNIB24A	; do byte shift if BEXP >= 8
	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 DEVD atill >= 0 then
	GOTO	BNIB24B	; if BEXP still >= 8, then ; AARG = 0 relative to BARG
	G010	DIVIDATO	AARG - 0 TETACIVE CO BARG
BRETURN24	MOVFP	SIGN,AARGB0	; return BARG
	MOVFP	BARGB1,AARGB1	
	CLRF	AARGB2,F	
	RETLW	0x 0 0	
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	3
	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 0x0F	
	ANDLW 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	· aligned if PEVD - 0
	GOTO	BLIGNED24	; aligned if BEXP = 0
	BCF RRCF	_C AARGB1,F	
	RRCF	AARGB1,F	
	DCFSNZ	BEXP,F	
	GOTO	BLIGNED24	; aligned if BEXP = 0
			-

	D.G.D.	a	
	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	AARGBO,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	
-1VII () 2. 1	MOVPF	AARGB0,SIGN	; save sign in SIGN
	BSF	AARGB0, SIGN	; make MSB's explicit
	BSF	BARGBO,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
	MOVEP	BARGBO, WREG	
	MOVPF CLRF	WREG,BARGB1 BARGB0,F	
	DCFSNZ	BEXP,F	; BEXP = BEXP - 1
	GOTO	ALIGNED24	, pent - pent I
	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	
	MOME III	7	
	MOVLW CPFSGT	BEXP	; if BEXP still >= 8, then
	GOTO	ANIB24B	; BARG = 0 relative to AARG
	0010	ANIDZ ID	/ DANG - 0 ICIACIVE CO AANG
	MOVFP	SIGN, AARGBO	; 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
ALOOPZ4B	RRCF	_C BARGB2,F	/ light shift by bear
	DCFSNZ	BEXP,F	
	GOTO	ALIGNED24	; aligned if BEXP = 0
	BCF	_C	3
	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	
7NTD045	MOLITE	2	. de withten is is a series
ANIB24A	MOVLW	3 DEVD	; do nibbleshift if BEXP >= 4
	CPFSGT	BEXP	
	GOTO	ALOOP24A	· nevn - nevn 2
	SUBWF SWAPF	BEXP,F BARGB2,W	; BEXP = BEXP -3
	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 SWAPF	BARGB2,F BARGB1,W	
	ANDLW	0x0F	
	MOVPF	WREG, BARGB1	
	SWAPF	BARGBO, W	
	ANDLW	0xF0	
	ADDWF	BARGB1,F	
	SWAPF	BARGBO,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	BARGBO, F	; possible
	RRCF RRCF	BARGB1,F BARGB2,F	
	MOF	Dinioba / I	
ALIGNED24	CLRF	AARGB2,W	
	BTFSS	TEMP, MSB	; negate if signs opposite
	GOTO	AOK24	
	COMF	BARGB2,F	
	COMF	BARGB1,F	

	COMF INCF ADDWFC ADDWFC	BARGB0,F BARGB2,F BARGB1,F BARGB0,F	
AOK24	MOVFP ADDWF MOVFP ADDWFC MOVFP ADDWFC	BARGB2, WREG AARGB2, F BARGB1, WREG AARGB1, F BARGB0, WREG AARGB0, F	; add
	BTFSC GOTO BTFSS GOTO	TEMP,MSB ACOMP24 _C NRMRND3224	
	RRCF RRCF RRCF INCFSZ GOTO	AARGB0, F AARGB1, F AARGB2, F AEXP, F NRMRND3224	; shift right and increment EXP
ACOMP24	GOTO BTFSC GOTO	SETFOV24 _C NRM3224	<pre>; set floating point overflow flag ; normalize and fix sign</pre>
	COMF COMF COMF INCF CLRF ADDWFC ADDWFC BTG GOTO	AARGB2, F AARGB1, F AARGB0, F AARGB2, F WREG, F AARGB1, F AARGB0, F SIGN, MSB NRM3224	; negate, toggle sign bit and ; then normalize

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
FL02432
                24 bit integer to 32 bit floating point conversion
FLO32
        Timing:
                           RND
                        Λ
                                 1
                        104
                                 104
           SAT
                        110
                                 110
NRM3232
          32 bit normalization of unnormalized 32 bit floating point numbers
NRM32
        Timing:
                           RND
                        O
                                 1
                                 90
           SAT
                                 96
                        96
INT3224
                32 bit floating point to 24 bit integer conversion
INT32
        Timing:
                           RND
                        0
                                 1
                        104
                                  112
           SAT
                        104
                                  114
FLO3232 32 bit integer to 32 bit floating point conversion
        Timing:
                           RND
                0
                        129
                                 145
           SAT
                        129
                                 152
```

```
NRM4032 32 bit normalization of unnormalized 40 bit floating point numbers
        Timing:
                           RND
                        0
                                 1
                        112
                                 128
           SAT
                        112
                                 135
INT3232
                32 bit floating point to 32 bit integer conversion
        Timing:
                            RND
                        0
                                 1
                        130
           SAT
                        130
                                 137
FPA32
                32 bit floating point add
        Timing:
                            RND
                        251
                                 265
           SAT
                1
                        251
                                 271
FPS32
                32 bit floating point subtract
        Timing:
                        253
                                 267
           SAT
                1
                        253
                                 273
FPM32
                32 bit floating point multiply
        Timing:
                            RND
                        574
                                 588
           SAT
                        574
                                 591
FPD32
                32 bit floating point divide
        Timing:
                            RND
                                 1
                        932
                                 968
           SAT
                        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.
                      8 bit most significant byte of fraction in sign-magnitude representation,
       HIGHBYTE
                      with SIGN = MSB, implicit MSB = 1 and radix point to the right of MSB
                      8 bit middle significant byte of sign-magnitude fraction
       MIDBYTE
       LOWBYTE
                      8 bit least significant byte of sign-magnitude fraction
       EXPONENT
                     HIGHBYTE
                                    MIDBYTE
                                                  LOWBYTE
       XXXXXXX
                      S.xxxxxxx
                                   xxxxxxxx
                                                  xxxxxxx
                     RADIX
                     POINT
Integer to float conversion
       Input: 24 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2
       Use: CALL
                    FL02432 or
                                   CALL
                                          FLO32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- FLOAT( AARG )
                    14+90 = 104 \text{ clks}
                                                   SAT = 0
       Max Timing:
                     14+96 = 110 \text{ clks}
                                                   SAT = 1
       Min Timing: 6+28 = 34 clks
                     6+18 = 24 \text{ clks}
       PM: 14+38 = 52
                                                   DM: 7
FLO2432
FLO32
                            D'23'+EXPBIAS
                                                  ; initialize exponent and add bias
              MOVLW
              MOVWF
                            EXP
              CLRF
                            SIGN
              BTFSS
                            AARGB0,MSB
                                                  ; test sign
                             NRM3232
              GOTO
              COMF
                             AARGB2,F
                                                   ; if < 0, negate and set MSB in SIGN
              COMF
                             AARGB1,F
              COMF
                             AARGB0,F
              INCF
                             AARGB2,F
              BTFSC
                             _{\rm Z}
              INCF
                             AARGB1,F
              BTFSC
                             _{\rm Z}
              INCF
                             AARGB0,F
                             SIGN, MSB
              BSF
```

```
******************************
        Normalization routine
        Input: 32 bit unnormalized floating point number in AEXP, AARGBO, AARGB1,
                AARGB2, with sign in SIGN, MSB
        Use:
                CALL
                        NRM3232 or
                                        CALL
                                                NRM32
        Output: 32 bit normalized floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- NORMALIZE( AARG )</pre>
        Max Timing:
                       21+6+7*8+7 = 90 \text{ clks}
                                                        SAT = 0
                        21+6+7*8+1+12 = 96 \text{ clks}
                                                        SAT = 1
        Min Timing:
                        22+6 = 28 \text{ clks}
                                                        AARG = 0
                        5+9+4 = 18 \text{ clks}
       PM: 38
                                                        DM: 7
NRM3232
                CLRF
                                TEMP
                                                         ; clear exponent decrement
NRM32
                MOVF
                                AARGB0,W
                                                        ; test if highbyte=0
                BTFSS
                                _{\rm Z}
                                NORM3232
                GOTO
                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
                                _{\rm Z}
                COTO
                                NORM3232
                                AARGB1,W
                                                        ; if so, shift 8 bits by move
                MOVF
                MOVWF
                                AARGB0
                CLRF
                                AARGB1
                                TEMP, 3
                BCF
                                                         ; increase decrement by 8
                                TEMP,4
                BSF
                MOVF
                                AARGB0,W
                                                        ; if highbyte=0, result=0
                BTFSC
                                _{\rm Z}
                                RES032
                GOTO
NORM3232
                MOVF
                                TEMP,W
                SUBWF
                                EXP,F
                BTFSS
                                _{\rm Z}
                BTFSS
                                _C
                                SETFUN32
                GOTO
                BCF
                                _C
                                                        ; clear carry bit
NORM3232A
                BTFSC
                                AARGB0,MSB
                                                        ; if MSB=1, normalization done
                GOTO
                                FIXSIGN32
                RLF
                                AARGB2,F
                                                        ; otherwise, shift left and
                                                        ; decrement EXP
                RLF
                                AARGB1,F
                RLF
                                AARGB0,F
                DECFSZ
                                EXP,F
                GOTO
                                NORM3232A
                                                        ; underflow if EXP=0
                GOTO
                                SETFUN32
```

```
FIXSIGN32
             BTFSS
                           SIGN, MSB
             BCF
                           AARGB0,MSB
                                               ; clear explicit MSB if positive
             RETLW
RES032
             CLRF
                          AARGB0
                                               ; result equals zero
                          AARGB1
             CLRF
             CLRF
                          AARGB2
             CLRF
                           AARGB3
             CLRF
                           EXP
             RETLW
Integer to float conversion
      Input: 32 bit 2's complement integer right justified in AARGB0, AARGB1, AARGB2,
             AARGB3
;
                   FL03232
;
      Use:
            CALL
;
      Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- FLOAT( AARG )
;
      Max Timing:
                   17+112 = 129 \text{ clks}
                                               RND = 0
                    17+128 = 145 \text{ clks}
                                               RND = 1, SAT = 0
;
                                               RND = 1, SAT = 1
;
                    17+135 = 152 \text{ clks}
      Min Timing:
                   6+39 = 45 \text{ clks}
                                               AARG = 0
                    6+22 = 28 \text{ clks}
      PM: 17+66 = 83
                                                DM: 8
             MOVLW
                          D'31'+EXPBIAS
FL03232
                                             ; initialize exponent and add bias
             MOVWF
                           EXP
             CLRF
                           SIGN
             BTFSS
                           AARGB0,MSB
                                               ; test sign
             GOTO
                           NRM4032
                                               ; if < 0, negate and set MSB in SIGN
             COMF
                           AARGB3,F
             COMF
                           AARGB2,F
                           AARGB1,F
             COMF
             COMF
                           AARGB0,F
             INCF
                          AARGB3,F
             BTFSC
                           _{\rm Z}
             INCF
                           AARGB2,F
                           _{\rm Z}
             BTFSC
             INCF
                           AARGB1,F
             BTFSC
                           _{\rm Z}
                           AARGB0,F
             INCF
             BSF
                           SIGN, MSB
```

```
*****************************
       Normalization routine
       Input: 40 bit unnormalized floating point number in AEXP, AARGBO, AARGB1,
              AARGB2, AARGB3 with sign in SIGN, MSB
       Use:
              CALL
                      NRM4032
       Output: 32 bit normalized floating point number in AEXP, AARGBO, AARGB1, AARGB2,
       Result: AARG <-- NORMALIZE( AARG )
                                                   RND = 0
                     38+6*9+12+8 = 112 \text{ clks}
       Max Timing:
                      38+6*9+12+24 = 128 \text{ clks}
                                                    RND = 1, SAT = 0
                      38+6*9+12+31 = 135 \text{ clks}
                                                    RND = 1, SAT = 1
                      33+6 = 39 \text{ clks}
       Min Timing:
                                                    AARG = 0
;
                      5+9+8 = 22 \text{ clks}
       PM: 66
                                                    DM: 8
TEMP
NRM4032
              CLRF
                                                    ; clear exponent decrement
              MOVF
                             AARGB0,W
                                                   ; test if highbyte=0
              BTFSS
                             _{\rm 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
                             AARGB0,W
                                                    ; test if highbyte=0
              MOVF
               BTFSS
                              Z
               GOTO
                              NORM4032
                                                    ; if so, shift 8 bits by move
               MOVF
                             AARGB1,W
                             AARGB0
              MOVWF
                             AARGB2,W
              MOVF
              MOVWF
                             AARGB1
               CLRF
                              AARGB2
              BCF
                             TEMP,3
                                                    ; increase decrement by 8
               BSF
                             TEMP,4
              MOVF
                              AARGB0,W
                                                    ; test if highbyte=0
               BTFSS
                              _{\rm Z}
                             NORM4032
               GOTO
              MOVF
                             AARGB1,W
                                                    ; if so, shift 8 bits by move
              MOVWF
                             AARGB0
               CLRF
                              AARGB1
               BSF
                             TEMP,3
                                                    ; increase decrement by 8
                              AARGB0,W
                                                    ; if highbyte=0, result=0
              MOVF
               BTFSC
                              Z
               GOTO
                              RES032
NORM4032
              MOVF
                              TEMP,W
               SUBWF
                              EXP,F
               BTFSS
                              _{\rm Z}
               BTFSS
                              _C
               GOTO
                              SETFUN32
```

```
BCF
                           _C
                                               ; clear carry bit
NORM4032A
                                               ; if MSB=1, normalization done
             BTFSC
                           AARGB0,MSB
             GOTO
                           NRMRND4032
             RLF
                           AARGB3.F
                                               ; otherwise, shift left and
                                               ; decrement EXP
             RLF
                           AARGB2,F
             RLF
                          AARGB1,F
             RLF
                           AARGB0,F
             DECFSZ
                           EXP,F
             GOTO
                           NORM4032A
             GOTO
                           SETFUN32
                                               ; underflow if EXP=0
NRMRND4032
             BTFSC
                           FPFLAGS, RND
             BTFSS
                           AARGB2,LSB
                           FIXSIGN32
             GOTO
             BTFSS
                          AARGB3,MSB
                                               ; round if next bit is set
                          FIXSIGN32
             GOTO
             INCF
                          AARGB2,F
             BTFSC
                           _{\rm Z}
             INCF
                           AARGB1,F
             BTFSC
                           AARGB0,F
             INCF
             BTFSS
                           _{\rm Z}
                                               ; has rounding caused carryout?
             GOTO
                          FIXSIGN32
                          AARGB0,F
                                               ; if so, right shift
             RRF
             RRF
                          AARGB1,F
             RRF
                          AARGB2,F
             INCF
                           EXP,F
             BTFSC
                           _{\rm Z}
                                               ; check for overflow
                           SETFOV32
             GOTO
                           FIXSIGN32
             COTO
Float to integer conversion
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Use:
            CALL
                  INT3224
                                 or
                                       CALL
                                             INT32
      Output: 24 bit 2's complement integer right justified in AARGBO, AARGBI, AARGBO
;
      Result: AARG <-- INT( AARG )
;
      Max Timing:
                    40+6*7+6+16 = 104 \text{ clks}
                                               RND = 0
                    40+6*7+6+24 = 112 \text{ clks}
                                               RND = 1, SAT = 0
                    40+6*7+6+26 = 114 \text{ clks}
                                               RND = 1, SAT = 1
      Min Timing:
                   4 clks
      PM: 82
                                               DM: 6
;------
INT3224
INT32
                           EXP,W
             MOVF
                                               ; test for zero argument
             BTFSC
                           _{\rm Z}
             RETLW
                           0x00
             MOVF
                           AARGB0,W
                                               ; save sign in SIGN
             MOVWF
                           SIGN
```

	BSF	AARGB0,MSB	; make MSB explicit
	MOM IN	EADDING D 1 33 1	· remove bing from EVD
	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	, as ansener byte built in EAF >- 0
	BTFSS	_C	
	GOTO	_C TSHIFT3224	
	MOVWF		
		EXP	· mant bit fan manudina
	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	- SHIFT32240K	
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	INT32240K	
	BTFSS	_C	
	GOTO	INT32240K	
	INCF	AARGB2,F	
	BTFSC	_Z	
	INCF	AARGB1,F	
	BTFSC	_Z	
	INCF	AARGB0,F	
	BTFSC	AARGB0,MSB	; test for overflow
	GOTO	SETIOV3224	

```
INT32240K
              BTFSS
                            SIGN, MSB
                                                  ; if sign bit set, negate
              RETLW
                            AARGB0,F
              COMF
              COMF
                            AARGB1,F
              COME
                            AARGB2.F
                            AARGB2,F
              INCF
              BTFSC
                            _{\rm Z}
              INCF
                            AARGB1,F
              BTFSC
                            _{\rm Z}
              INCF
                            AARGB0,F
              RETLW
IRES03224
              CLRF
                            AARGB0
                                                  ; integer result equals zero
              CLRF
                            AARGB1
              CLRF
                            AARGB2
              RETLW
SETTOV3224
              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
                                                  ; complement 24 bit integer
              BTFSS
                            SIGN, MSB
              MOVLW
                            0xFF
                                                  ; SIGN = 0, 0x 7F FF FF
              MOVWF
                            AARGB0
              MOVWF
                            AARGB1
                                                  ; SIGN = 1, 0x 80 00 00
              MOVWF
                            AARGB2
              RLF
                            SIGN, F
              RRF
                            AARGB0,F
              RETLW
                                                  ; return error code in WREG
Float to integer conversion
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
              CALL
                     INT3232
       Use:
       Output: 32 bit 2's complement integer right justified in AARGBO, AARGB1, AARGB2,
              AARGB3
       Result: AARG <-- INT( AARG )
;
       Max Timing:
                     54+6*8+7+21 = 130 \text{ clks}
                                                 RND = 0
                     54+6*8+7+29 = 137 \text{ clks}
                                                  RND = 1, SAT = 0
;
                     54+6*8+7+29 = 137 \text{ clks}
                                                  RND = 1, SAT = 1
       Min Timing:
                  5 clks
       PM: 102
                                                  DM: 7
TNT3232
              CLRF
                            AARGB3
                            EXP,W
              MOVF
                                                 ; test for zero argument
              BTFSC
                            _{\rm Z}
                            0x00
              RETLW
                            AARGB0,W
              MOVF
                                                  ; 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 MOVF	AARGB2 AARGB0,W	
	MOVF	AARGBU,W AARGB1	
	CLRF	AARGB1 AARGB0	
	CLRF	AARGBU	
	MOVLW	8	<pre>; do another byte shift if EXP >= 8</pre>
	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	INT32320K	
	BTFSS	_C	
	GOTO	INT32320K	
	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	
	DEED G	a. a	. 16 1 11 11
INT32320K	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	
a=== a== a	202		
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
	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, AARGBO, AARGB1, AARGB2
              32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
       Use:
              CALL
                   FPM32
       Output: 32 bit floating point product in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- AARG * BARG
                     26+23*22+21+21 = 574 \text{ clks}
                                                 RND = 0
       Max Timing:
                     26+23*22+21+35 = 588 clks
                                                 RND = 1, SAT = 0
                     26+23*22+21+38 = 591 \text{ clks}
                                                 RND = 1, SAT = 1
                                                  AARG * BARG = 0
      Min Timing:
                     6+6 = 12 \text{ clks}
                     24+23*11+21+17 = 315 \text{ clks}
      PM: 94
                                                  DM: 14
FPM32
              MOVF
                           AEXP,W
                                                 ; test for zero arguments
              BTFSS
                            _{\rm Z}
              MOVF
                            BEXP,W
              BTFSC
                            _{\rm Z}
              GOTO
                            RES032
M32BNE0
              MOVF
                            AARGB0,W
              XORWF
                            BARGB0,W
                                                  ; save sign in SIGN
              MOVWF
                            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
                            _C
              BTFSS
              GOTO
                            SETFUN32
MOK32
              MOVF
                            AARGB0,W
              MOVWF
                            AARGB3
              MOVF
                            AARGB1,W
              MOVWF
                            AARGB4
              MOVF
                            AARGB2,W
              MOVWF
                            AARGB5
                            AARGB3,MSB
                                                  ; make argument MSB's explicit
              BSF
              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	
	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
			; test for saturation
	BTFSS	FPFLAGS, SAT	
	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
			, modulo one appropriace sign bit
	MOVWF	AARGB2	

```
SIGN, F
                RLF
                RRF
                                 AARGB0,F
                RETLW
                                 0xFF
                                                          ; return error code in WREG
        Floating Point Divide
        Input: 32 bit floating point dividend in AEXP, AARGBO, AARGB1, AARGB2
                32 bit floating point divisor in BEXP, BARGBO, BARGB1, BARGB2
        Use:
                CALL
                        FPD32
        Output: 32 bit floating point quotient in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- AARG / BARG
                        43+12+23*36+35+14 = 932 \text{ clks} RND = 0
        Max Timing:
                        43+12+23*36+35+50 = 968 \text{ clks} RND = 1, SAT = 0
                        43+12+23*36+35+53 = 971 clks
                                                        RND = 1, SAT = 1
        Min Timing:
                        7+6 = 13 \text{ clks}
        PM: 155
                                                          DM: 14
FPD32
                MOVF
                                BEXP.W
                                                         ; test for divide by zero
                                 _Z
                BTFSC
                GOTO
                                 SETFDZ32
                                 AEXP,W
                MOVF
                BTFSC
                                 _{\rm Z}
                                 RES032
                GOTO
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
                                                          ; test for alignment
                                 AARGB3
                MOVF
                                 AARGB1,W
                MOVWF
                                 AARGB4
                                 AARGB2,W
                MOVF
                MOVWF
                                 AARGB5
                MOVF
                                 BARGB2,W
                SUBWF
                                 AARGB5,F
                                 BARGB1,W
                MOVF
                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
                                 AARGB5
                CLRF
```

	BTFSS	_C	
	GOTO	DALIGN32OK	
	D.C.D.		
	BCF	_C	; align if necessary
	RRF	AARGB0,F	
	RRF	AARGB1,F	
	RRF	AARGB2,F	
	RRF	AARGB3,F	
	MOVINE	0x01	
	MOVWF	TEMP	; save align increment
DALIGN32OK	MOVF	BEXP,W	; compare AEXP and BEXP
DIMIGNOZOR	SUBWF	EXP,F	, compare him and bim
	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	
	MOLTE	DADGDO M	; subtract
	MOVF	BARGB2,W	, subtract
	SUBWF	AARGB2,F	
	MOVF BTFSS	BARGB1,W _C	
	INCFSZ	_	
DS132	SUBWF	BARGB1,W AARGB1,F	
DSISZ	SODWI	AARGBI, 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	, for rounding
	RLF	TEMP, F	
	KLIF	IEMP, 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	7.	; test if rounding caused carryout
	GOTO	_Z DIV320K	, cost if founding caused carryout
	RRF	AARGB3,F	
	RRF	AARGB4,F	
	RRF	AARGB5,F	
	INCF	EXP,F	
	BTFSC	_Z	; test for overflow
	GOTO	SETFOV32	/ test for overflow
	G010	SEIFOVSZ	
DIM2200	DTTCC	CICN MCD	
DIV32OK	BTFSS	SIGN, MSB	· along andigit MCD if monition
	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
                                                ; set floating point underflag
             BSF
                           FPFLAGS.FUN
             BTFSS
                           FPFLAGS, SAT
                                                ; test for saturation
             RETLW
                                                ; return error code in WREG
                           0xFF
             MOVLW
                           0 \times 01
                                                ; saturate to smallest floating
                                                ; point number = 0 \times 01 \ 00 \ 00 \ 00
             MOVWF
                           AEXP
             CLRF
                           AARGB0
                                                ; modulo the appropriate sign bit
             CLRF
                          AARGB1
             CLRF
                          AARGB2
             RLF
                           SIGN, F
             RRF
                           AARGB0,F
                                                ; return error code in WREG
             RETLW
                           0xFF
SETFDZ32
             BSF
                           FPFLAGS, FDZ
                                               ; set divide by zero flag
             RETLW
                           0xFF
Floating Point Subtract
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
             32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
             CALL FPS32
;
      Use:
      Output: 32 bit floating point sum in AEXP, AARGBO, AARGB1, AARGB2
;
      Result: AARG <-- AARG - BARG
      Max Timing:
                    2+251 = 253 \text{ clks}
;
                    2+265 = 267 \text{ clks}
                                                RND = 1, SAT = 0
                    2+271 = 273 \text{ clks}
                                                RND = 1, SAT = 1
      Min Timing:
                   2+12 = 14 \text{ clks}
      PM: 2+146 = 148
                                                DM: 14
;-----
FPS32
             MOVLW
             XORWF
                           BARGB0,F
      Floating Point Add
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
;
             32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
      Use: CALL FPA32
      Output: 32 bit floating point sum in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- AARG - BARG
                   31+41+6*7+6+41+90 = 251 \text{ clks} RND = 0
;
      Max Timing:
                    31+41+6*7+6+55+90 = 265 \text{ clks} RND = 1, SAT = 0
                    31+41+6*7+6+55+96 = 271 \text{ clks}
;
                                                RND = 1, SAT = 1
      Min Timing:
                   8+4 = 12 \text{ 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	
		D1D6D0	
	MOVF	BARGBO,W	
	MOVWF	AARGB5	
	MOVF	AARGBO,W	
	MOVWF	BARGB0	
	MOVF	AARGB5,W	
	MOVWF	AARGB0	
	MOTTE	DIDGD1 H	
	MOVF	BARGB1,W	
	MOVWF	AARGB5	
	MOVF	AARGB1,W	
	MOVWF	BARGB1	
	MOVE	AARGB5,W	
	MOVWF	AARGB1	
	MOVF	BARGB2,W	
	MOVWF	AARGB5	
	MOVF	AARGB2,W	
	MOVWF	BARGB2	
	MOVF	AARGB5,W	
	MOVWF	AARGB2	
	MOVWI	AAKGDZ	
USEA32	MOVF	BEXP,W	; return AARG if BARG = 0
	BTFSC	_Z	
	RETLW		
	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	
	MONT I	8	
	MOVLW		
	SUBWF	BEXP,W	· if DEVD >= 0 do hhift
	BTFSS	_C	; if BEXP >= 8, do byte shift
	GOTO	ALIGNB32	
	MOVWF	BEXP	. 1
	MOVF	BARGB2,W	; keep for postnormalization
	MOVWF	BARGB3	
	MOVF	BARGB1,W	
	MOVWF	BARGB2	
	MOVF	BARGBO,W	
	MOVWF	BARGB1	

	CLRF	BARGB0	
	MOVIT II	8	
	MOVLW SUBWF	o BEXP,W	
	BTFSS	_C	; if BEXP >= 8, do byte shift
	GOTO	ALIGNB32	/ II bear >= 0, do byte shirt
	MOVWF	BEXP	
	MOVF	BARGB2,W	; keep for postnormalization
	MOVWF	BARGB3	, heep for postnormalization
	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	$0 \times 0 0$	
ALIGNB32	MOVF	BEXP,W	<pre>; already aligned if BEXP = 0</pre>
	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
1111011111111	GOTO	AOK32	, negace ii bignb eppoblee
	3010	1101132	
	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	
	ADDWF MOVF	AARGB2,F BARGB1,W	
	ADDWF MOVF BTFSC	AARGB2,F BARGB1,W _C	
	ADDWF MOVF BTFSC INCFSZ	AARGB2,F BARGB1,W _C BARGB1,W	
	ADDWF MOVF BTFSC INCFSZ ADDWF	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF BTFSC	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W _C	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF BTFSC INCFSZ	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W _C BARGB0,W	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF BTFSC	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W _C	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF BTFSC INCFSZ	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W _C BARGB0,W	
	ADDWF MOVF BTFSC INCFSZ ADDWF MOVF BTFSC INCFSZ ADDWF	AARGB2,F BARGB1,W _C BARGB1,W AARGB1,F BARGB0,W _C BARGB0,W AARGB0,F	

	BTFSS GOTO	_C NRMRND4032	
	RRF RRF RRF INCFSZ GOTO GOTO	AARGB0,F AARGB1,F AARGB2,F AARGB3,F AEXP,F NRMRND4032 SETFOV32	; shift right and increment EXP
ACOMP32	BTFSC GOTO	_C NRM4032	; normalize and fix sign
	COMF COMF COMF COMF INCF BTFSC INCF BTFSC INCF	AARGB3,F AARGB1,F AARGB0,F AARGB3,F _Z AARGB2,F _Z AARGB1,F _Z AARGB1,F _Z AARGB1,F	; negate, toggle sign bit and ; then normalize
	MOVLW XORWF GOTO	0x80 SIGN,F 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
                                1
                        60
                                 60
           SAT
                        67
                                 67
NRM3232
          32 bit normalization of unnormalized 32 bit floating point numbers
NRM32
        Timing:
                           RND
                0
                        48
                                 48
           SAT
                        55
                                 55
                1
                32 bit floating point to 24 bit integer conversion
INT3224
INT32
        Timing:
                           RND
                        Ω
                                 1
                        70
                                 79
           SAT
                        70
                                 81
FL03232
        32 bit integer to 32 bit floating point conversion
        Timing:
                           RND
                        Ω
                                 1
                0
                        75
                                91
           SAT
                        75
                                 97
```

```
NRM4032 32 bit normalization of unnormalized 40 bit floating point numbers
       Timing:
                        RND
                      0
                             77
                      61
          SAT
                      61
                             83
INT3232
        32 bit floating point to 32 bit integer conversion
       Timing:
                        RND
                      0
                             1
          SAT
                      82
                             93
              32 bit floating point add
FPA32
       Timing:
                        RND
                             1
                      160
                             176
          SAT
                      160
                             182
FPS32
              32 bit floating point subtract
       Timing:
                             1
                      161
                             177
          SAT
                      161
                             183
             32 bit floating point multiply
FPM32
                     RND
       Timing:
                      100
                             114
          SAT
                      100
                             122
        32 bit floating point divide
FPD32
       Timing:
                       RND
                      323
                             337
          SAT
                      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
      xxxxxxx
                   S.xxxxxxx
                                xxxxxxx
                                             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, AARGBO, AARGB1, AARGB2
      Result: AARG <-- FLOAT( AARG )
                  12+48 = 60 \text{ clks}
                                              SAT = 0
      Max Timing:
                   12+55 = 67 \text{ clks}
                                              SAT = 1
                  6+24 = 30 \text{ clks}
      Min Timing:
                                              AARG = 0
                   6+20 = 26 \text{ 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
                          AARGB0,F
             COMF
                          AARGB2,F
             INCF
             ADDWFC
                          AARGB1,F
             ADDWFC
                          AARGB0,F
```

```
******************************
        Normalization routine
        Input: 32 bit unnormalized floating point number in AEXP, AARGBO, AARGB1,
                AARGB2 with sign in SIGN, MSB.
        Use:
                CALL
                       NRM3232 or
                                       CALL
                                                NRM32
        Output: 32 bit normalized floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- NORMALIZE( AARG )</pre>
        Max Timing:
                        21+19+8 = 48 \text{ clks}
                                                        SAT = 0
                        21+19+15 = 55 \text{ clks}
                                                        SAT = 1
        Min Timing:
                        4+7+7+6 = 24 \text{ clks}
                                                        AARG = 0
                        3+5+4+8 = 20 \text{ clks}
       PM: 115
                                                        DM: 7
NRM3232
NRM32
                                TEMP,W
                CLRF
                                                        ; clear exponent decrement
                CPFSGT
                                AARGB0
                                                        ; test if highbyte=0
                                NRM3232A
                GOTO
TNIB3232
                MOVLW
                                0xF0
                                                        ; test if highnibble=0
                ANDWF
                                AARGB0,W
                TSTFSZ
                                WREG
                GOTO
                               NORM3232
                                                       ; if so, shift 4 bits
                SWAPF
                               AARGB0,F
                SWAPF
                                AARGB1,W
                ANDLW
                                0x0F
                ADDWF
                                AARGB0,F
                SWAPF
                                AARGB1,W
                ANDLW
                                0xF0
                MOVPF
                                WREG, AARGB1
                SWAPF
                                AARGB2,W
                ANDI-W
                                0x0F
                ADDWF
                                AARGB1,F
                SWAPF
                                AARGB2,W
                ANDLW
                                0xF0
                MOVPF
                                WREG, AARGB2
                                TEMP, 2
                                                        ; increase decrement by 4
NORM3232
                BCF
                                                        ; clear carry bit
                                _C
                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
                                AARGB1,F
                RLCF
                                AARGB0,F
                RLCF
                INCF
                                TEMP,F
                                                        ; since highnibble != 0, at most
                BTFSC
                                AARGB0,MSB
```

	GOTO	TNORMUN3232	; 3 left shifts are required
	RLCF	AARGB2,F	-
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	INCF	TEMP,F	
	11101	12.11 /1	
TNORMUN3232	MOVFP	TEMP, WREG	; if EXP <= decrement in TEMP,
	CPFSGT	EXP	; floating point underflow has
	GOTO	SETFUN32	; occured
	SUBWF	EXP,F	; otherwise, compute EXP
	DODNI	HAL /1	, defici wise, compace in
FIXSIGN32	BTFSS	SIGN,MSB	
	BCF	AARGB0,MSB	; clear explicit MSB if positive
	RETLW	0	1
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	3 47 11
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
	BSF		; increase decrement by 4
NORM3232A	BSF BCF		<pre>; increase decrement by 4 ; clear carry bit</pre>
NORM3232A		TEMP,2	; clear carry bit
NORM3232A		TEMP,2	_
NORM3232A	BCF	TEMP,2	<pre>; clear carry bit ; if MSB=1, normalization done</pre>
NORM3232A	BCF BTFSC GOTO RLCF	TEMP,2 _C AARGB0,MSB	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO	TEMP, 2 _C AARGBO, MSB TNORMUN3232	<pre>; clear carry bit ; if MSB=1, normalization done</pre>
NORM3232A	BCF BTFSC GOTO RLCF	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB1, F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF	TEMP, 2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most</pre>
NORM3232A	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF RLCF	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most</pre>
	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF INCF BTFSC GOTO RLCF RLCF INCF	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TEMP, F TEMP, F TNORMUN3232	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required</pre>
NORM3232A NRM3232B	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF INCF BTFSC GOTO RLCF RLCF INCF MOVFP	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TEMP, F TNORMUN3232 AARGB1, AARGB0	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most</pre>
	BCF BTFSC GOTO RLCF RLCF INCF RLCF INCF RLCF INCF RLCF INCF INCF GOTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TORMUN3232 AARGB1, F AARGB0, F TEMP, F TNORMUN3232 AARGB1, AARGB0 AARGB1, W	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move</pre>
	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF INCF BTFSC GOTO RLCF RLCF INCF MOVFP	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TEMP, F TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TNORMUN3232 AARGB1, AARGB0 AARGB1, W TEMP, 3	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required</pre>
	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TEMP, F TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TNORMUN3232 AARGB1, AARGB0 AARGB1, W TEMP, 3 TEMP, 4	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8</pre>
	BCF BTFSC GOTO RLCF RLCF INCF BCF RLCF INCF COTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TORMUN3232 AARGB1, F AARGB0, F TEMP, F TNORMUN3232 AARGB1, F TEMP, F TNORMUN3232 AARGB1, AARGB0 AARGB1, W TEMP, 3 TEMP, 4 AARGB0	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move</pre>
	BCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO RLCF RLCF RLCF INCF BTFSC GOTO RLCF RLCF INCF BTFSC GOTO	TEMP, 2 _C AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F AARGB0, F TEMP, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, MSB TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TEMP, F TNORMUN3232 AARGB1, F AARGB0, F TEMP, F TNORMUN3232 AARGB1, AARGB0 AARGB1, W TEMP, 3 TEMP, 4	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8</pre>
NRM3232B	BCF BTFSC GOTO RLCF RLCF INCF GOTO	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TAMP,F TORMUN3232 AARGB1,F AARGB0,F TEMP,F TEMP,F TNORMUN3232 AARGB1,W TEMP,3 TEMP,4 AARGB0 RES032	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8 ; if highbyte=0, result=0</pre>
	BCF BTFSC GOTO RLCF RLCF INCF GOTO MOVFP CLRF BCF BSF CPFSGT GOTO MOVLW	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TAMRGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TEMP,F TNORMUN3232 AARGB1,AARGB0 AARGB1,W TEMP,3 TEMP,4 AARGB0 RES032	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8</pre>
NRM3232B	BCF BTFSC GOTO RLCF RLCF INCF GOTO MOVFP CLRF BCF BSF CPFSGT GOTO MOVLW ANDWF	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TNORMUN3232 AARGB1,W TEMP,F TNORMUN3232 AARGB1,AARGB0 AARGB1,W TEMP,3 TEMP,4 AARGB0 RES032 0xF0 AARGB0,W	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8 ; if highbyte=0, result=0</pre>
NRM3232B	BCF BTFSC GOTO RLCF RLCF INCF GOTO MOVFP CLRF BCF BSF CPFSGT GOTO MOVLW	TEMP,2 _C AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F AARGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TAMRGB0,MSB TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TNORMUN3232 AARGB1,F AARGB0,F TEMP,F TEMP,F TNORMUN3232 AARGB1,AARGB0 AARGB1,W TEMP,3 TEMP,4 AARGB0 RES032	<pre>; clear carry bit ; if MSB=1, normalization done ; otherwise, shift left and ; increment decrement ; since highnibble != 0, at most ; 3 left shifts are required ; shift 8 bits by move ; increase decrement by 8 ; if highbyte=0, result=0</pre>

```
GOTO
                          NORM3232B
             SWAPF
                                              ; if so, shift 4 bits
                          AARGB0,F
                          TEMP,2
             BSF
                                              ; increase decrement by 4
NORM3232B
             BCF
                                              ; clear carry bit
                           _C
             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
                          TEMP,F
             INCF
             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
Integer to float conversion
      Input: 32 bit 2's complement integer right justified in AARGB0, AARGB1,
             AARGB2, AARGB3
      Use:
            CALL
                  FL03232
      Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- FLOAT( AARG )
      Max Timing:
                   14+61 = 75 \text{ clks}
                                              RND = 0
                   14+77 = 91 \text{ clks}
                                              RND = 1, SAT = 0
                   14+83 = 97 \text{ clks}
                                              RND = 1, SAT = 1
      Min Timing: 6+32 = 38 clks
                                              AARG = 0
                   6+26 = 32 \text{ clks}
      PM: 14+178 = 192
                                              DM: 8
;------
FL03232
             MOVLW
                          D'31'+EXPBIAS
                                             ; initialize exponent and add bias
             MOVWF
                          EXP
             MOVPF
                          AARGB0,SIGN
                                             ; save sign in SIGN
                          AARGB0,MSB
             BTFSS
                                              ; test sign
             GOTO
                          NRM4032
                          WREG,F
             CLRF
                                              ; if < 0, negate, set MSB in SIGN
             COMF
                          AARGB3,F
             COMF
                          AARGB2,F
             COMF
                          AARGB1,F
                          AARGB0,F
             COMF
             INCF
                          AARGB3,F
             ADDWFC
                          AARGB2,F
                          AARGB1,F
             ADDWFC
             ADDWFC
                          AARGB0,F
```

```
Normalization routine
        Input: 40 bit unnormalized floating point number in AEXP, AARGBO, 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 \text{ clks}
                                                          RND = 0
                                                          RND = 1, SAT = 0
                         27+22+24+4 = 77 clks
                         27+22+23+11 = 83 clks
                                                          RND = 1, SAT = 1
                         4+8+8+6+6 = 32 \text{ clks}
                                                          AARG = 0
;
        Min Timing:
                         3+5+4+8+6 = 26 \text{ clks}
        PM: 178
                                                          DM: 8
NRM4032
                CLRF
                                 TEMP,W
                                                          ; clear exponent decrement
                                 AARGB0
                                                          ; test if highbyte=0
                CPFSGT
                GOTO
                                 NRM4032A
TNIB4032
                MOVLW
                                 0xF0
                                                          ; test if highnibble=0
                ANDWF
                                 AARGB0,W
                TSTFSZ
                                 WREG
                COTO
                                 NORM4032
                                                          ; if so, shift 4 bits
                SWAPF
                                 AARGB0,F
                SWAPF
                                 AARGB1,W
                ANDLW
                                 0x0F
                ADDWF
                                 AARGB0,F
                                 AARGB1,W
                SWAPF
                ANDLW
                                 0xF0
                                 WREG, AARGB1
                MOVPF
                SWAPF
                                 AARGB2,W
                ANDLW
                                 0 \times 0 F
                ADDWF
                                 AARGB1,F
                SWAPF
                                 AARGB2,W
                ANDT.W
                                 0xF0
                MOVPF
                                 WREG, AARGB2
                SWAPF
                                 AARGB3,W
                ANDLW
                                 0x0F
                                 AARGB2,F
                ADDWF
                SWAPF
                                 AARGB3,W
                ANDLW
                                 0xF0
                MOVPF
                                 WREG, AARGB3
                BSF
                                 TEMP, 2
                                                          ; increase decrement by 4
NORM4032
                BCF
                                                          ; clear carry bit
                                 _C
                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	

	V DDME	AADCD1 E	
	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	AARGBO,MSB	
	GOTO	TNORMUN4032	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F TEMP,F	
	INCF BTFSC		: gingo highnibblo I= 0 at most
	GOTO	AARGB0,MSB TNORMUN4032	<pre>; since highnibble != 0, at most ; 3 left shifts are required</pre>
	RLCF	AARGB2,F	, 3 lett shirts are required
	RLCF	AARGB1,F	
	RLCF	AARGBO,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	
ENT D 4 0 2 0 D	MOLITIE	070	. took if highwibble 0
TNIB4032B	MOVLW	0xF0	; test if highnibble=0
	ANDWF	AARGB0,W	
	TSTFSZ GOTO	WREG NORM4032B	
	G010	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	AARGBO, MSB	
	GOTO RLCF	TNORMUN4032 AARGB1,F	
	RLCF	AARGB1,F	
	INCF	TEMP, F	

```
BTFSC
                            AARGB0,MSB
                                                  ; since highnibble != 0, at most
                                                  ; 3 left shifts are required
              GOTO
                            TNORMUN4032
              RLCF
                            AARGB1,F
              RLCF
                            AARGB0,F
                            TEMP,F
              INCF
              COTO
                            TNORMUN4032
NRM4032C
              MOVFP
                            AARGB1,AARGB0
                                                  ; shift 8 bits by move
              CLRF
                            AARGB1,W
                            TEMP, 3
                                                  ; increase decrement by 8
              BSF
              CPFSGT
                            AARGB0
                                                  ; if highbyte=0, result=0
                            RES032
              GOTO
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
                                                  ; if MSB=1, normalization done
              BTFSC
                            AARGB0,MSB
              GOTO
                            TNORMUN4032
              RLCF
                                                  ; otherwise, shift left and
                            AARGB0,F
              INCF
                            TEMP,F
                                                  ; increment decrement
                            AARGB0,MSB
              BTFSC
              GOTO
                            TNORMUN4032
              RLCF
                            AARGB0,F
              INCF
                            TEMP,F
              BTFSC
                            AARGB0,MSB
                                                  ; since highnibble != 0, at most
                                                  ; 3 left shifts are required
              GOTO
                            TNORMUN4032
                            AARGB0,F
              RLCF
              INCF
                            TEMP,F
              GOTO
                            TNORMUN4032
Float to integer conversion
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Use:
              CALL
                     INT3224
                                   or
                                          CALL
                                                 INT32
       Output: 24 bit 2's complement integer right justified in AARGBO, AARGB1, AARGB2
       Result: AARG <-- INT( AARG )
                    10+45+15 = 70 \text{ clks}
                                                  RND = 0
       Max Timing:
                     10+45+24 = 79 clks
                                                  RND = 1, SAT = 0
                     10+45+26 = 81 \text{ clks}
                                                  RND = 1, SAT = 1
      Min Timing:
                     4 clks
      PM: 183
                                                  DM: 8
INT3224
INT32
                            AARGB3,W
              CPFSGT
                            EXP
                                                 ; test for zero argument
              RETLW
                            0x00
```

	MOVPF BSF	AARGB0,SIGN AARGB0,MSB	<pre>; save sign in SIGN ; make MSB explicit</pre>
	MOVLW SUBWF	EXPBIAS+D'23' EXP,W	; remove bias+23 from EXP
	BTFSS GOTO NEGW	WREG,MSB SETIOV3224 EXP,F	<pre>; if >= 23, integer overflow ; will occur</pre>
	MOVLW CPFSGT	7 EXP	; do byte shift if EXP >= 8
	GOTO SUBWF	SNIB3224 EXP,F	; EXP = EXP - 7
	MOVFP MOVFP MOVFP	AARGB2,AARGB3 AARGB1,AARGB2 AARGB0,AARGB1	; save for rounding
	CLRF	AARGB0,F	
	DCFSNZ GOTO	EXP,F SHIFT3224OK	; EXP = EXP - 1 ; shift completed if EXP = 0
	GOTO	SHIF132240K	, shirt completed if EXP = 0
	CPFSGT GOTO	EXP SNIB3224A	<pre>; do another byte shift if EXP >= 8</pre>
	SUBWF	EXP,F	; $EXP = EXP - 7$
	MOVED	AARGB2, AARGB3	; save for rounding
	MOVFP CLRF	AARGB1,AARGB2 AARGB1,F	
	DCFSNZ	EXP,F	; $EXP = EXP - 1$
	GOTO	SHIFT32240K	; shift completed if $EXP = 0$
	CPFSGT	EXP	; do another byte shift if EXP >= 8
	GOTO	SNIB3224B	_
	SUBWF	EXP,F	; EXP = EXP - 7
	MOVFP CLRF	AARGB2,AARGB3 AARGB2,F	; save for rounding
	DCFSNZ	EXP,F	; $EXP = EXP - 1$
	GOTO	SHIFT32240K	; shift completed if $EXP = 0$
SNIB3224C	MOVLW	3	; do nibble shift if EXP >= 4
DIVIDUE	CPFSGT	EXP	, do hisbic shift if him , - 1
	GOTO	SHIFT3224C	
	SWAPF	AARGB3,W	
	ANDLW	0×0F	
	MOVPF	WREG, AARGB3	· abift gomploted if EVD - 0
	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	. which warmland if DVD 0
	GOTO BCF	SHIFT3224OK _C	; shift completed if EXP = 0
	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 MOVPF	AARGB2,W 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	, purity domproduct in him
	RRCF	AARGB2,F	
	RRCF	AARGB3,F	
	GOTO	SHIFT3224OK	
	G010	SHIF 132240K	
SNIB3224A	MOVLW	3	; do nibble shift if EXP >= 4
DIVIDUELIII	CPFSGT	EXP	r do nibble bille il bili r i
	GOTO	SHIFT3224A	
	SUBWF		; $EXP = EXP - 3$
		EXP,F	/ EAP = EAP - 3
	SWAPF	AARGB2,W	
	MOVPF	WREG, AARGB3	; save for rounding
	ANDLW	0x0F	
	MOVPF	WREG,AARGB2	
		3.3.D.G.D.1	
	SWAPF	AARGB1,W	
	ANDLW	0xF0	
	ADDWF	AARGB2,F	
	CIVA DE	AADOD1 M	
	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
SHIFIJZZAR	RRCF		; right shift by EXP
	RRCF	AARGB1,F	/ light shift by EAP
		AARGB2,F	
	RRCF	AARGB3,F	
	DCFSNZ	EXP, F	· which was label is TWD 0
	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
OTT 1200 2 2 2 4	DOE	G.	·
SHIFT3224	BCF	_C	; at most 3 right shifts are required
	RRCF	AARGB0,F	; right shift by EXP
	RRCF	AARGB1,F AARGB2,F	
	RRCF		
	RRCF DCFSNZ	AARGB3,F EXP,F	
	GOTO	SHIFT32240K	; shift completed if EXP = 0
	BCF	_C	/ SHITE COMPTECED II EAF - 0
	RRCF	AARGB0,F	
	RRCF	AARGB1,F	
	RRCF	AARGB2,F	
	RRCF	AARGB3,F	
	DCFSNZ	EXP,F	
	GOTO	SHIFT32240K	; shift completed if EXP = 0
	BCF	_C	, Shill Completed II Em
	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	
INT32240K	DTFCC	SIGN MSP	; if sign bit set, negate
IN132240K	BTFSS	SIGN,MSB 0	, II sign bit set, negate
	RETLW	AARGB2,F	
	COMF COMF	AARGB1,F	
	COMF	AARGBO,F	
	INCF	AARGB2,F	
	CLRF	WREG,F	
	ADDWFC	AARGB1,F	
	ADDWFC	AARGBO,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
                           AARGRO.F
                                                ; saturate to largest two's
              BTFSS
                           SIGN, MSB
                                                ; complement 24 bit integer
              SETF
                           AARGB0,F
                                                 ; SIGN = 0, 0 \times 7 F FF FF
             MOVPF
                           AARGB0, AARGB1
                                                 ; SIGN = 1, 0x 80 00 00
             MOVPF
                           AARGB0,AARGB2
             RLCF
                           STGN.F
              RRCF
                           AARGB0,F
              RETLW
                            0xFF
                                                 ; return error code in WREG
Float to integer conversion
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Use: CALL
                   INT3232
       Output: 32 bit 2's complement integer right justified in AARGBO, AARGB1, AARGB2,
             AARGB3
      Result: AARG <-- INT( AARG )
                    11+54+17 = 82 \text{ clks}
                                                RND = 0
      Max Timing:
                    11+54+26 = 91 \text{ clks}
                                                RND = 1, SAT = 0
                    11+54+28 = 93 \text{ clks}
                                                RND = 1, SAT = 1
      Min Timing: 4 clks
      PM: 249
                                                 DM: 9
INT3232
             CLRF
                           AARGB3,W
              CPFSGT
                                                ; test for zero argument
                           EXP
             RETLW
                           0x00
             MOVPF
                           AARGB0,SIGN
                                               ; save sign in SIGN
              BSF
                           AARGB0,MSB
                                                 ; make MSB explicit
              CLRF
                            AARGB4,F
             MOVIW
                            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
                                                 ; do byte shift if EXP >= 8
              CPFSGT
                            EXP
              GOTO
                           SNIB3232
                                                 ; EXP = EXP - 7
              SUBWE
                           EXP,F
              MOVFP
                           AARGB3,AARGB4
                                                 ; save for rounding
              MOVFP
                           AARGB2,AARGB3
             MOVFP
                           AARGB1,AARGB2
             MOVFP
                           AARGB0, AARGB1
              CLRF
                            AARGB0,F
                                                 ; EXP = EXP - 1
              DCFSNZ
                            EXP,F
              GOTO
                            SHIFT3232OK
                                                 ; shift completed if EXP = 0
              CPFSGT
                           EXP
                                                 ; do another byte shift if EXP >= 8
              GOTO
                           SNIB3232A
                                                 i EXP = EXP - 7
              SUBWF
                           EXP,F
              MOVFP
                           AARGB3,AARGB4
                                                 ; save for rounding
              MOVFP
                           AARGB2,AARGB3
```

	MOVFP	AARGB1,AARGB2	
	CLRF	AARGB1, F	
	DCFSNZ	EXP,F	; EXP = EXP - 1
	GOTO	SHIFT32320K	; shift completed if EXP = 0
	G010	5111F 132320K	/ Shirt completed if EAF - 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	, pave for rounding
	DCFSNZ	EXP,F	; EXP = EXP - 1
	GOTO	SHIFT32320K	; shift completed if EXP = 0
	0010	51111 132320K	/ BHITE COMPTERED IT BM = 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	. 1 '6' 1 '6 5
	GOTO	SHIFT3232OK	; shift completed if EXP = 0
	BCF	_C	
	RRCF	AARGB4,F	
	DCFSNZ	EXP,F	116.
	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	5
	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
SHIF13232C		-	-
	RRCF	AARGB3,F	; right shift by EXP
	RRCF	AARGB4,F	
	DCFSNZ	EXP,F	: abift completed if EVD = 0
	GOTO	SHIFT3232OK	; shift completed if EXP = 0
	BCF	_C	
	RRCF	AARGB3,F	
	RRCF	AARGB4,F	
	DCFSNZ	EXP, F	. 116 116
	GOTO	SHIFT3232OK	; shift completed if EXP = 0
	BCF	_C	
	RRCF	AARGB3,F	
	RRCF	AARGB4,F	

	GOTO	SHIFT32320K	
SNIB3232B	MOVLW	3	; do nibble shift if EXP >= 4
	CPFSGT	EXP	
	GOTO	SHIFT3232B	
	SUBWF	EXP,F	; $EXP = EXP - 3$
	SWAPF	AARGB3,W	, 1111 – 1111 – 3
	MOVPF	WREG, AARGB4	; save for rounding
	ANDLW	0x0F	, save for rounding
	MOVPF	WREG, AARGB3	
	MOVEL	WILEG, AARGDS	
	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
CIII DU 2000	DOE	C	t at most 2 right shifts are required
SHIFT3232B	BCF	_C	<pre>; at most 3 right shifts are required ; right shift by EXP</pre>
	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	SHIFT32320K	
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	, right bille by im
	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	SHIFT32320K	; shift completed if EXP = 0
			/ Shire completed if Exr - 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	; $EXP = EXP - 3$
			, pur - pur 2
	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		
	ADDWF	AARGB2,F	
	CHADE	AADGD1 M	
	SWAPF	AARGB1,W	
	ANDLW	0x0F	
	MOVPF	WREG,AARGB1	
	SWAPF	AARGB0,W	
	ANDLW	0xF0	
	ADDWF	AARGB1,F	
	SWAPF	AARGB0,W	
	ANDLW	0x0F	
	MOVPF	WREG, AARGBO	
			; EXP = EXP - 1
	DCFSNZ	EXP,F	
	GOTO	SHIFT3232OK	; shift completed if EXP = 0
arr==-0.05 =	202		
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	SHIFT32320K	; shift completed if EXP = 0
			. DILLO COMPLECCO IL DIL - 0
	BCF	_C	
	RRCF	AARGB0,F	

```
RRCF
                            AARGB1,F
              RRCF
                            AARGB2,F
              RRCF
                            AARGB3,F
              RRCF
                            AARGB4,F
              DCFSNZ
                            EXP,F
                            SHIFT3232OK
                                                  ; shift completed if EXP = 0
              COTO
              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
                            INT32320K
              BSF
                            _C
                                                  ; set carry for rounding
                            0x80
              MOVLW
                                                  ; if NSB = 0x80, select even
              CPFSGT
                            AARGB4
              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
                            SETIOV3232
              GOTO
INT32320K
              BTFSS
                            SIGN, MSB
                                                  ; if sign bit set, negate
              RETLW
              COMF
                            AARGB3,F
              COME
                            AARGB2,F
              COME
                            AARGB1,F
              COMF
                            AARGB0,F
              INCF
                            AARGB3,F
              CLRF
                            WREG,F
              ADDWFC
                            AARGB2,F
              ADDWFC
                            AARGB1,F
              ADDWFC
                            AARGB0,F
              RETLW
SETIOV3232
              BSF
                                                 ; set integer overflow flag
                            FPFLAGS, IOV
              BTFSS
                            FPFLAGS, SAT
                                                 ; test for saturation
              RETLW
                                                 ; return error code in WREG
                            0xFF
              CLRF
                            AARGB0,F
                                                 ; saturate to largest two's
              BTFSS
                            SIGN, MSB
                                                 ; complement 32 bit integer
              SETF
                            AARGB0,F
                                                  ; SIGN = 0, 0x 7F 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, AARGBO, AARGB1, AARGB2
              32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
       Use:
              CALL
                     FPM32
```

```
Output: 32 bit floating point product in AEXP, AARGBO, AARGB1, AARGB2
;
        Result: AARG <-- AARG * BARG
                                                              RND = 0, SAT = 0
                         19+73+8 = 100 \text{ clks}
;
        Max Timing:
                         19+73+22 = 114 \text{ clks}
                                                              RND = 1, SAT = 0
;
                          19+73+30 = 122 \text{ clks}
                                                              RND = 1, SAT = 1
;
                         5+6 = 11 \text{ clks}
                                                              AARG * BARG = 0
        Min Timing:
        PM: 123
                                                              DM: 15
FPM32
                 CLRF
                                  AARGB3,W
                                                            ; test for zero arguments
                 CPFSEQ
                                  BEXP
                 CPFSGT
                                  AEXP
                                  RES032
                 GOTO
M32BNE0
                                  AARGB0, WREG
                 MOVFP
                 XORWF
                                  BARGB0,W
                 MOVPF
                                  WREG, SIGN
                                                            ; save sign in SIGN
                 MOVFP
                                  BEXP, WREG
                                  EXP,F
                 ADDWF
                 MOVLW
                                  EXPBIAS-1
                 BTFSS
                                  _C
                 GOTO
                                  MTUN32
                 SUBWF
                                  EXP,F
                                                             ; remove bias and overflow test
                 BTFSC
                                  _C
                                  SETFOV32
                 GOTO
                                  MOK32
                 COTO
MTUN32
                 SUBWF
                                  EXP,F
                                                            ; remove bias and underflow test
                 BTFSS
                                  _C
                                  SETFUN32
                 GOTO
MOK32
                                  AARGB0,MSB
                                                             ; make argument MSB's explicit
                 BSF
                 BSF
                                  BARGB0,MSB
                 MOVPF
                                  AARGB0,TEMPB0
                                                             ; multiply mantissas
                 MOVPF
                                  AARGB1, TEMPB1
                 MOVPF
                                  AARGB2, TEMPB2
                 MOVFP
                                  AARGB2, WREG
                 MIIIWE
                                  BARGB2
                 MOVPF
                                  PRODH, AARGB4
                 MOVFP
                                  AARGB1,WREG
                 MULWF
                                  BARGB1
                 MOVPF
                                  PRODH, AARGB2
                 MOVPF
                                  PRODL, AARGB3
                 MULWF
                                  BARGB2
                 MOVPF
                                  PRODL, WREG
                 ADDWF
                                  AARGB4,F
                                  PRODH, WREG
                 MOVPF
                 ADDWFC
                                  AARGB3,F
                 CLRF
                                  WREG,F
                 ADDWFC
                                  AARGB2,F
                 MOVFP
                                  TEMPB2, WREG
                 {\tt MULWF}
                                  BARGB1
                 MOVPF
                                  PRODL, WREG
```

ADDWF	AARGB4,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB3,F	
CLRF	WREG, F	
ADDWFC	AARGB2,F	
ADDWIC	AANGDZ , I	
	3.3.0.0.0. rm.n.c	
MOVFP	AARGB0,WREG	
MULWF	BARGB2	
MOVPF	PRODL, WREG	
ADDWF	AARGB3,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB2,F	
MOVFP	AARGBO, 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	AARGBO,F	
ADDWIC	AARODO , I	
MOMED	mempp1 wped	
MOVFP	TEMPB1,WREG	
MULWF	BARGB0	
MOVPF	PRODL, WREG	
ADDWF	AARGB2,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB1,F	
CLRF	WREG, F	
ADDWFC	AARGB0,F	
ADDWFC	AARGBU, F	
MOVFP	TEMPB0,WREG	
MULWF	BARGB0	
MOVPF	PRODL, WREG	
ADDWF	AARGB1,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB0,F	
	·	
BTFSC	AARGB0,MSB	; check for postnormalization
		/ Check for postnormalization
GOTO	MROUND32	
RLCF	AARGB3,F	
RLCF	AARGB2,F	
RLCF	AARGB1,F	
RLCF	AARGB0,F	
DECF	EXP,F	
BTFSC	_Z	
GOTO	SETFUN32	
0010	5211 01132	
D.T.E.C.C	EDELYCG DND	: is rounding anablada
BTFSC	FPFLAGS, RND	; is rounding enabled?
BTFSS	AARGB3,MSB	; is NSB > 0x80?
GOTO	MUL32OK	
BSF	_C	<pre>; set carry for rounding</pre>
MOVLW	0x80	
CPFSGT	AARGB3	; if NSB = $0x80$, select even
RRCF	AARGB2,W	; using lsb in carry
CLRF	WREG, F	- · · · · ·

MROUND32

```
ADDWFC
                            AARGB2.F
              ADDWFC
                            AARGB1,F
              ADDWFC
                            AARGB0,F
                            _C
              BTFSS
                                                  ; has rounding caused carryout?
                            MIII.320K
              COTO
              RRCF
                            AARGB0,F
                                                  ; if so, right shift
              RRCF
                            AARGB1,F
              RRCF
                            AARGB2,F
              TNFSNZ
                            EXP,F
                                                  ; test for floating point overflow
              GOTO
                            SETFOV32
MUL32OK
              BTFSS
                            SIGN, MSB
              BCF
                            AARGB0, MSB
                                                 ; clear explicit MSB if positive
              RETLW
SETFOV32
              BSF
                            FPFLAGS, FOV
                                                 ; set floating point underflag
                            FPFLAGS, SAT
                                                 ; test for saturation
              BTFSS
                                                 ; return error code in WREG
              RETLW
                            0xFF
              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
                                                  ; return error code in WREG
              RETIW
                            0xFF
Floating Point Divide
       Input: 32 bit floating point dividend in AEXP, AARGBO, AARGB1, AARGB2
              32 bit floating point divisor in BEXP, BARGBO, BARGB1, BARGB2
;
       Use:
              CALL
                     FPD32
       Output: 32 bit floating point quotient in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- AARG / BARG
;
                     10+34+69+80+111+11+8 = 323 \text{ clks} \text{ RND} = 0
       Max Timing:
                     10+34+69+80+111+11+22 = 337 \text{ clks RND} = 1, \text{ SAT} = 0
;
                     10+34+69+80+111+11+30 = 345 \text{ clks RND} = 1, \text{ SAT} = 1
       Min Timing:
                   6+6 = 12 \text{ 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
                                    LOW (IBTBL256I)
                ADDLW
                MOVWF
                                     TBLPTRL
                BTFSC
                                     _C
                TNCF
                                    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
                                    TEMPB2,F
                SUBWF
                                    PRODH, WREG
                MOVPF
                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.
                                    HIGH (IBTBL128I)
                MOVLW
                                                           ; access table for F0
                MOVWF
                                    TBLPTRH
                MOVFP
                                    BARGB0, WREG
                ANDLW
                                     0x7F
                ADDLW
                                    LOW (IBTBL128I)
                MOVWF
                                    TBLPTRL
                BTFSC
                                     _C
                                    TBLPTRH,F
                INCF
                TABLED
                                     0,1,TEMPB0
                TLRD
                                    1,TEMPB0
                TABLRD
                                     0,0,TEMPB1
                TLRD
                                    1,AARGB4
                                     0,AARGB5
                TLRD
```

; calculate difference

```
SUBWF
                                       TEMPB1,W
                 MOVWF
                                       AARGB5
                 MOVFP
                                       AARGB4, WREG
                 SUBWEB
                                       TEMPB0,W
                 MOVWF
                                       AARGB4
                                       AARGB5,TEMPB2
                 MOVPF
                 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
                                       AARGB5,F
                 ADDWFC
                 CLRF
                                       WREG, F
                 ADDWFC
                                       AARGB4,F
                 MOVED
                                       TEMPB2, WREG
                 MULWF
                                       BARGB1
                 MOVPF
                                       PRODL, WREG
                 ADDWF
                                       TBLPTRH, F
                 MOVPF
                                       PRODH, WREG
                 ADDWFC
                                       AARGB5,F
                                       WREG, F
                 CLRF
                 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
                                       BARGB0,W
                 RRCF
                 ANDLW
                                       0x3F
                 ADDLW
                                       LOW (IBTBL64I)
                 MOVWF
                                       {\tt TBLPTRL}
                 BTFSC
                                       _C
```

AARGB5, WREG

MOVFP

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	, carcarace arrectined
MOVWF	AARGB5	
MOVFP	AARGB4,WREG	
SUBWFB	TEMPBO, W	
MOVWF	AARGB4	
	3.3.5.6.1. mp.; pm.;	
MOVPF	AARGB4,TBLPTRH	
MOVPF	AARGB5,TBLPTRL	
MOVFP	BARGB2,WREG	
MULWF	AARGB5	
MOVPF	PRODH,TMR0H	
MOVPF	PRODL, TMROL	
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	
MOVIED	DADCD2 MDEC	
MOVFP MULWF	BARGB2,WREG 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	BARGBO, WREG	
ANDLW MULWF	0x01 TBLPTRH	
CLRF	AARGB4,W	
ADDWFC	AARGB1,W	
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 CLRF MOVFP SUBWF MOVFP SUBWFB MOVFP SUBWFB	TEMPB2, TBLPTRH TEMPB2, F TMR0H, WREG TEMPB2, F TBLPTRH, WREG TEMPB1, F AARGB5, WREG TEMPB0, F	; F0
;			
FPD32	CLRF CPFSGT GOTO	TEMPB3,W BEXP SETFDZ32	; clear exponent modification ; test for divide by zero
	CPFSGT GOTO	AEXP RES032	
D32BNE0	MOVFP XORWF	AARGB0,WREG BARGB0,W	
	MOVPF	WREG, SIGN	; save sign in SIGN
	BSF BSF	AARGB0,MSB BARGB0,MSB	; make argument MSB's explicit
FPD32SEED			; generation of F0
	MOVPF MOVPF MOVPF	AARGB0,AARGB5 AARGB1,TBLPTRH AARGB2,TBLPTRL	; A0 = F0 * A
	MOLITER		
	MOVFP MULWF	AARGB2,WREG 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 ADDWFC	WREG,F AARGB2,F	
	ADDWIC	AARODZ , I	
	MOVFP	TBLPTRL,WREG	
	MULWF	TEMPB1	
	MOVPF	PRODL, WREG	
	ADDWF	AARGB4,F	
	MOVPF	PRODH, WREG	
	ADDWFC	AARGB3,F	
	CLRF	WREG,F	
	ADDWFC	AARGB2,F	
	MOVFP	AARGBO, 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	AARGBO,W	
	ADDWFC	AARGB1,F	
	ADDWFC	AARGB0,F	
	MOVFP	TBLPTRH, WREG	
	MULWF	TEMPBO	
	MOVPF	PRODL, WREG	
	ADDWF	AARGB2,F	
	MOVPF	PRODH, WREG	
	ADDWFC	AARGB1,F	
	CLRF	WREG, F	
	ADDWFC	AARGBO,F	
	122112	1111020,1	
	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	
D3.01/2.0			
DAOK32	MOVIED		• D0 = E0 + D
	MOVED	BARGBO, AARGB5	; B0 = F0 * B
	MOVFP MOVFP	BARGB1,TBLPTRH BARGB2,TBLPTRL	
	PIOVEE	BARGBZ, IBBF IRB	
	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	
	MOVED	TRIPTRI MREC	
	MOVFP	TBLPTRL, WREG	

MULWF	TEMPB1	
MOVPF	PRODL, WREG	
ADDWF	AARGB4,F	
MOVPF	PRODH, WREG	
ADDWFC	BARGB3,F	
CLRF	WREG, F	
ADDWFC	BARGB2,F	
ADDWIC	BANGBZ, F	
MOVFP	BARGB0, WREG	
MULWF	TEMPB2	
MOVPF	PRODL, WREG	
ADDWF	BARGB3,F	
	·	
MOVPF	PRODH, WREG	
ADDWFC	BARGB2,F	
MOVFP	BARGBO, 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	BARGBO,F	
	•	
MOVFP	TBLPTRH, WREG	
MULWF	TEMPB0	
MOVPF	PRODL, WREG	
ADDWF	BARGB2,F	
MOVPF	PRODH, WREG	
MOVPF ADDWFC		
	BARGB1,F	
ADDWFC		
ADDWFC CLRF	BARGB1,F WREG,F	
ADDWFC CLRF	BARGB1,F WREG,F	
ADDWFC CLRF ADDWFC	BARGB1,F WREG,F BARGB0,F	
ADDWFC CLRF ADDWFC MOVFP	BARGB1,F WREG,F BARGB0,F AARGB5,WREG	
ADDWFC CLRF ADDWFC MOVFP MULWF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F PRODH,WREG	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F PRODH,WREG	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F PRODH,WREG BARGB0,F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F PRODH,WREG BARGB0,F BARGB0,MSB BARGB0,MSB BARGB0,MSB-1 DBOK32 AARGB4,F BARGB3,F BARGB2,F BARGB1,F BARGB0,F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF	BARGB1,F WREG,F BARGB0,F AARGB5,WREG TEMPB0 PRODL,WREG BARGB1,F PRODH,WREG BARGB0,F BARGB0,MSB BARGB0,MSB BARGB0,MSB-1 DBOK32 AARGB4,F BARGB3,F BARGB2,F BARGB1,F BARGB0,F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB1, F BARGB0, F TEMPB3, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF RLCF RLCF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB0, F TEMPB3, F	
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF COMF COMF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB1, F BARGB1, F BARGB0, F TEMPB3, F BARGB2, F BARGB2, F	; F] = 2 - RO
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF COMF COMF COMF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB1, F BARGB1, F BARGB1, F BARGB1, F BARGB3, F BARGB1, F	; F1 = 2 - B0
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF COMF COMF COMF COMF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB1, F BARGB0, F TEMPB3, F BARGB2, F BARGB1, F BARGB2, F BARGB0, F TEMPB3, F BARGB1, F BARGB0, F BARGB1, F BARGB0, F	; F1 = 2 - B0
ADDWFC CLRF ADDWFC MOVFP MULWF MOVPF ADDWF MOVPF ADDWFC BTFSS BTFSC GOTO RLCF RLCF RLCF RLCF RLCF RLCF COMF COMF COMF	BARGB1, F WREG, F BARGB0, F AARGB5, WREG TEMPB0 PRODL, WREG BARGB1, F PRODH, WREG BARGB0, F BARGB0, MSB BARGB0, MSB-1 DBOK32 AARGB4, F BARGB3, F BARGB2, F BARGB1, F BARGB1, F BARGB1, F BARGB1, F BARGB1, F BARGB3, F BARGB1, F	; F1 = 2 - B0

DBOK32

ADDWFC	BARGB2,F	
ADDWFC	BARGB1,F	
ADDWFC	BARGBO,F	
MOVPF	AARGB0,TEMPB0	; A1 = F1 * A0
	AARGB1, TEMPB1	/ AI - FI AO
MOVPF	•	
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	
ADDWITC	AARGD3,1	
MOMED	TOLDTOL MORC	
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	
11DDW1 C	THREEDZ / I	
MOVFP	AARGBO, WREG	
MULWF	BARGB2	
MOVPF	PRODL, WREG	
ADDWF	AARGB3,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB2,F	
MOVFP	AARGBO,WREG	
MULWF	BARGB1	
CLRF	AARGB1,W	
ADDWFC	AARGB1,F	
MOVPF	PRODL, WREG	
ADDWF	AARGB2,F	
MOVPF	PRODH, WREG	
ADDWFC	AARGB1,F	
MOVFP	TEMPBO, WREG	
	•	

MULWF	BARGB3	
MOVPF	PRODL, WREG	
	AARGB4,F	
ADDWF		
MOVPF	PRODH, WREG	
ADDWFC	AARGB3,F	
CLRF	WREG, F	
ADDWFC	AARGB2,F	
ADDWFC	AARGB1,F	
MOVFP	TEMPBO, WREG	
MULWF	BARGBO	
MOVPF	PRODH,AARGB0	
MOVPF	PRODL,WREG	
ADDWF	AARGB1,F	
CLRF	WREG, F	
ADDWFC	AARGB0,F	
MOVFP	TBLPTRL, WREG	
MULWF	BARGBO	
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		
	AARGBO,F	
DECF	TEMPB3,F	
BTFSC	AARGB0,MSB	
GOTO	DEXP32	
RLCF	AARGB3,F	
RLCF	AARGB2,F	
RLCF	AARGB1,F	
RLCF	AARGBO,F	
DECF	TEMPB3,F	
MOVFP	BEXP, WREG	; compute AEXP - BEXP
SUBWF	EXP,F	
MOVLW	EXPBIAS+1	; add bias + 1 for scaling of F0

DEXP32

	BTFSS	_C	
	GOTO	ALTB32	
AGEB32	ADDWF	TEMPB3,W	; if AEXP > BEXP, test for overflow
	ADDWF	EXP,F	
	BTFSC GOTO	_C SETFOV32	
	GOTO	DROUND32	
	G010	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	. 46 MOD 0-00 3
	CPFSGT	AARGB3	; if NSB = 0x80, select even
	RRCF CLRF	AARGB2,W WREG,F	; using lsb in carry
	ADDWFC	WRLG,F AARGB2,F	
	ADDWFC	AARGB1,F	
	ADDWFC	AARGBO,F	
	D==0.0	~	
	BTFSS	_C	; test if rounding caused carryout
	GOTO	DIV32OK	
	RRCF RRCF	AARGBO,F	
	RRCF	AARGB1,F AARGB2,F	
	INFSNZ	EXP,F	; test for overflow
	GOTO	SETFOV32	, test for overflow
DIV32OK	BTFSS	SIGN,MSB	
DIVSZOR	BCF	AARGBO,MSB	; clear explicit MSB if positive
	201	1111.020 /1.02	, order empriors has in 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	AARGBO,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
;			
; tab	ele for interpola	ating between consecuti	ive 16 bit approximations
	_	_	5 explicit bits of BARG as a pointer
; and	the remaining 1	17 explicit bits as the	e 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	0xA0A1
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

AN575

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 0xFE04DATA 0xFC10 DATA 0xFA23 0xF83E DATA DATA 0xF660 DATA 0xF48A 0xF2BA DATA DATA 0xF0F1 DATA 0xEF2F DATA 0xED730xEBBEDATA DATA 0xEA0F DATA 0xE866 DATA 0xE6C3 DATA 0xE526 0xE38EDATA 0xE1FC DATA 0xE070 DATA DATA 0xDEE9 DATA 0xDD68 DATA 0xDBEB DATA 0xDA74 DATA 0xD902 DATA 0xD794DATA 0xD62C DATA $0 \times D4C7$ DATA 0xD368 DATA 0xD20D DATA 0xD0B7 DATA 0xCF64DATA 0xCE17 DATA 0xCCCD DATA 0xCB87 DATA 0xCA46 DATA 0xC908 0xC7CE DATA 0xC698 DATA DATA 0xC566 DATA 0xC437 DATA 0xC30C DATA 0xC1E5 DATA 0xC0C1 DATA 0xBFA0 DATA 0xBE83 DATA 0xBD69 DATA 0xBC52 DATA 0xBB3F DATA 0xBA2F 0xB921 DATA DATA 0xB817 DATA 0xB710 DATA 0xB60B DATA 0xB50A DATA 0xB40B0xB30F DATA DATA 0xB216 DATA 0xB120

DATA

0xB02C

DATA	0xAF3B
DATA	0xAE4C
DATA	0xAD60
	0xAC77
DATA	
DATA	0xAB8F
DATA	0xAAAB
DATA	0xA9C8
DATA	0xA8E8
DATA	0xA80B
DATA	0xA72F
DATA	0xA656
DATA	0xA57F
DATA	0xA37F
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	$0 \times 90 FE$
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
	3250200

		0.0105	
	DATA	0x8185	
	DATA	0x8102	
	DATA DATA	0x8081 0x8001	
;			
;			between consecutive 16 bit approximations
; ;			th the top 8 explicit bits of BARG as a pointer oits as the argument to linear interpolation.
IBTBL256	S.T.		
IDIDDES	DATA	0xFFFF	
	DATA	0xFF01	
	DATA	0xFE04	
	DATA	0xFD09	
	DATA	0xFC10	
	DATA	0xFB19	
	DATA	0xFA23	
	DATA	0xF930	
	DATA	0xF83E	
	DATA	0xF74E	
	DATA	0xF660	
	DATA DATA	0xF574 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 0xE6C3	
	DATA 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 DATA	0xDBEB 0xDB2F	
	DATA	0xDB2F 0xDA74	
	DATA	0xDA74 0xD9BA	
	DATA	0xD9BA 0xD902	
	DATA	0xD84A	
	DATA	0xD794	
	DATA	0xD6DF	
	DATA	0xD62C	
	DATA	0xD579	

DATA

DATA

0xD4C7

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	0xC7CE
DATA	0xC698
	0xC5FE
DATA	
DATA	0xC566
DATA DATA	0xC4CE 0xC437
DATA	0xC437
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 DATA	0xA306 0xA29F
DATA	0xA23F
DATA	0xA250
DATA	0xA16B
DATA	0xA10B
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 0x95F8
DATA	0x95F8
DATA DATA	0x95A0 0x9549
DATA	0x9349 0x94F2
DATA	0x94F2 0x949C
	JAJ 196

DATA	0x9446
DATA	0x93F0
DATA	0x939B
DATA	0x9346
DATA	0x92F1
DATA	0x929D
DATA	0x9249
DATA	0x91F6
DATA	0x91F0
DATA	0x91A3
	0x9150
DATA	
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
	0x88D2
DATA	0x8889
DATA	
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, AARGBO, AARGB1, AARGB2
            32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
          CALL FPS32
      Use:
      Output: 32 bit floating point difference in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- AARG - BARG
      Max Timing:
                  1+160 = 161 \text{ clks}
                                           RND = 0
                  1+176 = 177 \text{ clks}
                                           RND = 1, SAT = 0
                  1+182 = 183 \text{ clks}
                                           RND = 1, SAT = 1
      Min Timing:
                 1+20 = 21 \text{ clks}
      PM: 425
                                           DM: 14
;------
                        BARGB0,MSB
                                           ; toggle sign bit for subtraction
Floating Point Add
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
            32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
      Use:
           CALL FPA32
      Output: 32 bit floating point sum in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- AARG - BARG
      Max Timing:
                  7+70+22+61 = 160 \text{ clks}
                                           RND = 0
                                           RND = 1, SAT = 0
                  7+70+22+77 = 176 \text{ clks}
                  7+70+22+83 = 182 \text{ clks}
                                           RND = 1, SAT = 1
      Min Timing: 6+14 = 20 clks
                                           DM: 14
     PM: 424
FPA32
            MOVFP
                        AARGB0, WREG
                                          ; exclusive or of signs in TEMPBO
            XORWF
                        BARGB0,W
            MOVPF
                        WREG, TEMPBO
            CLRF
                        AARGB3,F
            MOVFP
                        AEXP, WREG
                                           ; use AARG if AEXP >= BEXP
            CPFSGT
                        BEXP
                        USEA32
            GOTO
USEB32
                        BARGB0, WREG
            MOVFP
                                           ; use BARG if AEXP < BEXP
```

```
MOVPF
                                 WREG, SIGN
                                                           ; save sign in SIGN
                                                           ; make MSB's explicit
                 BSF
                                 BARGB0, MSB
                 BSF
                                 AARGB0,MSB
                 MOVED
                                 AEXP, WREG
                                                           ; compute shift count in BEXP
                 MOVPE
                                 WREG, TEMPB1
                                 BEXP, WREG
                 MOVFP
                 MOVPF
                                 WREG, AEXP
                                 WREG, F
                 CLRF
                 CPFSGT
                                 TEMPB1
                                                           ; return BARG if AARG = 0
                                  BRETURN32
                 GOTO
                 MOVFP
                                 TEMPB1, WREG
                 SUBWF
                                 BEXP,F
                 BTFSC
                                  _{\rm Z}
                 GOTO
                                 BLIGNED32
                 MOVLW
                                 7
                                                           ; do byte shift if BEXP >= 8
                                 BEXP
                 CPFSGT
                                 BNIB32
                 GOTO
                 SUBWF
                                 BEXP,F
                                                           ; BEXP = BEXP - 7
                 MOVED
                                 AARGB2, AARGB3
                                                           ; keep for postnormalization
                 MOVFP
                                 AARGB1, AARGB2
                 MOVFP
                                 AARGB0, AARGB1
                 CLRF
                                 AARGB0,F
                 DCFSNZ
                                 BEXP,F
                                                           ; BEXP = BEXP - 1
                                 BLIGNED32
                 GOTO
                 CPFSGT
                                 BEXP
                                                           ; do another byte shift if BEXP >= 8
                 GOTO
                                 BNIB32A
                                                           ; BEXP = BEXP - 7
                 SUBWF
                                 BEXP,F
                                 AARGB2,AARGB3
                 MOVFP
                                                           ; keep for postnormalization
                 MOVFP
                                 AARGB1, AARGB2
                 CLRF
                                 AARGB1,F
                 DCFSNZ
                                 BEXP,F
                                                           ; BEXP = BEXP -1
                 GOTO
                                 BLIGNED32
                 CPESGT
                                 BEXP
                                                           ; do another byte shift if BEXP >= 8
                                 BNIB32B
                 GOTO
                 SUBWF
                                 BEXP,F
                                                           ; BEXP = BEXP - 7
                 MOVFP
                                 AARGB2,AARGB3
                                                           ; keep for postnormalization
                 CLRF
                                 AARGB2,F
                 DCFSNZ
                                 BEXP,F
                                                           ; BEXP = BEXP - 1
                                 BLIGNED32
                 GOTO
                 CPFSGT
                                 BEXP
                                                           ; if BEXP still >= 8, then
                                 BNIB32C
                                                           ; AARG = 0 relative to BARG
                 GOTO
BRETURN32
                 MOVFP
                                  SIGN, AARGB0
                                                           ; return BARG
                 MOVFP
                                  BARGB1, AARGB1
                 MOVFP
                                 BARGB2, AARGB2
                 CLRF
                                 AARGB3,F
                 RETLW
                                 0 \times 00
BNIB32C
                 MOVLW
                                  3
                                                           ; do nibbleshift if BEXP >= 4
                 CPFSGT
                                 BEXP
                 GOTO
                                 BLOOP32C
                 SUBWF
                                                           ; BEXP = BEXP -3
                                 BEXP,F
                 SWAPF
                                 AARGB3,W
                 ANDLW
                                  0x0F
                 MOVPF
                                 WREG, AARGB3
                 DCFSNZ
                                 BEXP,F
                                                           ; BEXP = BEXP - 1
                 GOTO
                                 BLIGNED32
                                                           ; aligned if BEXP = 0
BLOOP32C
                                                           ; right shift by BEXP
                 BCF
                                  _C
```

	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 SWAPF	WREG, AARGB3	
	ANDLW	AARGB2,W 0xF0	
	ADDWF	AARGB3,F	
	SWAPF	AARGB2,W	
	ANDLW	0x0F	
	MOVPF	WREG, AARGB2	
	DCFSNZ	BEXP,F	; $BEXP = BEXP - 1$
	GOTO	BLIGNED32	; aligned if BEXP = 0
DT 00D 20D	202	a	
BLOOP32B	BCF RRCF	_C	; right shift by BEXP
	RRCF	AARGB2,F 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 MOVPF	0x0F 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	· DEVD _ DEVD 1
	DCFSNZ	BEXP, F	; BEXP = BEXP - 1 : aligned if BEXD - 0
	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
                                                           ; aligned if BEXP = 0
                GOTO
                                 BLIGNED32
                BCF
                                 _C
                RRCF
                                 AARGB1,F
                                 AARGB2,F
                RRCF
                RRCF
                                 AARGB3,F
                DCFSNZ
                                 BEXP,F
                                 BLIGNED32
                                                           ; aligned if BEXP = 0
                GOTO
                BCF
                                                           ; at most 3 right shifts are
                                 _C
                RRCF
                                 AARGB1,F
                                                           ; possible
                RRCF
                                 AARGB2,F
                RRCF
                                 AARGB3,F
                                 BLIGNED32
                GOTO
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
                                 WREG, AARGB2
                MOVPF
                SWAPF
                                 AARGB1,W
                ANDLW
                                 0xF0
                ADDWF
                                 AARGB2,F
                SWAPF
                                 AARGB1,W
                ANDI-W
                                 0x0F
                MOVPF
                                 WREG, AARGB1
                SWAPF
                                 AARGB0,W
                ANDLW
                                 0xF0
                ADDWF
                                 AARGB1,F
                SWAPF
                                 AARGB0,W
                ANDLW
                                 0x0F
                MOVPF
                                 WREG, AARGB0
                                                           ; BEXP = BEXP -1
                DCFSNZ
                                 BEXP,F
                                 BLIGNED32
                                                           ; aligned if BEXP = 0
                GOTO
BLOOP32
                BCF
                                                           ; right shift by BEXP
                                 _C
                RRCF
                                 AARGB0,F
                RRCF
                                 AARGB1,F
                                 AARGB2,F
                RRCF
                RRCF
                                 AARGB3,F
                DCFSNZ
                                 BEXP,F
                                                           ; aligned if BEXP = 0
                GOTO
                                 BLIGNED32
                BCF
                                 _C
                                 AARGB0,F
                RRCF
                RRCF
                                 AARGB1,F
                RRCF
                                 AARGB2,F
                RRCF
                                 AARGB3,F
                DCFSNZ
                                 BEXP,F
                GOTO
                                 BLIGNED32
                                                           ; aligned if BEXP = 0
                                                           ; at most 3 right shifts are
                BCF
                                 _C
                RRCF
                                 AARGB0,F
                                                           ; possible
                RRCF
                                 AARGB1,F
                RRCF
                                 AARGB2,F
                RRCF
                                 AARGB3,F
BLIGNED32
                CLRF
                                 BARGB3,W
                                                           ; negate if signs opposite
                BTFSS
                                 TEMPB0,MSB
```

	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	
	1121211	0110 0	
DATE 0.2.0	OI DE	DADGD2 E	
BNE032	CLRF	BARGB3,F	
	MOVPF	AARGB0,SIGN	; save sign in SIGN
	BSF	AARGB0,MSB	; make MSB's explicit
	BSF	BARGB0,MSB	
	MOVFP	DEAD MDEG	; compute shift count in BEXP
		BEXP, WREG	, compute shirt count in BEAP
	SUBWF	AEXP,W	
	MOVPF	WREG,BEXP	
	BTFSC	_Z	
	GOTO	ALIGNED32	
	MOVI W	7	
	MOVLW		1 1
	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	BARGBO,F	
	DCFSNZ	BEXP,F	; BEXP = BEXP - 1
			/ BEAF - BEAF - I
	GOTO	ALIGNED32	
	MOVLW	7	
	CPFSGT	BEXP	; do another byte shift if BEXP >= 8
	GOTO	ANIB32A	_
	SUBWF	BEXP, F	; BEXP = BEXP - 7
			/ BEAF - BEAF - /
	MOVFP	BARGB2, WREG	
	MOVPF	WREG, BARGB3	
	MOVFP	BARGB1,WREG	
	MOVPF	WREG, BARGB2	
	CLRF	BARGB1,F	
	DCFSNZ	BEXP,F	; BEXP = BEXP - 1
			/ BEAF - BEAF - I
	GOTO	ALIGNED32	
	MOVLW	7	
	CPFSGT	BEXP	; do another byte shift if BEXP >= 8
	GOTO	ANIB32B	- -
	SUBWF	BEXP, F	; BEXP = BEXP - 7
			, DEAF - DEAF - /
	MOVFP	BARGB2, WREG	
	MOVPF	WREG,BARGB3	
	CLRF	BARGB2,F	
	DCFSNZ	BEXP,F	; BEXP = BEXP - 1
	GOTO	ALIGNED32	
	0010		
	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	· DEVD DEVD 1
	DCFSNZ	BEXP, F	; BEXP = BEXP - 1
	GOTO	ALIGNED32	; aligned if BEXP = 0
ALOOP32C	BCF	_C	; right shift by BEXP
	RRCF	BARGB3,F	5
	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	
		2	. 1 1111 116, 16 ppm
ANIB32B	MOVLW	3 DEVD	; do nibbleshift if BEXP >= 4
	CPFSGT GOTO	BEXP ALOOP32B	
	SUBWF	BEXP,F	; BEXP = BEXP -3
	SWAPF	BARGB3,W	/ DEAF - DEAF -5
	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 $BEXP = 0$
** 00D 2 2D	DOE	G.	. wishe shift has DEVD
ALOOP32B	BCF RRCF	_C	; right shift by BEXP
	RRCF	BARGB2,F 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	MOVT.IM	3	; do nibbleshift if BEXP >= 4
WINTDOOM	MOVLW CPFSGT	BEXP	, do hiddleshill il bear >= 4
	GOTO	ALOOP32A	
	SUBWF	BEXP,F	; BEXP = BEXP -3
	SWAPF	BARGB3,W	. 22 22 3
	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
	G010	ALIGNEDSZ	/ aligned if bear = 0
ALOOP32A	BCF	_C	; right shift by BEXP
112001 0211	RRCF	BARGB1,F	, right burie by bank
	RRCF	BARGB2,F	
	RRCF	BARGB3,F	
	DCFSNZ	BEXP,F	· alienad if DEVD 0
	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	
		BEXP,F	
	DCFSNZ		
	DCFSNZ GOTO		; aligned if $REXP = 0$
	GOTO BCF	ALIGNED32 _C	; aligned if BEXP = 0

	DD 45	D3D6D0 =	
	RRCF	BARGBO, 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	, ada
	MOVFP	BARGB1,WREG	
	ADDWFC	AARGB1,F	
		BARGBO, WREG	
	MOVFP		
	ADDWFC	AARGB0,F	
	BTFSC	TEMPB0,MSB	
	GOTO	ACOMP32	
	BTFSS		
		_C NRMRND4032	
	GOTO	NRMRND4032	
	RRCF	AARGB0,F	; shift right and increment EXP
	RRCF	AARGB1,F	, pulle right and merement bar
	RRCF	AARGB2,F	
	RRCF	AARGB3,F	
	INCFSZ	AEXP, F	
	GOTO	NRMRND4032	
	GOTO	SETFOV32	; set floating point overflow flag
ACOMP32	BTFSC	_C	
ACOM 32		NRM4032	; normalize and fix sign
	GOTO		/ normalize and lix sign
	CLRF	WREG, F	: nogato togglo sign bit and
	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.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.





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, KEELO© 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 Divvasree 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 - Ier 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