

# **Floating Point Math Functions**

Author: Frank.J. Testa FJT Consulting

#### INTRODUCTION

rand(x)

This application note presents implementations of the following math routines for the Microchip PICmicro™ microcontroller family:

	•
$\operatorname{sqrt}(\boldsymbol{x})$	square root function, $\sqrt{x}$
$\exp(x)$	exponential function, $e^x$
$\exp 10(x)$	base 10 exponential function, $10^x$
$\log(x)$	natural log function, $\ln x$
log10(x)	common log function, $\log 10 x$
sin(x)	trigonometric sine function
cos(x)	trigonometric cosine function
$\sin\cos(x)$	trigonometric sine and cosine functions
pow(x, y)	power function, $x^y$
floor(x)	floor function, largest integer not greater than x, as float, $\lfloor x \rfloor$
taxxb(a, b)	floating point logical comparison tests

Routines for the PIC16CXXX and PIC17CXXX families are provided in a modified IEEE 754 32-bit format together with versions in 24-bit reduced format.

integer random number generator

The techniques and methods of approximation presented here attempt to balance the usually conflicting goals of execution speed verses memory consumption, while still achieving full machine precision estimates. Although 32-bit arithmetic routines are available and constitute extended precision for the 24-bit versions, no extended precision routines are currently supported for use in the 32-bit routines, thereby requiring more sophisticated error control algorithms for full or nearly full machine precision function estimation. Differences in algorithms used for the PIC16CXXX and PIC17CXXX families are a result of performance and memory considerations and reflect the significant platform dependence in algorithm design.

# MATHEMATICAL FUNCTION EVALUATION

Evaluation of elementary and mathematical functions is an important part of scientific and engineering computing. Although straightforward Taylor series approximations for many functions of interest are well known, they are generally not optimal for high performance function evaluation. Many other approaches are available and the proper choice is based on the relative speeds of floating point and fixed point arithmetic operations and therefore is heavily implementation dependent.

Although the precision of fixed point arithmetic is usually discussed in terms of absolute error, floating point calculations are typically analyzed using relative error. For example, given a function f and approximation p, absolute error and relative error are defined by

abs error 
$$\equiv |p - f|$$
 rel error  $\equiv \left| \frac{p - f}{f} \right|$ 

In binary arithmetic, an absolute error criterion reflects the number of correct bits to the right of the binary point, while a relative error standard determines the number of significant bits in a binary representation and is in the form of a percentage.

In the 24-bit reduced format case, the availability of extended precision arithmetic routines permits strict 0.5\*ulp, or one-half **U**nit in the **L**ast **P**osition, accuracy, reflecting a relative error standard that is typical of most floating point operations. The 32-bit versions cannot meet this in all cases. The absence of extended precision arithmetic requires more time consuming pseudo extended precision techniques to only approach this standard. Although noticeably smaller in most cases, the worst case relative error is usually less than 1\*ulp for the 32-bit format. Most of the approximations, presented here on the PIC16CXXX and PIC17CXXX processors, utilize minimax polynomial or minimax rational approximations together with range reduction and some segmentation of the interval on the transformed argument. Such segmentation is employed only when it occurs naturally from the range reduction, or when the gain in performance is worth the increased consumption of program memory.

#### RANGE REDUCTION

Since most functions of scientific interest have large domains, function identities are typically used to map the argument to a considerably smaller region where accurate approximations require a reasonable effort. In most cases range reduction must be performed carefully in order to prevent the introduction of cancellation error to the approximation. Although this process can be straightforward when extended precision routines are available, their unavailability requires more complex pseudo extended precision methods[3,4]. The resulting interval on the transformed argument sometimes naturally suggests a segmented representation where dedicated approximations are employed in each subinterval. In the case of the trigonometric functions  $\sin(x)$  and  $\cos(x)$ , reduction of the infinite natural domain to a region small enough to effectively employ approximation cannot be performed accurately for an arbitrarily large x using finite precision arithmetic, resulting in a threshold in |x| beyond which a loss of precision occurs. The magnitude of this threshold is implementation dependent.

#### MINIMAX APPROXIMATION

Although series expansions for the elementary functions are well known, their convergence is frequently slow and they usually do not constitute the most computationally efficient method of approximation. For example, the exponential function has the Maclaurin series expansion given by

$$e^{x} = \sum_{j=0}^{\infty} \frac{x^{j}}{j!} = 1 + x + \frac{x^{2}}{2!} + \frac{x^{3}}{3!} + \dots$$

To estimate the function on the interval [0,1], truncation of the series to the first two terms yields the linear approximation,

$$e^x \approx 1 + x$$
.

a straight line tangent to the graph of the exponential function at x=0. On the interval [0,1], this approximation has a minimum relative error of zero at x=0, and a maximum relative error of |2-e|/e=0.26424 at x=1, underestimating the function throughout the interval. Recognizing that this undesirable situation is in part caused by using a tangent line approximation at one of the endpoints, an improvement could be made by using a tangent line approximation, for example, at the midpoint x=0.5, yielding the linear function,

$$e^x \approx e^{1/2}(x+0.5),$$

with a minimum relative error of zero at x = 0.5, a maximum relative error of 0.17564 at x = 0, and relative error of 0.09020 at x = 1, again underestimating the function throughout the interval. We could reduce the maximum error even further by adjusting the intercept of the above approximation, producing subintervals of

both positive and negative error, together with possibly equalizing the values of maximum error at each occurrence by manipulating both the slope and intercept of the linear approximation. This is a simple example of a very powerful result in approximation theory known as minimax approximation, whereby a polynomial approximation of degree n to a continuous function can always be found such that the maximum error is a minimum, and that the maximum error must occur at least at n + 2 points with alternating sign within the interval of approximation. It is important to note that the resulting minimax approximation depends on the choice of a relative or absolute error criterion. The evaluation of the minimax coefficients is difficult, usually requiring an iterative procedure known as Remes' method, and historically accounting for the attention given to near-minimax approximations such as Chebyshev polynomials because of greater ease of computation. With the advances in computing power, Remes' method has become much more tractable, resulting in iterative procedures for minimax coefficient evaluation[3]. Remarkably, this theory can be generalized to rational functions, offering a richer set of approximation methods in cases where division is not too slow. In the above simple example, the minimax linear approximation on the interval [0,1] is given by

$$e^x \approx 1.71828x + 0.89407$$
  
max error = 0.10593,

with a maximum relative error of 0.10593, occurring with alternating signs at the n + 2 = 3 points (x = 0, x = 0.5413, and x = 1). Occasionally, constrained minimax approximation[ $\overline{2}$ ] can be useful in that some coefficients can be required to take on specific values because of other considerations, leading to effectively near-minimax approximations.

The great advantage in using minimax approximations lies in the fact that minimizing the maximum error leads to the fewest number of terms required to meet a given precision. The number of terms is also dramatically affected by the size of the interval of approximation[1], leading to the concept of segmented representations, where the interval of approximation is split into sub-intervals, each with a dedicated minimax approximation. For the above example, the interval [0,1] can be split into the subintervals [0,0.5] and [0.5,1], with the linear minimax approximations given by

$$e^{\mathbf{x}} \approx \begin{cases} 1.29744\mathbf{x} + 0.97980, [0, 0.5], \text{ max error } = 0.02020 \\ 2.13912\mathbf{x} + 0.54585, [0.5, 1], \text{ max error } = 0.03331. \end{cases}$$

Since the subintervals were selected for convenience, the maximum relative error is different for the two sub-intervals but nevertheless represents a significant improvement over a single approximation on the interval [0,1], with the maximum error reduced by a factor greater than three. Although a better choice for the split, equalizing the maximum error over the subinter-

vals, can be found, the overhead in finding the correct subinterval for a given argument would be much greater than that for the convenient choice used above. The minimax approximations used in the implementations for the PIC16CXXX and PIC17CXXX device families presented here, have been produced by applying Remes' method to the specific intervals in question[3].

#### **USAGE**

For the unary operations, input argument and result are in AARG, with the exception of the sincos routines where the cosine is returned in AARG and the sine in BARG. The power function requires input arguments in AARG and BARG, and produces the result in AARG. Although the logical test routines also require input arguments in AARG and BARG, the result is returned in the W register.

#### **SQUARE ROOT FUNCTION**

The natural domain of the square root function is all nonnegative numbers, leading to the effective domain [0,MAXNUM] for the given floating point representation. All routines begin with a domain test on the argument, returning a domain error if outside the above interval.

On the PIC17CXXX, the greater abundance of program memory together with improved floating point division, using the hardware multiply permits a standard Newton-Raphson iterative approach for square root evaluation[1]. Range reduction is produced naturally by the floating point representation,

$$x = f \cdot 2^e$$
, where  $1 \le f < 2$ ,

leading to the expression

$$\sqrt{x} = \begin{cases} \sqrt{f} \cdot 2^{e/2}, & e \text{ even} \\ \sqrt{f} \cdot \sqrt{2} \cdot 2^{e/2}, & e \text{ odd} \end{cases}$$

The approximation to  $\sqrt{f}$  utilizes a table lookup of 16-bit estimates of the square root as a seed to a single Newton-Raphson iteration

$$y = \left(y_0 + \frac{f}{y_0}\right)/2,$$

where the precision of the result is guaranteed by the precision of the seed and the quadratic conversion of the method, whereby the number of significant bits is doubled upon each iteration. For the 24-bit case, the seed is generated by zeroth degree minimax approximations, while in the 32-bit case, linear interpolation between consecutive square root estimates is employed.

Because of limited memory on the PIC16CXXX as well as a slower divide routine, alternative methods must be used.

For the 24-bit format, the approximation to  $\sqrt{f}$  is obtained from segmented fourth degree minimax polynomials on the intervals [1,1.5] and [1.5,2.0]. In the 32-bit case, the function  $\sqrt{f}=\sqrt{1+z}$  on the interval [0,1] in z, is obtained from a minimax rational approximation of the form

$$\sqrt{1+z} \approx 1 + z \frac{p(z)}{q(z)}$$
, where  $z \equiv f - 1$ .

#### **EXPONENTIAL FUNCTIONS**

While the actual domain of the exponential function consists of all the real numbers, a limitation must be made to reflect the finite range of the given floating point representation. In our case, this leads to the effective domain for the exponential function [MINLOG,MAXLOG], where

$$MINLOG \equiv ln(2^{-126}) \qquad MAXLOG \equiv ln(2^{128}).$$

All routines begin with a domain test on the argument returning a domain error if outside the above interval.

For the 24-bit reduced format, given the availability of extended precision routines, the exponential function is evaluated using the identity

$$e^x = 2^{x/\ln 2} = 2^{n+z} = 2^n \cdot 2^z$$

where n is an integer and  $0 \le z < 1$ . Range reduction is performed by first finding the integer n and then computing z. The base two exponential function is then approximated by third degree minimax polynomials in a segmented representation on the subintervals [0,0.25], [0.25,0.5], [0.5,0.75] and [0.75,1.0], permitting 0.5\*ulp accuracy throughout the domain [MINLOG,MAXLOG].

For the 32-bit modified IEEE format, the lack of extended precision routines requires a more complex algorithm to approach a 0.5\*ulp standard in most cases, leading to a worst case error less than 1\*ulp. The exponential function in this case is based on the expansion

$$e^x = e^{z + n \ln 2} = 2^n \cdot e^z ,$$

where n is an integer and  $-0.5 \ln 2 \le z < 0.5 \ln 2$ , with the exponential function evaluated on this interval using segmented fifth degree minimax approximations on the subintervals  $[-0.5 \ln 2, 0]$  and  $[0, 0.5 \ln 2]$ .

During range reduction, the integer n is first evaluated and then the transformed argument z is obtained from the expression  $z = x - n \ln 2$ .

Because of the problem of serious cancellation error in this difference, pseudo extended precision methods have been developed  $[\overline{4}]$ , where  $\ln 2$  is decomposed into a number close to  $\ln 2$  but containing slightly more than

half its lower significant bits zero, and a much smaller residual number. Specifically, the decomposition given

$$by \quad \ln 2 = \boldsymbol{c}_1 - \boldsymbol{c}_2,$$

where  $c_1 \equiv 0.693359375$ 

and  $c_2 \equiv 0.00021219444005469$ ,

produces the evaluation of z in the form

$$z = (x - n \cdot c_1) + n \cdot c_2,$$

where the term in parentheses is usually computed exactly, with only rounding errors present in the second term[3].

The base 10 exponential function routines for the reduced 24-bit and 32-bit formats are completely analogous to the standard exponential routines with the base e replaced by the base 10 in most places.

#### **LOG FUNCTIONS**

The effective domain for the natural log function is (0,MAXNUM], where MAXNUM is the largest number in the given floating point representation. All routines begin with a domain test on the argument, returning a domain error if outside the above interval.

For the 24-bit reduced format, given the availability of extended precision routines, the natural log function is evaluated using the identity[1]

$$\ln x = \ln 2 \cdot \log_2 x = \ln 2 \cdot (n + \log_2 f),$$

where n is an integer and  $0.5 \le f < 1$ . The final argument z is obtained through the additional transformation  $\overline{3}$ 

$$z \equiv \begin{cases} 2f - 1, & n = n - 1, f < 1/\sqrt{2}, \\ f - 1, & otherwise \end{cases}$$

naturally leading to a segmented representation of  $\log_2 f = \log_2 (1+z)$  on the subintervals  $[1/\sqrt{2}-1,0]$  and  $[0,\sqrt{2}-1]$ , utilizing minimax rational approximations in the form

$$\log_2(1+z) \approx z \frac{p(z)}{q(z)},$$

where p(x) is linear and q(x) is quadratic in x.

For the 32-bit format, computation of the natural log is based on the alternative expansion[3]

$$\ln x = \ln f + \ln 2^n = \ln f + n \cdot \ln 2,$$

where n is an integer and  $0.5 \le f < 1$  . The final argument z is obtained through the additional transformation

$$z = \begin{cases} 2f - 1, n = n - 1, f < 1/\sqrt{2}, \\ f - 1, otherwise \end{cases}$$

naturally leading to a segmented representation of  $\ln f = \ln(1+z)$  on the subintervals  $[1/\sqrt{2}-1,0]$  and  $[0,\sqrt{2}-1]$ , using the effectively constrained minimax form [4] given by

$$\log_2(1+z) \approx z - 0.5 \cdot z^2 + z \left(z^2 \cdot \frac{p(z)}{q(z)}\right),$$

where p(x) is linear and q(x) is quadratic in x. The rationale for this form is that if the argument z is exact, the first term has no error and the second has only rounding error, thereby leading to more control over the propagation of rounding error than is possible in the simpler form used in the 24-bit case. The final step in the log evaluation is again performed in pseudo extended precision arithmetic in the form[ $\overline{3}$ ]

$$\ln f + \mathbf{n} \cdot \ln 2 = (\ln f - \mathbf{n} \cdot \mathbf{c}_2) + \mathbf{n} \cdot \mathbf{c}_1$$

where the decomposition of  $\ln 2$  is the same used in the exponential function.

The common logarithm routine for the reduced 24-bit format is completely analogous to the natural log routine with the base e replaced by the base 10 in most places. In the 32-bit case, the common log is obtained from the natural log through a standard conversion via fixed point multiplication by the common log of e in extended precision.

#### TRIGONOMETRIC FUNCTIONS

Evaluation of the sine and cosine functions, given their infinite natural domains, clearly requires careful range reduction techniques, especially in the absence of extended precision routines in the 32-bit format.

Susceptible to cancellation and roundoff errors, this process will always fail for arguments beyond some large threshold, leading to potentially serious loss of precision. The size of this threshold is heavily dependent on the range reduction algorithm and the available precision, leading to the value  $\boxed{3.4}$ 

$$LOSSTHR = \frac{\pi}{4} \cdot 2^{\frac{24}{2}} = 1024 \cdot \pi$$

for this implementation utilizing pseudo extended precision methods and the currently available fixed point and single precision floating point routines. A domain error is reported if this threshold is exceeded.

The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the alternative trigonometric argument z on  $\left[-\frac{\pi}{4},\frac{\pi}{4}\right]$ , through the definition[ $\overline{3}$ ]

$$z = x \mod \frac{\pi}{4}$$
,

produced by first evaluating y and j through the relations

$$y = \left\lfloor \frac{x}{\pi/4} \right\rfloor, \quad j = y - 8 \cdot \left\lfloor \frac{y}{8} \right\rfloor,$$

where j equals the correct octant. For j odd, adding one to j and y eliminates the odd octants. Additional logic on j and the sign of the result, representing a reflection of angles greater than  $\pi$  through the origin, leads to appropriate use of the sine or cosine routine in each case. The calculation of z is then obtained through a pseudo extended precision method [3,4]

$$z = x \mod \frac{\pi}{4} = x - y \cdot \frac{\pi}{4}$$
$$= ((x - p_1 \cdot y) - p_2 \cdot y) - p_3 \cdot y$$

where

$$\frac{\pi}{4} = \boldsymbol{p}_1 + \boldsymbol{p}_2 + \boldsymbol{p}_3, \quad \boldsymbol{p}_1 \approx \frac{\pi}{4} \quad \text{and} \quad \boldsymbol{p}_2 \approx \frac{\pi}{4} - \boldsymbol{p}_1$$

with

$$p_1 = 0.78515625$$
  
 $p_2 = 2.4187564849853515624 \times 10^{-4}$   
 $p_3 = 3.77489497744597636 \times 10^{-4}$ 

The numbers  $p_1$  and  $p_2$  are chosen to have an exact machine representation with slightly more than the lower half of the mantissa bits zero, typically leading to no error in computing the terms in parenthesis. This calculation breaks down leading to a loss of precision for x beyond the loss threshold or for x close to an integer multiple of  $\frac{\pi}{4}$ . In the latter case, the loss in precision is proportional to the size of y and the number of guard bits available. In the 32-bit modified IEEE implementation, an additional stage of pseudo extended precision is added to control error in this case, where  $p_3$  is chosen to have an exact machine representation with slightly more than the lower half of the mantissa bits zero and  $p_4$  is the residual.

$$p_3 = 3.7747668102383613583x10^{-8}$$
  
 $p_4 = 1.28167207614641725x10^{-12}$ 

Although some of the multiplications are performed in fixed point arithmetic, additions are all in floating point and therefore limited by the current single precision routines. It is useful to note that although only the sine and cosine are currently implemented, relatively simple modifications to this range reduction algorithm are necessary for evaluation of the remaining trigonometric functions.

Minimax polynomial expansions for the sine and cosine functions on the interval  $\begin{bmatrix} -\frac{\pi}{4}, \frac{\pi}{4} \end{bmatrix}$  are in the constrained forms[ $\overline{4}$ ]

$$\sin x \approx x + x \cdot x^{2} \cdot p(x^{2})$$

$$\cos x \approx 1 - 0.5 \cdot x^{2} + x^{4} \cdot q(x^{2})$$

for the full 32-bit single precision format, where p is degree three and q is degree two. In the reduced 24-bit format, we use the simpler forms

$$\sin x \approx x \cdot p(x^2)$$

$$\cos x \approx 1 - x^2 \cdot q(x^2)$$

where p and q are degree two. Because of the patently odd and even nature, respectively, of the sine and cosine functions, the minimax polynomial approximations were generated on the interval  $\begin{bmatrix} 0, \frac{\pi}{4} \end{bmatrix}$ . In addition to both sine and cosine routines, a  $\sin\cos(x)$  routine, utilizing only one range reduction calculation, is provided for those frequent situations where both the sine and cosine functions are needed, returning  $\cos(x)$  in AARG and  $\sin(x)$  in BARG. Generally, in the 32-bit case, these routines meet the 1\*ulp relative error performance criterion except in an extremely small number of cases as implied above. The reduced 24-bit format always meets the 0.5\*ulp criterion.

#### **POWER FUNCTION**

The power function  $x^y$ , while defined for all y with x>0, is clearly only defined for negative x when y is an integer or an odd root. Unfortunately, odd fractions such as 1/3 for the cube root, cannot be represented exactly in a binary floating point representation, thereby posing problems in defining and recognizing such cases. Therefore, since an integer data type for y in this function is not currently supported, the domain of the power function will be restricted to the interval [0,MAXNUM] for x and [-MAXNUM,MAXNUM] for y, subject to the requirement that the range is also [0,MAXNUM]. In addition, the following special cases will be satisfied:

$$x^0 \equiv 1, \quad x \ge 0$$

$$0^y \equiv MAXNUM, \quad y < 0,$$

where MAXNUM will be returned through the floating point overflow and saturate if enabled. When extended precision routines are available, evaluation of the power function  $x^y$  is usually performed through direct calculation using the identity

$$x^y = \exp(y \cdot \ln x)$$

relying on the extended precision evaluation of the log and exponential functions for control of error propagation. The implementation for the 24-bit format utilizes the 32-bit log and exponential functions to successfully meet the 0.5\*ulp relative error criterion.

The unavailability of extended precision routines for the 32-bit format requires considerably more effort with more sophisticated pseudo extended precision methods to control error propagation  $[\overline{3,4}]$ . Because the relative error in the exponential function is proportional to the absolute error of its argument  $[\overline{4}]$ , great care must be taken in any algorithm based on an exponential identity. Such methods generally rely on extracting as much of the result as an integer power of two as possible, followed by computations requiring approximations over a relatively small interval. To that end, consider the representation of the argument x given by

$$x = f \cdot 2^{e}$$
, where  $0.5 \le f < 1$ .

The power function can then be expressed in the form

$$x^y = 2^{y \cdot \log_2 x},$$

with the base  $2 \log of x$  represented as

$$\log_2 x = \log_2(f \cdot 2^e) = e + \log_2\left(\frac{a \cdot f}{a}\right)$$
$$= e + \log_2 a + \log_2\left(1 + \frac{f - a}{a}\right),$$

where a is chosen so that (f-a)/a is small. Rather than a single value of a, we choose a set of values of the form

$$a_k = 2^{-k/16}, \quad k = 0, 1..., 16$$

resulting in an effectively segmented representation  $[\overline{3.4}]$ . For a given f, the value of  $a_k$  for even k, nearest to f is chosen, resulting in an argument  $v=(f-a_k)/a_k$  to the function

$$\log_2(1+\nu)$$
,  $2^{-1/16}-1<\nu<2^{1/16}-1$ .

Since the numbers  $a_k$  cannot be represented exactly in full precision, psuedo extended precision evaluation of v is performed through the expansion

$$=\frac{(f-a_k)}{a_k}=\left(\frac{f-A_k-B_k}{A_k+B_k}\right)=\frac{f-A_k-(f\cdot C_k)}{A_k}$$
 
$$C_k\equiv\frac{B_k}{A_k},$$

where  $a_k = A_k + B_k$ . The number  $A_k$  is equal to  $a_k$  rounded to machine precision, and then  $B_k$  is the difference computed in higher precision. This method assures evaluation of  $\nu$  with a maximum relative error less than 1\*ulp. A minimax approximation of the form

$$\log(1+v) \approx v - \frac{v^2}{2} + v^3 \cdot \frac{p(v)}{q(v)},$$

with first degree polynomials p and q, is used to estimate  $\log(1+v)$ , followed by conversion to the required function  $\log_2(1+v)$ , leading to the result

$$\log_2 x = e - \frac{k}{16} + \log_2(1 + v).$$

The product  $y \cdot \log_2 x$  is now carefully computed by reducing the number y into a sum of two parts with one less than 1/16 and first evaluating small products of similar magnitude and collecting terms. Each stage of this strategy is followed by a similar reduction operation where the large part is an integer plus a number of 16ths. The final form of the product is then expressed as an integer plus a number of 16ths plus a number on the interval [-0.0625,0], leading to a final result expressed in the form

$$x^{y} = 2^{y \cdot \log_2 x} = 2^{i} \cdot 2^{-n/16} \cdot 2^{h}$$

where  $2^h$  is evaluated by a minimax approximation of the form

$$2^{\boldsymbol{h}} - 1 \approx \boldsymbol{h} + \boldsymbol{h} \cdot \boldsymbol{p}(\boldsymbol{h}),$$

with a second degree polynomial p. These elaborate measures for controlling error propagation are necessitated by attempting to obtain a full machine precision estimate without any extended precision routines. This is an especially difficult problem in the case of the power function since the relative error in the exponential function is proportional to the absolute error of its argument  $\boxed{4}$ . Notwithstanding these efforts, the

absence of a sticky bit in the floating point implementation leads to a maximum relative error of approximately 2\*ulp in a small number of cases. Currently, this function is only supported on the PIC17CXXX.

#### FLOOR FUNCTION

As a member of the standard C library of mathematical functions,  $floor(x) \equiv \lfloor x \rfloor$ , finds the largest integer not greater than x, as a floating point number. The implementation used here finds the location of the binary point implied by the exponent, thereby determining the number of low ordered bits to be zeroed. The bits are cleared by byte while greater than or equal to eight, and the remaining bits are cleared by a table lookup for the appropriate mask. When x is negative, the result is rounded down by one in the units position followed by a check for carry out and possible overflow.

```
FLOOR24(123.45) =
FLOOR24(0x8576E6) = 0x857600 = 123.0
FLOOR24(-123.45) =
FLOOR24(0x85F6E6) = 0x857800 = -124.0
```

# FLOATING POINT LOGICAL COMPARISON TESTS

Scientific computing frequently requires relational tests on floating point numbers with the operators < (less), <= (less or equal), > (greater), >= (greater or equal), == (equal), != (not equal). The necessary comparisons are made beginning with the exponent, followed if necessary by the mantissa bytes in the format in decreasing order of significance, all modulo the signs of the arguments. The arguments to be tested are placed in AARG and BARG, returning an integer result in the W register of one if the test is true and zero if false.

#### **INTEGER RANDOM NUMBER GENERATOR**

The utility function  $\operatorname{rand}()$  in the standard C library generates random nonnegative integers initially seeded by the related function  $\operatorname{srand}(x)$ , where x is an integer. This implementation of an integer random number generator uses a standard linear congruential method, based on the relation  $\boxed{6}$ 

$$x_{i+1} = (a \cdot x_i + c) \mod m,$$

with multiplier a, increment c, modulus m and initial seed  $x_0$ . Considerable research has yielded spectral methods for carefully selecting these constants to insure a maximum period together with other important performance criteria. Since the best such performance is usually associated with the largest word size, x is chosen here as a 32-bit integer, together with the following constants useful for this implementation [6]

$$a = 1664525,$$
 $c = 1,$ 
 $m = 2^{32},$ 

producing excellent results from standard spectral tests  $[\underline{\overline{0}}]$ . In this case, the value of m corresponds to the period of the generator, indicating that all possible 32-bit integers will be generated before any repetitions and leading to the corresponding definition

RAND\_MAX = 
$$2^{32} - 1 = 4294967295$$
.

Actually, the non-zero value of c is arbitrary for a good choice of the multiplier a with the restriction that it has no common factor with m. Although the calculation must be performed exactly, performance can be improved by recognizing that the binary representation of the multiplier a uses only three bytes, thereby requiring only a 32- by 24-bit fixed point multiply in the algorithm with no possible carryout after the addition of c, chosen here as c=1 for simplicity. It is important to note that the initial seed  $x_0$  may be chosen arbitrarily and the full 32-bit current value of c must saved between calls to preserve the efficacy of the method. Additional RAM locations, RANDBc, c0,1,2,3, have been added for this purpose and are not used by any other routine in the library.

Since the least significant bits of x are not very random, the best approach in constructing random integers over a given range is to view x/m as a random fraction between 0 and 1 with the binary point to the left of the MSb, and multiply by the desired integer range [6].

#### **EXAMPLES**

In evaluating any of the above functions, the appropriate PIC16CXXX or PIC17CXXX floating point values must be loaded into AARG for a unary operation, and AARG and BARG for a binary operation. For example, the argument x = 27.465 has the extended PICmicro<sup>TM</sup> microcontroller floating point representation 0x835BB851EB, leading to the 32-bit, rounded to the nearest number, 0x835BB852. An extended precision calculation of this nearest machine number is given by

27.465000152587890625, illustrating the effect of truncation error in floating point representations of even apparently simple numbers. Evaluation of sqrt(x) is then implemented as follows:

MOVLW	0x83
MOVWF	AEXP
MOVLW	0x5B
MOVWF	AARGB0
MOVLW	0xB8
MOVWF	AARGB1
MOVLW	0x52
MOVWF	AARGB3
CALT.	SORT32

If rounding is enabled, the 32-bit result in AARG is 0x8127B3DD. If rounding is disabled, an additional byte of guard bits is available contiguously and AARG = 0x8127B3DD00. For any of the other unary operations, simply call the appropriate function in place of the square root. Using the values x = 0x835BB852 and y = 0x8127B3DD, calls to the above functions yield the results shown in Table 1.

It is important to note that the exact PIC16CXXX results were computed on an extended precision calculator and converted to Microchip format using the exact decimal values of the 32-bit numbers x and y. The relative errors are all less than  $0.5^*$ ulp except for the sine function, where the error is slightly greater than  $0.5^*$ ulp, resulting in a rounded to the nearest result with a  $1^*$ ulp error.

On the PIC17CXXX, the fractional part of AARG resides in p-registers, thereby permitting direct register to register moves using the MOVFP and MOVPF instructions during loading of AARG and BARG from other RAM locations.

**TABLE 1: FUNCTION ROUTINES PERFORMANCE DATA** 

Routine	Unrounded PICmicro	Exact PICmicro	Decimal
SQRT32	0x8127B3DD00	0x8127B3DD39	5.24070607
EXP32	0xA64536D500	0xA64536D4DE	8.47028477x10 <sup>11</sup>
EXP1032	0xDA16D3D6E0	0xDA16D3D6AE	2.91742804x10 <sup>27</sup>
LOG32	0x805406C210 0x805406C208		3.31291247
LOG1032	0x7F3829EE22	0x7f3829EE1C	1.43877961
SIN32	0x7E394CC500	0x7E394CC459	7.23827621x10 <sup>-1</sup>
COS32	0x7EB0A29580	0x7EB0A295C5	-6.89980851x10 <sup>-1</sup>
POW32	0x9804563F38	0x9804563EC1	3.46913232x10 <sup>7</sup>

#### **APPENDIX A: PERFORMANCE DATA**

TABLE A-1: PIC17CXXX ELEMENTARY FUNCTION PERFORMANCE DATA

Routine	Max Cycles	Min Cycles	<b>Program Memory</b>	Data Memory	
SQRT24	327 6		325	7	
EXP24	999	645	339	5	
EXP1024	1002	646	339	5	
LOG24	1442	12	235	10	
LOG1024	1457	12	236	10	
SIN24	1625	834	317	11	
COS24	1637	942	317	11	
SINCOS24	2248	1516	339	15	
POW24	4255	2852	43	4	
FLOOR24	39	18	94	8	
TALTB24	27	8	43	6	
TALEB24	25	8	47	6	
TAGTB24	27	8	47	6	
TAGEB24	25	8	43	6	
TAEQB24	11	4	10	6	
TANEB24	11	4	10	6	
RND3224	21	3	20	5	
SQRT32	568	10	357	10	
EXP32	2024	14	374	15	
EXP1032	2084	14	392	15	
LOG32	2147	12	264	14	
LOG1032	2308	2001	31	1	
SIN32	2408	1338	462	11	
COS32	2405			11	
SINCOS32	3432	2328	482	15	
POW32	5574	4280 699		29	
FLOOR32	45	30	138	8	
RAND32	117	117	25	4	
TALTB32	33	8	59	8	
TALEB32	31	8	54	8	
TAGTB32	33	8	59	8	
TAGEB32	31	8	54	8	
TAEQB32	14	4	13	8	
TANEB32	14	4	13	8	
RND4032	23	3	22	6	

**Note:** Program and data memory values do not include dependency requirements.

TABLE A-2: PIC16CXXX ELEMENTARY FUNCTION PERFORMANCE DATA

Routine	Max Cycles	Min Cycles	Program Memory	Data Memory
SQRT24	2968	7	197	6
EXP24	2600	1990	349	6
EXP1024	2561	2043	355	6
LOG24	3555	1662	261	10
LOG1024	3567	1674	259	10
SIN24	4494	2564	368	11
COS24	4505	2736	368	11
SINCOS24	6478	4525	397	15
FLOOR24	55	37	107	8
TALTB24	28	9	48	6
TALEB24	26	9	44	6
TAGTB24	28	9	48	6
TAGEB24	26	9	44	6
TAEQB24	14	5	13	6
TANEB24	14	5	13	6
RND3224	26	3	25	5
SQRT32	4966	7	142	10
EXP32	5411	16	401	14
EXP1032	5384	3515	401	14
LOG32	5406	4797	297	14
LOG1032	5949	5208	16	1
SIN32	6121	4030	474	11
COS32	6098	3568	474	11
SINCOS32	8858	6611	503	15
FLOOR32	61	41	159	10
RAND32	487	487	37	4
TALTB32	34	9	60	8
TALEB32	32	9	56	8
TAGTB32	34	9	60	8
TAGEB32	32	9	56	8
TAEQB32	18	5	17	8
TANEB32	18	5	17	8
RND4032	29	3	28	8

**Note:** Program and data memory values do not include dependency requirements.

#### **REFERENCES**

- 1. Cavanagh, J.J.F., "Digital Computer Arithmetic," McGraw-Hill, 1984.
- 2. Hwang, K., "Computer Arithmetic," John Wiley & Sons, 1979.
- 3. Scott, N.R., "Computer Number Systems & Arithmetic," Prentice Hall, 1985.
- 4. Knuth, D.E., "The Art of Computer Programming, Volume 2," Addison-Wesley, 1981.
- F.J.Testa, "IEEE 754 Compliant Floating Point Routines," AN575, Embedded Control Handbook, Microchip Technology Inc., 1995.

TABLE A-3: PIC17CXXX ELEMENTARY FUNCTION DEPENDENCIES

Routine	Dependencies						
SQRT24	FPA32	FPD32	FXM1616U	RND3224			
EXP24	FPX32	FXM2416U	FLOOR24	INT2416	RND3224		
EXP1024	FPX32	FXM2416U	FLOOR24	INT2416	RND3224		
LOG24	FPX32	FLO1624	FXM2424U	RND3224			
LOG1024	FPX32	FLO1624	FXM2424U	RND3224			
SIN24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224	
COS24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224	
SINCOS24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224	
POW24	LOG32	EXP32	RND3224				
SQRT32	FPA32	FPD32	FXM2424U	RND4032			
EXP32	FPX32	FXM3224U	FLOOR32	FXM2416U	FXM2424U	INT2416	RND4032
EXP1032	FPX32	FXM3224U	FLOOR32	FXM2416U	FXM2424U	INT2416	RND4032
LOG32	FPX32	FLO1624	FXM2424U	FXM2416U	RND4032		
LOG1032	LOG32	FXM3232U	RND4032				
SIN32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	FXM3232U	RND4032
COS32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	FXM3232U	RND4032
SINCOS32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	FXM3232U	RND4032
POW32	FPX32	TAXXB32	FLO1624	INT3224	FLOOR32	RND4032	
RAND32	FXM3224U						

TABLE A-4: PIC16CXXX ELEMENTARY FUNCTION DEPENDENCIES

Routine	Dependencies					
SQRT24	FPX32	FXM2424U	RND3224			
EXP24	FPX32	FXM2416U	FLOOR24	INT2416	RND3224	
EXP1024	FPX32	FXM2416U	FLOOR24	INT2416	RND3224	
LOG24	FPX32	FLO1624	FXM2424U	RND3224		
LOG1024	FPX32	FLO1624	FXM2424U	RND3224		
SIN24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224
COS24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224
SINCOS24	FPX32	FXM2416U	INT3224	FLO2432	FXM2424U	RND3224
SQRT32	FPX32	FXM3232U	RND4032			
EXP32	FPX32	FXM3224U	FLOOR32	INT2416	RND4032	
EXP1032	FPX32	FXM3224U	FLOOR32	INT2416	RND4032	
LOG32	FPX32	FLO1624	FXM2424U	FXM2416U	RND4032	
LOG1032	LOG32					
SIN32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	RND4032
COS32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	RND4032
SINCOS32	FPX32	FXM3224U	INT3224	FLO2432	FXM2416U	RND4032
RAND32	FXM3224U					



**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
   IFDEF
           <u>__</u>16C63
          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
   TROTE
          __16C72
P16CXX
          SET TRUE
                           ; If P16C72, use INHX8M file format.
P16_MAP2
          SET
                TRUE
   ENDIF
  IFDEF
           16C73
P16CXX
          SET TRUE
                           ; If P16C73, use INHX8M file format.
P16_MAP2
          SET TRUE
  ENDIF
  IFDEF
          __16C73A
P16CXX
          SET TRUE
                           ; If P16C73A, use INHX8M file format.
P16_MAP2
          SET
                 TRUE
   ENDIF
  IFDEF
           __16C74
P16CXX
          SET TRUE
                           ; If P16C74, use INHX8M file format.
P16_MAP2
          SET TRUE
   ENDIF
           _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.
           SET TRUE
P16CXX
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
FIDUDX SET TRUE ; If P16C54A, use INHX8M file format.

RESET_V SET 0x01FF ; Reset Vector of cold for all cold format.
  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
                               STATUS, 2
              _{\rm Z}
       general register variables
   IF ( P16_MAP1 )
ACCB7
               equ
                      0x0C
                      0x0D
ACCB6
               equ
ACCB5
                      0x0E
               equ
ACCB4
               equ
                      0x0F
ACCB3
               equ
                      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
                      0x10
              eau
AARGB2
              equ
                      0x11
AARGB1
              equ
                      0x12
AARGRO
                      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
BEXP
                    0x1B
                         ; 8 bit biased exponent for argument B
;
       floating point library exception flags
;
;
FPFLAGS
             equ
                    0x16
                           ; floating point library exception flags
IOV
             equ
                    0
                           ; bit0 = integer overflow flag
                           ; 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
                           ; bit6 = floating point rounding flag, 0 = truncation
RND
             equ
                           ; 1 = unbiased rounding to nearest LSB
SAT
                           ; bit7 = floating point saturate flag, 0 = terminate on
             equ
                            ; exception without saturation, 1 = terminate on
                            ; exception with saturation to appropriate value
   ENDIF
;
```

```
IF ( P16_MAP2 )
ACCB7
                     0x20
              equ
ACCB6
              equ
                     0x21
ACCB5
              equ
                     0x22
                     0 \times 23
ACCB4
              equ
ACCB3
                     0x24
              equ
ACCB2
              equ
                     0x25
ACCB1
                     0x26
              equ
                     0 \times 27
ACCB0
              equ
ACC
                     0x27
                            ; most significant byte of contiguous 8 byte accumulator
              equ
SIGN
                     0x29
                            ; save location for sign in MSB
                     0 \times 30
TEMPB3
              equ
TEMPB2
                     0x31
              equ
TEMPB1
              equ
                     0x32
TEMPB0
              equ
                     0x33
TEMP
                     0x33
              equ
                            ; temporary storage
       binary operation arguments
AARGB7
              equ
                     0x20
                     0x21
AARGB6
              equ
                     0x22
AARGB5
              equ
AARGB4
              equ
                     0x23
AARGB3
                     0x24
              equ
                     0x25
AARGB2
              equ
AARGB1
              equ
                     0x26
AARGB0
                     0x27
              equ
AARG
                     0x27
                            ; most significant byte of argument A
              equ
                     0x2B
BARGB3
              equ
                     0x2C
BARGB2
              equ
BARGB1
                     0x2D
              equ
BARGB0
                     0x2E
              equ
BARG
                     0x2E
                            ; most significant byte of argument B
              equ
       Note that AARG and ACC reference the same storage location
FIXED POINT SPECIFIC DEFINITIONS
       remainder storage
REMB3
              equ
                     0x20
                     0 \times 21
REMB2
              equ
REMB1
                     0x22
              equ
REMB0
              equ
                     0x23
                            ; most significant byte of remainder
LOOPCOUNT
                     0x34
                            ; loop counter
              equ
FLOATING POINT SPECIFIC DEFINITIONS
       literal constants
EXPBIAS
              equ
                     D'127'
;
;
       biased exponents
                     0x28
                            ; 8 bit biased exponent
EXP
              equ
AEXP
              equ
                     0x28
                            ; 8 bit biased exponent for argument A
BEXP
                     0x2F
                            ; 8 bit biased exponent for argument B
```

```
floating point library exception flags
;
                     0x2A
                            ; floating point library exception flags
FPFLAGS
              equ
                            ; bit0 = integer overflow flag
IOV
              equ
FOV
                            ; bit1 = floating point overflow flag
              eau
                     1
                            ; bit2 = floating point underflow flag
FUN
              equ
FDZ
              equ
                           ; bit3 = floating point divide by zero flag
NAN
              equ
                           ; bit4 = not-a-number exception flag
DOM
                     5
                            ; bit5 = domain error exception flag
              equ
RND
                     6
                            ; bit6 = floating point rounding flag, 0 = truncation
              equ
                            ; 1 = unbiased rounding to nearest LSb
                     7
SAT
              equ
                            ; bit7 = floating point saturate flag, 0 = terminate on
                             ; exception without saturation, 1 = terminate on
                             ; exception with saturation to appropriate value
ELEMENTARY FUNCTION MEMORY
CEXP
                        0x35
              eau
CARGB0
              equ
                        0x36
CARGB1
              equ
                        0x37
CARGB2
                        0x38
              equ
CARGB3
                        0x39
              equ
DEXP
              equ
                        0x3A
                        0x3B
DARGRO
              equ
DARGB1
              equ
                        0x3C
DARGB2
                        0x3D
              equ
DARGB3
                        0x3E
              equ
EEXP
                        0x3F
              equ
                        0x40
EARGRO.
              equ
EARGB1
              equ
                        0x41
EARGB2
                        0x42
              equ
EARGB3
              equ
                        0x43
ZARGB0
              equ
                        0x44
ZARGB1
                        0x45
              equ
ZARGB2
              equ
                        0x46
ZARGB3
              equ
                        0x47
RANDB0
                        0x48
              equ
RANDB1
                        0x49
              equ
RANDB2
              equ
                        0x4A
RANDB3
                        0x4B
              equ
24-BIT FLOATING POINT CONSTANTS
       Machine precision
                        0x6F
MACHEP24EXP
              equ
                                          ; 1.52587890625e-5 = 2**-16
MACHEP24B0
                        0x00
              equ
MACHEP24B1
                        0 \times 00
              equ
        Maximum argument to EXP24
MAXLOG24EXP
              equ
                        0x85
                                          ; 88.7228391117 = log(2**128)
MAXLOG24B0
                        0x31
              equ
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
MAXLOG1024EXP
                                            ; 38.531839445 = log10(2**128)
               equ
                          0x84
MAXLOG1024B0
                          0x1A
               equ
MAXLOG1024B1
                          0x21
               ean
        Minimum argument to EXP1024
                          0x84
                                             ; -37.9297794537 = log10(2**-126)
MINLOG1024EXP
               equ
MINLOG1024B0
                          0x97
               eau
MINLOG1024B1
               equ
                          0xB8
        Maximum representable number before overflow
                          0xFF
                                             ; 6.80554349248E38 = (2**128) * (2 - 2**-15)
MAXNUM24EXP
               eau
MAXNUM24B0
               equ
                          0x7F
MAXNUM24B1
               equ
                          0xFF
        Minimum representable number before underflow
MINNUM24EXP
                          0 \times 01
                                             ; 1.17549435082E-38 = (2**-126) * 1
               equ
MTNNIIM24B0
                          0x00
               equ
                          0x00
MINNUM24B1
               equ
        Loss threshold for argument to SIN24 and COS24
                          0x8B
LOSSTHR24EXP
                                             ; 4096 = sqrt(2**24)
               equ
LOSSTHR24B0
                          0x00
               eau
LOSSTHR24B1
                          0x00
               equ
32-BIT FLOATING POINT CONSTANTS
        Machine precision
MACHEP32EXP
                          0x67
                                              ; 5.96046447754E-8 = 2**-24
               equ
MACHEP32B0
                          0x00
               equ
MACHEP32B1
                          0x00
               equ
                          0 \times 00
MACHEP32B2
               equ
       Maximum argument to EXP32
MAXLOG32EXP
               equ
                          0x85
                                              ; 88.7228391117 = log(2**128)
MAXLOG32B0
               equ
                          0x31
MAXIOG32B1
                          0x72
               equ
MAXLOG32B2
                          0x18
               equ
        Minimum argument to EXP32
MINIOG32EXP
                          0 \times 85
                                             i - 87.3365447506 = log(2**-126)
               eau
MINLOG32B0
                          0xAE
               equ
MINLOG32B1
                          0xAC
               equ
MINLOG32B2
                          0x50
               equ
        Maximum argument to EXP1032
MAXLOG1032EXP
                          0x84
                                              ; 38.531839445 = log10(2**128)
               equ
MAXLOG1032B0
               equ
                          0x1A
MAXIOG1032B1
                          0 \times 20
               equ
MAXLOG1032B2
                          0x9B
               equ
```

```
Minimum argument to EXP1032
MINLOG1032EXP
                           0x84
                                                ; -37.9297794537 = log10(2**-126)
                equ
                           0x97
MINLOG1032B0
                equ
MINLOG1032B1
                           0xB8
                equ
MINLOG1032B2
                equ
                           0x18
         Maximum representable number before overflow
MAXNUM32EXP
                           0xFF
                                                ; 6.80564774407E38 = (2**128) * (2 - 2**-23)
                equ
MAXNUM32B0
                           0x7F
                equ
MAXNUM32B1
                equ
                           0xFF
MAXNUM32B2
                equ
                           0xFF
         Minimum representable number before underflow
MINNUM32EXP
                           0x01
                                               ; 1.17549435082E-38 = (2**-126) * 1
                equ
                           0x00
MINNUM32B0
                equ
MINNUM32B1
                           0x00
                equ
MINNUM32B2
                           0x00
                equ
         Loss threshold for argument to SIN32 and COS32
LOSSTHR32EXP
                           0x8B
                                                ; 4096 = sqrt(2**24)
                equ
LOSSTHR32B0
                           0x00
                equ
LOSSTHR32B1
                equ
                           0x00
LOSSTHR32B2
                           0 \times 00
                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
                        2
В2
                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
                      0x1C
              eau
AARGB2
              equ
                      0x1D
AARGB1
              equ
                      0x1E
                      0x1F
AARGRO
              equ
AARG
                      0x1F
              equ
                             ; most significant byte of argument A
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
                             ; 8 bit biased exponent for argument A
              eau
                      0x20
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
                      4
              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
```

```
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 ELEMENTARY FUNCTION LIBRARY

```
RCS Header $Id: math16.mac 1.3 1996/10/05 19:52:32 F.J.Testa Exp $
       $Revision: 1.3 $
polynomial evaluation macros
POLL124 macro
                       COF, N, ROUND
       32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
;
       with leading coefficient of one, and where AARG is assumed have been be saved
;
;
       in DARG when N>1. The result is in AARG.
;
       ROUND = Ono rounding is enabled; can be previously enabled
       ROUND = 1rounding is enabled
;
       ROUND = 2rounding is enabled then disabled before last add
;
       ROUND = 3rounding is assumed disabled then enabled before last add
       ROUND = 4rounding is assumed enabled and then disabled before last
               add if DARGB3,RND is clear
       {\tt ROUND} = 5rounding is assumed disabled and then enabled before last
               add if DARGB3,RND is set
       local
                      i,j
       variablei = N, j = 0
       variablei = i - 1
               ROUND == 1 || ROUND == 2
       if
               BSF
                              FPFLAGS, RND
       endif
               M.TVOM
                              COF#v(i)
               MOVWF
                              BEXP
       variable j = 0
       while
                       j <= 2
               MOVLW
                              COF#v(i)#v(j)
               MOVWF
                               BARGB#v(j)
       variable j = j + 1
       endw
               CALL
                             FPA32
       variablei = i - 1
                      i >= 0
       while
               MOVF
                               DEXP,W
               MOVWF
                               BEXP
               MOVF
                               DARGB0,W
               MOVWF
                               BARGB0
               MOVF
                               DARGB1,W
               MOVWF
                               BARGB1
```

```
MOVF
                          DARGB2,W
        MOVWF
                          BARGB2
        CALL
                          FPM32
        MOVLW
                          COF#v(i)
        MOVWF
                          BEXP
variable j = 0
while
                 j <= 2
        MOVLW
                          COF#v(i)#v(j)
        MOVWF
                          BARGB#v(j)
variable j = j + 1
endw
        i == 0
if
        if
                 ROUND == 2
        BCF
                         FPFLAGS, RND
        endif
        if
                 ROUND == 3
        BSF
                         FPFLAGS, RND
        endif
        if
                 ROUND == 4
        BTFSS
                          DARGB3, RND
                          FPFLAGS, RND
        BCF
        endif
                 ROUND == 5
        if
        BTFSC
                          DARGB3,RND
        BSF
                          FPFLAGS, RND
        endif
endif
        CALL
                          FPA32
variablei = i - 1
endw
endm
                 COF, N, ROUND
macro
32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
and where AARG is assumed have been be saved in DARG when N>1.
The result is in AARG.
ROUND = Ono rounding is enabled; can be previously enabled
ROUND = 1rounding is enabled
```

POL24

```
ROUND = 2rounding is enabled then disabled before last add
;
        ROUND = 3rounding is assumed disabled then enabled before last add
        ROUND = 4rounding is assumed enabled and then disabled before last
                add if DARGB3,RND is clear
;
        ROUND = 5rounding is assumed disabled and then enabled before last
;
                add if DARGB3,RND is set
        local
                        i,j
        variablei = N, j = 0
        if
                ROUND == 1 || ROUND == 2
                BSF
                                 FPFLAGS, RND
        endif
                MOVLW
                                COF#v(i)
                MOVWF
                                 BEXP
                        j <= 2
        while
                MOVLW
                                 COF#v(i)#v(j)
                MOVWF
                                 BARGB#v(j)
        variablej = j + 1
        endw
                CALL
                               FPM32
        variablei = i - 1
                MOVLW
                                 COF#v(i)
                MOVWF
                                 BEXP
        variable j = 0
                        j <= 2
        while
                MOVLW
                                 COF#v(i)#v(j)
                MOVWF
                                 BARGB#v(j)
        variable j = j + 1
        endw
                CALL
                               FPA32
        variablei = i - 1
        while
                       i >= 0
                MOVF
                                 DEXP,W
                MOVWF
                                 BEXP
                MOVF
                                 DARGB0,W
                MOVWF
                                 BARGB0
                                 DARGB1,W
                MOVF
                MOVWF
                                 BARGB1
                MOVF
                                 DARGB2,W
                MOVWF
                                 BARGB2
                                 FPM32
                CALL
                MOVLW
                                 COF#v(i)
                MOVWF
                                 BEXP
```

```
variable j = 0
        while
                         j <= 2
                 MOVLW
                                  COF#v(i)#v(j)
                 MOVWF
                                  BARGB#v(j)
        variable j = j + 1
        endw
        if
                 i == 0
                         ROUND == 2
                 if
                                  FPFLAGS, RND
                 endif
                 if
                         ROUND == 3
                 BSF
                                  FPFLAGS, RND
                 endif
                 if
                         ROUND == 4
                 BTFSS
                                  DARGB3, RND
                 BCF
                                  FPFLAGS, RND
                 endif
                         ROUND == 5
                 if
                 BTFSC
                                  DARGB3, RND
                 BSF
                                  FPFLAGS, RND
                 endif
        endif
                 CALL
                                 FPA32
        variable i = i - 1
        endw
        endm
POLL132 macro
                         COF, N, ROUND
        32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
        with leading coefficient of one, and where AARG is assumed have been be saved
        in DARG when N>1. The result is in AARG.
        ROUND = Ono rounding is enabled; can be previously enabled
        ROUND = 1rounding is enabled
        ROUND = 2rounding is enabled then disabled before last add
        ROUND = 3rounding is assumed disabled then enabled before last add
        ROUND = 4rounding is assumed enabled and then disabled before last
                 add if DARGB3,RND is clear
        ROUND = 5rounding is assumed disabled and then enabled before last
                 add if DARGB3,RND is set
```

```
local
local i,j variable i = N, j = 0
variablei = i - 1
if
        ROUND == 1 || ROUND == 2
                         FPFLAGS, RND
        BSF
endif
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variable j = 0
while
                 j <= 2
        MOVLW
                         COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variable j = j + 1
endw
        CALL
                        FPA32
variablei = i - 1
while
               i >= 0
        MOVF
                         DEXP,W
        MOVWF
                         BEXP
        MOVF
                         DARGB0,W
        MOVWF
                         BARGB0
        MOVF
                         DARGB1,W
        MOVWF
                         BARGB1
                         DARGB2,W
        MOVF
        MOVWF
                         BARGB2
        CALL
                         FPM32
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variable j = 0
                 j <= 2
while
        MOVLW
                         COF#v(i)#v(j)
                         BARGB#v(j)
        MOVWF
variablej = j + 1
endw
        i == 0
if
        if
                ROUND == 2
        BCF
                       FPFLAGS, RND
        endif
        if
                 ROUND == 3
```

```
BSF
                                  FPFLAGS, RND
                 endif
                         ROUND == 4
                 if
                 BTFSS
                                  DARGB3, RND
                 BCF
                                  FPFLAGS, RND
                 endif
                         ROUND == 5
                 BTFSC
                                  DARGB3, RND
                 BSF
                                  FPFLAGS, RND
                 endif
        endif
                                  FPA32
                 CALL
        variablei = i - 1
        endw
        endm
POL32
                         COF, N, ROUND
        macro
        32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
        and where AARG is assumed have been be saved in DARG when N>1.
        The result is in AARG.
        ROUND = Ono rounding is enabled; can be previously enabled
        ROUND = 1rounding is enabled
        ROUND = 2rounding is enabled then disabled before last add
        ROUND = 3rounding is assumed disabled then enabled before last add
        ROUND = 4rounding is assumed enabled and then disabled before last
                add if DARGB3,RND is clear
        ROUND = 5rounding is assumed disabled and then enabled before last
                add if DARGB3,RND is set
        ROUND = 6rounding is performed by RND4032 and then disabled before last add
        local
        variablei = N, j = 0
                ROUND == 1 || ROUND == 2
                 BSF
                                  FPFLAGS, RND
        endif
                 MOVLW
                                  COF#v(i)
                 MOVWF
                                  BEXP
        while
                         j <= 2
                 MOVLW
                                  COF#v(i)#v(j)
                MOVWF
                                  BARGB#v(j)
        variable j = j + 1
        endw
```

```
CALL
                         FPM32
if
        ROUND == 6
        CALL
                         RND4032
endif
variablei = i - 1
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variablej = 0
while
                 j <= 2
        MOVLW
                         COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variablej = j + 1
{\tt endw}
        CALL
                         FPA32
if
        ROUND == 6
        CALL
                         RND4032
endif
variablei = i - 1
while
                 i >= 0
        MOVF
                         DEXP,W
        MOVWF
                         BEXP
        MOVF
                         DARGB0,W
        MOVWF
                         BARGB0
                         DARGB1,W
        MOVF
        MOVWF
                         BARGB1
        MOVF
                         DARGB2,W
        MOVWF
                         BARGB2
        CALL
                         FPM32
if
        ROUND == 6
        CALL
                         RND4032
endif
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variable j = 0
while
                 j <= 2
        MOVLW
                         COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variablej = j + 1
endw
```

```
i == 0
if
        if
                ROUND == 2
        BCF
                       FPFLAGS, RND
        endif
        if
                ROUND == 3
        BSF
                       FPFLAGS, RND
        endif
        if
                ROUND == 4
        BTFSS
                        DARGB3,RND
                        FPFLAGS, RND
        BCF
        endif
                ROUND == 5
        if
        BTFSC
                        DARGB3,RND
                         FPFLAGS, RND
        BSF
        endif
endif
        CALL
                       FPA32
if
        ROUND == 6 && i != 0
                        RND4032
        CALL
endif
variablei = i - 1
endw
endm
```

```
RCS Header $Id: exp24.a16 1.6 1997/02/25 14:23:30 F.J.Testa Exp $
        $Revision: 1.6 $
        Evaluate expl0(x)
                24 bit floating point number in AEXP, AARGBO, AARGB1
                         EXP1024
        Use:
                CALL
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- EXP10( AARG )
        Testing on [MINLOG10, MAXLOG10] from 10000 trials:
                min
                         max
                                 mean
        Timing: 2043
                         2561
                                 2328.7 clks
;
                min
                                 mean
                         max
                                          rms
        Error:
                -0x75
                         0x77
                                 -0.95
                                          40.34
                                                  nsb
        This approximation of the base 10 exponential function is based upon the
        expansion
                exp10(x) = 10**x = 2**(x/log10(2)) = 2**z * 2**n
;
                         x/\log 10(2) = z + n,
        where 0 <= z < 1 and n is an integer, evaluated during range reduction.
        Segmented third degree minimax polynomial approximations are used to
        estimate 2**z on the intervals [0,.25], [.25,.5], [.5,.75] and [.75,1].
EXP1024
                                                   ; test for |x| < 2**(-24)/(2*LOG(10))
                MOVLW
                                  0x64
                SUBWF
                                 EXP,W
                MOVWF
                                 TEMPB0
                                  TEMPB0,MSB
                BTFSC
                 GOTO
                                 EXP1024ONE
                                                  ; return 10**x = 1
                BTFSC
                                 AARGB0,MSB
                GOTO
                                 TNEXP1024
TPEXP1024
                MOVF
                                 AEXP,W
                SUBLW
                                 MAXLOG1024EXP
                BTFSS
                                  _C
                GOTO
                                 DOMERR24
                                  _Z
                 BTFSS
                GOTO
                                  EXP1024ARGOK
                MOVF
                                 AARGB0,W
                SUBLW
                                 MAXLOG1024B0
                BTFSS
                                  _C
                GOTO
                                 DOMERR24
                BTFSS
                                  _{\rm Z}
                GOTO
                                 EXP1024ARGOK
                MOVF
                                 AARGB1,W
                                 MAXLOG1024B1
                 SUBLW
                BTFSS
                                  C
                                 DOMERR24
                 GOTO
                 GOTO
                                  EXP1024ARGOK
TNEXP1024
```

	MOVF	AEXP,W	
	SUBLW	MINLOG1024EXP	
	BTFSS	_C	
	GOTO	DOMERR24	
	BTFSS	_Z	
	GOTO	EXP1024ARGOK	
	MOVF	AARGB0,W	
	SUBLW	MINLOG1024B0	
	BTFSS	_C	
	GOTO	DOMERR24	
	BTFSS	_Z	
	GOTO	EXP1024ARGOK	
	MOVF	AARGB1,W	
	SUBLW	MINLOG1024B1	
	BTFSS	_C	
	GOTO	DOMERR24	
EXP1024ARGOK			
	MOVF	FPFLAGS,W	
	MOVWF	DARGB3	; save rounding flag
	BCF	FPFLAGS,RND	; disable rounding
	CALL	RREXP1024	
	MONT W	07E	
	MOVLW	0x7E	
	SUBWF	AEXP,W	
	BTFSS	_Z	
	GOTO	EXP1024L	
EXP1024H	BTFSS	AARGB0,MSB-1	
	GOTO	EXP1024HL	
	0010	DMI 102 IIID	
	POL24	EXP24HH,3,0	; minimax approximation on [.75,1]
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	RETLW	0x00	
	10111	01100	
EXP1024HL	POL24	EXP24HL,3,0	; minimax approximation on [.5,.75]
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	RETLW	0x00	
	KEILW	0200	
EXP1024L	MOVLW	0x7D	
	SUBWF	AEXP,W	
	BTFSS	_Z	
	GOTO	EXP1024LL	
	POL24	EXP24LH,3,0	; minimax approximation on [.25,.5]
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
		0x00	
	RETLW	0200	
EXP1024LL	POL24	EXP24LL,3,0	; minimax approximation on [0,.25]
EXP1024OK			
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	BTFSS	DARGB3,RND	
	RETLW	$0 \times 0 0$	
	BSF	FPFLAGS,RND	; restore rounding flag

```
GOTO
                                   RND3224
EXP1024ONE
                 MOVLW
                                   EXPBIAS
                                                     ; return e^*x = 1.0
                 MOVWF
                                   AEXP
                                   AARGB0
                  CLRF
                 CLRF
                                   AARGB1
                  CLRF
                                   AARGB2
                  RETLW
                                   0x00
DOMERR24
                  BSF
                                   FPFLAGS,DOM
                                                     ; domain error
                 RETLW
                                   0xFF
         Range reduction routine for the exponential function
                 x/\log 10(2) = z + n
RREXP1024
                                   AARGB0,W
                 MOVE
                                   DARGB0
                 MOVWF
                  BSF
                                   AARGB0,MSB
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   BARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   BARGB1
                                                     i = 1/\log(10) = 3.32192809489
                 MOVLW
                                   0xD4
                                   AARGB0
                 MOVWF
                 MOVLW
                                   0x9A
                 MOVWF
                                   AARGB1
                                   0x78
                 MOVLW
                 MOVWF
                                   AARGB2
                  CALL
                                   FXM2416U
                                                     ; x * (1/log10(2))
                                   AEXP,F
                  INCF
                  INCF
                                   AEXP,F
                  BTFSC
                                   AARGB0, MSB
                                   RREXP1024YOK
                 GOTO
                                   AARGB3,F
                 RLF
                 RLF
                                   AARGB2,F
                                   AARGB1,F
                 RLF
                 RLF
                                   AARGB0,F
                 DECF
                                   AEXP,F
RREXP1024YOK
                  BTFSS
                                   DARGB0, MSB
                 BCF
                                   AARGB0, MSB
                 MOVF
                                   AEXP,W
                                   BEXP
                                                     ; save y in BARG
                 MOVWF
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   BARGB0
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   BARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   BARGB2
                  CALL
                                   FLOOR24
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                                     ; save k in DARG
                                   AARGB0,W
                 MOVF
```

```
DARGB0
                 MOVWF
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   DARGB1
                                                     ; k = [x * (1/ln2)]
                 CALL
                                   INT2416
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   EARGB3
                                                     ; save k in EARG
                                   DEXP,W
                 MOVF
                 MOVWF
                                   AEXP
                                   DARGB0,W
                 MOVF
                 MOVWF
                                   AARGB0
                 MOVF
                                   DARGB1,W
                                   AARGB1
                 MOVWF
                 CLRF
                                   AARGB2
                 MOVLW
                                   0x80
                                   AARGB0,F
                 XORWF
                 CALL
                                   FPA32
                 MOVF
                                   AEXP,W
                                   DEXP
                                                     ; save y in DARG
                 MOVWF
                                   AARGB0,W
                 MOVF
                                   DARGB0
                 MOVWF
                 MOVF
                                   AARGB1,W
                                   DARGB1
                 MOVWF
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   DARGB2
                 RETLW
                                   0x00
        third degree minimax polynomial coefficients for 2**(x) on [.75,1]
EXP24HH0
                                   0x7E
                                                     ; EXP24HH0 = .99103284632
                 EQU
EXP24HH00
                 EQU
                                   0x7D
EXP24HH01
                 EQU
                                   0xB4
EXP24HH02
                 EQU
                                   0x54
                                                     ; EXP24HH1 = .73346850266
EXP24HH1
                 EQU
                                   0x7E
EXP24HH10
                 EQU
                                   0x3B
EXP24HH11
                 EQU
                                   0xC4
EXP24HH12
                                   0x97
                 EQU
EXP24HH2
                 EQU
                                   0x7C
                                                     ; EXP24HH2 = .17374128273
EXP24HH20
                                   0x31
                 EQU
EXP24HH21
                 EQU
                                   0xE9
EXP24HH22
                 EQU
                                   0x3C
                                   0x7B
                                                     ; EXP24HH3 = .10175678143
EXP24HH3
                 EQU
EXP24HH30
                                   0x50
                 EQU
EXP24HH31
                 EQU
                                   0x65
EXP24HH32
                 EQU
                                   0xDC
        third degree minimax polynomial coefficients for 2**(x) on [.5,.75]
EXP24HL0
                                   0x7E
                                                     ; EXP24HL0 = .99801686089
                 EQU
EXP24HL00
                 EQU
                                   0x7F
                                   0x7E
EXP24HL01
                 EQU
EXP24HL02
                 EQU
                                   0x08
EXP24HL1
                 EQU
                                   0x7E
                                                     ; EXP24HL1 = .70586404164
EXP24HL10
                 EQU
                                   0x34
EXP24HL11
                 EQU
                                   0xB3
```

```
EXP24HL12
                EQU
                                0x81
EXP24HL2
                EQU
                                 0x7C
                                                 ; EXP24HL2 = .21027360637
EXP24HL20
                EQU
                                0x57
                                 0x51
EXP24HL21
                EQU
EXP24HL22
                EQU
                                0xF7
EXP24HL3
                EQU
                                0x7B
                                                 ; EXP24HL3 = .85566912730E-1
EXP24HL30
                EQU
                                0x2F
EXP24HL31
                EQU
                                0x3D
EXP24HL32
                EOU
                                0xB5
        third degree minimax polynomial coefficients for 2**(x) on [.25,.5]
                                0x7E
                                                 ; EXP24LH0 = .99979384559
EXP24LH0
                EQU
EXP24LH00
                EQU
                                0x7F
EXP24LH01
                EQU
                                0xF2
EXP24LH02
                EQU
                                0x7D
                                                 ; EXP24LH1 = .69545887384
EXP24LH1
                EQU
                                0x7E
EXP24LH10
                EQU
                                0x32
EXP24LH11
                EQU
                                0x09
EXP24LH12
                EQU
                                0x98
EXP24LH2
                EQU
                                0x7C
                                                 ; EXP24LH2 = .23078300446
EXP24LH20
                                0x6C
                EQU
EXP24LH21
                EQU
                                0x52
EXP24LH22
                                0x61
                EQU
EXP24LH3
                EQU
                                0x7B
                                                 ; EXP24LH3 = .71952910179E-1
EXP24LH30
                EQU
                                 0x13
EXP24LH31
                EQU
                                 0x5C
EXP24TH32
                EQU
                                0x0C
        third degree minimax polynomial coefficients for 2**(x) on [0,.25]
EXP24LL0
                EQU
                                0x7E
                                                 ; EXP24LL0 = .99999970657
EXP24LL00
                EQU
                                0x7F
EXP24LL01
                EQU
                                0xFF
EXP24LL02
                                 0xFB
                EQU
                                0x7E
                                                 ; EXP24LL1 = .69318585159
EXP24LL1
                EQU
EXP241.1.10
                EQU
                                0 \times 31
EXP24LL11
                EOU
                                0x74
EXP24LL12
                                0xA1
                EQU
EXP24LL2
                EQU
                                0x7C
                                                 ; EXP24LL2 = .23944330933
EXP241.1.20
                                0x75
                EQU
EXP24LL21
                EQU
                                 0x30
EXP24LL22
                EQU
                                 0xA0
EXP24T.T.3
                                0x7A
                                                 ; EXP24LL3 = .60504944237E-1
                EOU
EXP24LL30
                                 0x77
                EQU
EXP24LL31
                EQU
                                 0xD4
EXP24LL32
                EQU
                                0x08
Evaluate exp(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Use:
                CALL
                        EXP24
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
```

```
Result: AARG <-- EXP( AARG )
        Testing on [MINLOG, MAXLOG] from 100000 trials:
                 min
                          max
                                   mean
        Timing: 1990
                          2600
                                   2347.6
                                           clks
                 min
                          max
                                   mean
                                            rms
                 -0x43
                          0x40
                                   -.77
                                            16.75
        Error:
                                                     nsb
        This approximation of the exponential function is based upon the
        expansion
                 \exp(x) = e^{*x}x = 2^{*x}(x/\log(2)) = 2^{*x}z * 2^{*x}n,
                          x/\log(2) = z + n,
        where 0 <= z < 1 and n is an integer, evaluated during range reduction.
        Segmented third degree minimax polynomial approximations are used to
        estimate 2**z on the intervals [0,.25], [.25,.5], [.5,.75] and [.75,1].
EXP24
                 MOVLW
                                                     ; test for |x| < 2**(-24)/2
                                   0x66
                 SUBWF
                                   EXP,W
                                   TEMPB0
                 MOVWF
                 BTFSC
                                   TEMPB0, MSB
                 GOTO
                                   EXP24ONE
                                                     ; return e^*x = 1
                 BTFSC
                                   AARGB0,MSB
                                                     ; determine sign
                 GOTO
                                   TNEXP24
TPEXP24
                                   AEXP,W
                 MOVF
                                                     ; positive domain check
                 SUBLW
                                   MAXLOG24EXP
                 BTFSS
                                   _C
                                   DOMERR24
                 GOTO
                 BTFSS
                                   _{\rm Z}
                 GOTO
                                   EXP24ARGOK
                                   AARGB0,W
                 MOVF
                                   MAXLOG24B0
                 SUBLW
                 BTFSS
                                   _C
                                   DOMERR24
                 GOTO
                 BTFSS
                 GOTO
                                   EXP24ARGOK
                 MOVF
                                   AARGB1,W
                 SUBLW
                                   MAXLOG24B1
                 BTFSS
                                   _C
                                   DOMERR24
                 GOTO
                 GOTO
                                   EXP24ARGOK
TNEXP24
                 MOVF
                                   AEXP,W
                                                     ; negative domain check
                                   MINLOG24EXP
                 SUBLW
                 BTFSS
                                   _C
                 GOTO
                                   DOMERR24
                 BTFSS
                                   _{\rm Z}
                                   EXP24ARGOK
                 GOTO
                 MOVF
                                   AARGB0,W
                                   MINLOG24B0
                 SUBLW
                 BTFSS
                                   _C
                                   DOMERR24
                 GOTO
```

	BTFSS	_Z	
	GOTO	EXP24ARGOK	
	MOLTE	33DCD1 11	
	MOVF	AARGB1,W MINLOG24B1	
	SUBLW BTFSS	_C	
	GOTO	_C DOMERR24	
EXP24ARGOK			
	MOVF	FPFLAGS,W	
	MOVWF	DARGB3	; save rounding flag
	BCF	FPFLAGS, RND	; disable rounding
	CALL	RREXP24	; range reduction
	CALL	KKEAFZI	, range reduction
	MOVLW	0x7E	
	SUBWF	AEXP,W	
	BTFSS	_Z	
	GOTO	EXP24L	
	D==0.0	33D0D0 M0D 1	
EXP24H	BTFSS GOTO	AARGB0,MSB-1 EXP24HL	
	G010	EXPZINL	
	POL24	EXP24HH,3,0	; minimax approximation on [.75,1]
	GOTO	EXP24OK	
EXP24HL	POL24	EXP24HL,3,0	; minimax approximation on [.5,.75]
	GOTO	EXP24OK	
	0010	2111 2 1011	
EXP24L	MOVLW	0x7D	
	SUBWF	AEXP,W	
	BTFSS	_Z	
	GOTO	EXP24LL	
	POL24	EXP24LH,3,0	; minimax approximation on [.25,.5]
	-	, , , ,	
	GOTO	EXP24OK	
EXP24LL	POL24	EXP24LL,3,0	; minimax approximation on [0,.25]
EXP24OK			
BM 2 1010	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	BTFSS	DARGB3,RND	
	RETLW	0x00	
	BSF	FPFLAGS,RND RND3224	; restore rounding flag
	GOTO	KND3224	
EXP24ONE	MOVLW	EXPBIAS	; return e**x = 1.0
	MOVWF	AEXP	
	CLRF	AARGB0	
	CLRF	AARGB1	
	CLRF	AARGB2	
	RETLW	0x00	
DOMERR24	BSF	FPFLAGS,DOM	; domain error
DOFIBILITZ I	RETLW	0xFF	, domain circi
; *********	******	*******	***********
		£ +1	i-1 for all or
; Range	reduction routine	ior the exponent	clai function
;	x/log(2) = z +	n	
•	,9(2) - 2	<del></del>	

RREXP24			
	MOVF	AARGB0,W	; save sign
	MOVWF	DARGB0	
	BSF	AARGB0,MSB	; make MSB explicit
	MOVF	AARGB0,W	
	MOVWF	BARGB0	
	MOVF	AARGB1,W	
	MOVWF	BARGB1	
	MOVLW	0xB8	; 1/ln(2) = 1.44269504089
	MOVWF	AARGB0	7 1/111(2) 1111203001003
	MOVLW	0xAA	
	MOVWF	AARGB1	
	MOVLW	0x3B	
	MOVWF	AARGB2	
	CALL	FXM2416U	; x * (1/ln2)
	INCF	AEXP,F	
	BTFSC	AARGB0,MSB	
	GOTO	RREXP24YOK	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
RREXP24YOK	BTFSS	DARGBO,MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	MOVF	AEXP,W	
	MOVWF	BEXP	; save z in BARG
	MOVF	AARGB0,W	
	MOVWF	BARGB0	
	MOVF	AARGB1,W	
	MOVWF	BARGB1	
	MOVF	AARGB2,W	
	MOVWF	BARGB2	
	CALL	FLOOR24	
	MOVF	AEXP,W	
	MOVWF	DEXP	; save float(n) in DARG
	MOVF	AARGB0,W	
	MOVWF	DARGB0	
	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	CALL	INT2416	; n = [x * (1/ln2)]
	MOVF	AARGB1,W	
	MOVWF	EARGB3	; save n in EARG
	MOVF	DEXP,W	
	MOVWF	AEXP	
	MOVF	DARGB0,W	
	MOVWF	AARGB0	
	MOVF	DARGB1,W	
	MOVWF	AARGB1	
	CLDE	7 7 DCD 2	

CLRF

AARGB2

```
MOVLW
                                   0x80
                                                     ; toggle sign
                 XORWF
                                   AARGB0,F
                                   FPA32
                 CALL
                 CALL
                                   RND4032
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                                     ; save z in DARG
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   DARGB0
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                                   DARGB2
                 MOVWF
                 RETLW
                                   0x00
        third degree minimax polynomial coefficients for 2**(x) on [.75,1]
EXP24HH0
                 EQU
                                   0x7E
                                                     ; EXP24HH0 = .99103284632
EXP24HH00
                 EQU
                                   0x7D
EXP24HH01
                 EQU
                                   0xB4
EXP24HH02
                                   0x54
                 EQU
EXP24HH1
                 EQU
                                   0x7E
                                                     ; EXP24HH1 = .73346850266
EXP24HH10
                 EQU
                                   0x3B
EXP24HH11
                 EQU
                                   0xC4
EXP24HH12
                 EQU
                                   0x97
                                                     ; EXP24HH2 = .17374128273
EXP24HH2
                 EQU
                                   0x7C
EXP24HH20
                                   0x31
                 EQU
EXP24HH21
                                   0xE9
                 EQU
EXP24HH22
                                   0x3C
                 EQU
EXP24HH3
                 EQU
                                   0x7B
                                                     ; EXP24HH3 = .10175678143
EXP24HH30
                 EQU
                                   0x50
EXP24HH31
                 EQU
                                   0x65
EXP24HH32
                 EQU
                                   0xDC
        third degree minimax polynomial coefficients for 2**(x) on [.5,.75]
EXP24HL0
                 EQU
                                   0x7E
                                                     ; EXP24HL0 = .99801686089
EXP24HL00
                 EQU
                                   0x7F
EXP24HL01
                 EQU
                                   0x7E
EXP24HL02
                                   0x08
                 EQU
EXP24HL1
                 EQU
                                   0x7E
                                                     ; EXP24HL1 = .70586404164
EXP24HL10
                 EQU
                                   0x34
EXP24HL11
                                   0xB3
                 EQU
EXP24HL12
                 EQU
                                   0x81
EXP24HL2
                 EQU
                                   0x7C
                                                     ; EXP24HL2 = .21027360637
EXP24HL20
                 EQU
                                   0x57
EXP24HL21
                                   0x51
                 EQU
EXP24HL22
                                   0xF7
                 EQU
EXP24HL3
                 EQU
                                   0x7B
                                                     ; EXP24HL3 = .85566912730E-1
EXP24HL30
                 EQU
                                   0x2F
EXP24HL31
                 EQU
                                   0x3D
EXP24HL32
                 EQU
                                   0xB5
```

```
third degree minimax polynomial coefficients for 2**(x) on [.25,.5]
EXP24LH0
               EQU
                              0x7E
                                            ; EXP24LH0 = .99979384559
                              0x7F
EXP24LH00
               EQU
                              0xF2
EXP24LH01
               EQU
EXP24LH02
               EQU
                              0x7D
EXP24LH1
               EQU
                              0x7E
                                             ; EXP24LH1 = .69545887384
EXP24LH10
               EQU
                              0x32
EXP24LH11
                              0x09
              EQU
EXP24LH12
              EQU
                              0x98
                              0x7C
                                            ; EXP24LH2 = .23078300446
EXP24LH2
               EQU
EXP24LH20
               EQU
                              0x6C
                              0x52
EXP24LH21
              EQU
EXP24LH22
              EQU
                              0x61
EXP24LH3
               EQU
                              0x7B
                                            ; EXP24LH3 = .71952910179E-1
EXP24LH30
                              0x13
              EQU
EXP24LH31
                              0x5C
               EQU
EXP24LH32
               EQU
                              0x0C
       third degree minimax polynomial coefficients for 2**(x) on [0,.25]
EXP24LL0
              EQU
                              0x7E
                                            ; EXP24LL0 = .99999970657
EXP24LL00
                              0x7F
              EQU
EXP24LL01
              EQU
                              0xFF
EXP24LL02
                              0xFB
              EQU
                              0x7E
EXP24LL1
              EQU
                                            ; EXP24LL1 = .69318585159
EXP24LL10
               EQU
                              0x31
EXP24LL11
               EQU
                              0x74
EXP24LL12
              EQU
                              0xA1
EXP24LL2
                              0x7C
                                            ; EXP24LL2 = .23944330933
              EOU
EXP24LL20
              EQU
                              0x75
EXP24LL21
               EQU
                              0x30
EXP24LL22
              EQU
                              0xA0
EXP24LL3
               EQU
                              0x7A
                                            ; EXP24LL3 = .60504944237E-1
EXP24LL30
               EQU
                              0x77
EXP24LL31
               EQU
                              0xD4
EXP241.1.32
               EQU
                              0x08
Evaluate floor(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                      FLOOR24
       Use:
              CALL
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- FLOOR( AARG )
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
               min
                              mean
                      max
       Timing: 37
                      55
                              42.7
                                     clks
               min
                              mean
                      max
                                     rms
       Error: 0x00
                      0x00
                              0.0
                                     0.0
                                            nsb
```

```
floor(x) evaluates the largest integer, as a float, not greater than x.
FLOOR24
                  CLRF
                                   AARGB2
                                                     ; test for zero argument
                 MOVF
                                   AEXP,W
                 BTFSC
                                   _Z
                 RETLW
                                   0x00
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   AARGB3
                                                     ; save mantissa
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   AARGB4
                                   EXPBIAS
                 MOVLW
                                                     ; computed unbiased exponent
                 SUBWF
                                   AEXP,W
                 MOVWF
                                   TEMPB1
                 BTFSC
                                   TEMPB1,MSB
                                   FLOOR24ZERO
                 GOTO
                                   0x10-1
                  SUBLW
                  MOVWF
                                   TEMPB0
                                                     ; save number of zero bits in TEMPBO
                 MOVWF
                                   TEMPB1
                 BTFSC
                                   TEMPB1, LSB+3
                                                     ; divide by eight
                  GOTO
                                   FLOOR24MASKH
FLOOR24MASKL
                 MOVLW
                                   0 \times 0.7
                                                     ; get remainder for mask pointer
                 ANDWF
                                   TEMPB0,F
                 MOVLW
                                   LOW FLOOR24MASKTABLE
                 ADDWF
                                   TEMPB0,F
                                   HIGH FLOOR24MASKTABLE
                 MOVLW
                 BTFSC
                                   C
                                   0x01
                 ADDLW
                 MOVWF
                                   PCLATH
                  INCF
                                   TEMPB0,W
                  CALL
                                   FLOOR24MASKTABLE ; access table for mask
                  ANDWF
                                   AARGB1,F
                                                     ; if negative, round down
                 BTFSS
                                   AARGB0,MSB
                 RETLW
                                   0x00
                 MOVWF
                                   AARGB7
                 MOVF
                                   AARGB4,W
                                   AARGB1,W
                  SUBWF
                 BTFSS
                                   _Z
                  GOTO
                                   FLOOR 24RNDL
                  RETLW
                                   0x00
FLOOR24RNDL
                                   AARGB7,W
                  COME
                                   TEMPB1
                 MOVWF
                  INCF
                                   TEMPB1,W
                 ADDWF
                                   AARGB1,F
                 BTFSC
                                   _{\rm Z}
                  INCF
                                   AARGB0, F
                  BTFSS
                                   _Z
                                                     ; has rounding caused carryout?
                  RETLW
                                   0x00
                                   AARGB0,F
                 RRF
                 RRF
                                   AARGB1,F
                  INCFSZ
                                   AEXP,F
                                                     ; check for overflow
                 RETLW
                                   0x00
                  GOTO
                                   SETFOV24
```

## **AN660**

FLOOR24MASKH			
F LOOK Z THASKII	MOVLW	0x07 ;	get remainder for mask pointer
	ANDWF	TEMPBO,F	get remainder for mask pointer
			D.
	MOVLW	LOW FLOOR24MASKTABI	ır.
	ADDWF	TEMPBO, F	
	MOVLW	HIGH FLOOR24MASKTAE	311F
	BTFSC	_C	
	ADDLW	0x01	
	MOVWF	PCLATH	
	INCF	TEMPB0,W	
	CALL	FLOOR24MASKTABLE;	access table for mask
	ANDWF	AARGB0,F	
	CLRF	AARGB1	
	BTFSS	AARGB0,MSB ;	if negative, round down
	RETLW	0x00	
	MOVWF	AARGB7	
	MOVF	AARGB4,W	
	SUBWF	AARGB1,W	
	BTFSS	_Z	
	GOTO		
		FLOOR24RNDH	
	MOVF	AARGB3,W	
	SUBWF	AARGB0,W	
	BTFSS	_Z	
	GOTO	FLOOR24RNDH	
	RETLW	0x00	
FLOOR24RNDH			
	COMF	AARGB7,W	
	MOVWF	TEMPB1	
	INCF	TEMPB1,W	
	ADDWF	AARGB0,F	
	BTFSS	_C ;	has rounding caused carryout?
	RETLW	0x00	
	RRF	AARGB0,F	
	RRF	AARGB1,F	
	INCFSZ	AEXP,F	
	RETLW	0x00	
	GOTO	SETFOV24 ;	check for overflow
FLOOR24ZERO			
	BTFSC	AARGB0,MSB	
	GOTO	FLOOR 24MINUSONE	
	CLRF	AEXP	
	CLRF	AARGB0	
	CLRF	AARGB1	
	RETLW	0x00	
TT 0 0 D 0 4 1			
FLOOR24MINUSONE	MOTITIE	078	
	MOVLW	0x7F	
	MOVWF	AEXP	
	MOVLW	0x80	
	MOVWF	AARGB0	
	CLRF	AARGB1	
	RETLW	0x00	
;			
			masking, using pointer from
; the rema	ainder of the num	per of zero bits div	rided by eight.
FLOOR24MASKTABLE	1		
THOUSTHANKINDLE	MOVWF	PCL	
	RETLW	0xFF	
	VETTAM	VAFF	

```
0xFE
               RETIW
               RETLW
                               0xFC
               RETLW
                               0xF8
               RETLW
                               0xF0
               RETLW
                               0xE0
               RETIM
                               0xC0
                               0x80
               RETLW
               RETLW
                               0x00
Evaluate log10(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
               CALL
                       LOG1024
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- LOG( AARG )
       Testing on [MINNUM, MAXNUM] from 100000 trials:
               min
                               mean
                       max
       Timing: 1674
                               3383.3 clks
                       3567
;
               min
                       max
                               mean
                                       rms
                       0x00
       Error: -0x01
                               -0.32
                                       0.57
       This approximation of the natural log function is based upon the
       expansion
               log10(x) = log10(2) * log2(x) = log10(2) * (n + log2(f))
       where .5 <= f < 1 and n is an integer. The additional transformation
                       2*f-1, f < 1/sqrt(2), n=n-1
                       f-1, otherwise
       produces a naturally segmented representation of log2(1+z) on the
       intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
       approximations.
LOG1024
               CLRF
                               AARGB2
                                               ; clear next significant byte
               MOVF
                               AEXP,W
               BTFSS
                               AARGB0,MSB
                                               ; test for negative argument
                                               ; test for zero argument
               BTFSC
                               _{\rm Z}
                               DOMERR24
               GOTO
               MOVF
                               FPFLAGS,W
                                               ; save rounding flag
               MOVWF
                               DARGB3
               BCF
                               FPFLAGS, RND
                                               ; disable rounding
               MOVF
                               AEXP,W
               MOVWF
                               EARGB3
               MOVLW
                               EXPBIAS-1
               SUBWF
                               EARGB3,F
               MOVWF
                               AEXP
                                               ; .70710678118655 = 7E3504F3
               MOVLW
                               0xF3
```

```
SUBWF
                                    AARGB2,W
                  MOVLW
                                    0x04
                  MOVWF
                                    TEMPB0
                  BTFSS
                                    _C
                                    TEMPB0,W
                  INCFSZ
                  SUBWF
                                    AARGB1,W
                  MOVLW
                                    0x35
                  MOVWF
                                    TEMPB0
                                    _C
                  BTFSS
                  INCFSZ
                                    TEMPB0,W
                  SUBWF
                                    AARGB0,W
                  BTFSS
                                    _C
                                    LOG1024L
                  GOTO
        minimax rational approximation on [0,.sqrt(2)-1]
LOG1024H
                                    0x7F
                  MOVLW
                                    BEXP
                  MOVWF
                  CLRF
                                    BARGB0
                  CLRF
                                    BARGB1
                  CLRF
                                    BARGB2
                  CALL
                                    FPS32
                  MOVF
                                    AEXP,W
                  MOVWF
                                    DEXP
                  MOVF
                                    AARGB0,W
                  MOVWF
                                    DARGB0
                  MOVF
                                    AARGB1,W
                                    DARGB1
                  MOVWF
                  MOVF
                                    AARGB2,W
                  MOVWF
                                    DARGB2
                  POLL124
                                    LOG24HQ,2,0
                  MOVF
                                    AEXP,W
                  MOVWF
                                    CEXP
                  MOVF
                                    AARGB0,W
                  MOVWF
                                    CARGB0
                                    AARGB1,W
                  MOVF
                  MOVWF
                                    CARGB1
                  MOVF
                                    AARGB2,W
                  MOVWF
                                    CARGB2
                                    DEXP,W
                  MOVF
                  MOVWF
                                    AEXP
                  MOVF
                                    DARGB0,W
                  MOVWF
                                    AARGB0
                  MOVF
                                    DARGB1,W
                  MOVWF
                                    AARGB1
                  MOVF
                                    DARGB2,W
                  MOVWF
                                    AARGB2
                  POL24
                                    LOG24HP, 1, 0
                  MOVF
                                    CEXP,W
                  MOVWF
                                    BEXP
                  MOVF
                                    CARGB0,W
                  MOVWF
                                    BARGB0
                                    CARGB1,W
                  MOVF
                  MOVWF
                                    BARGB1
                  MOVF
                                    CARGB2,W
                                    BARGB2
                  MOVWF
```

```
CALL
                                   FPD32
                 GOTO
                                   LOG1024OK
        minimax rational approximation on [1/sqrt(2)-1,0]
LOG1024L
                                   AEXP,F
                 INCF
                 MOVLW
                                   0x7F
                 MOVWF
                                   BEXP
                 CLRF
                                   BARGB0
                                   BARGB1
                 CLRF
                 CLRF
                                   BARGB2
                 CALL
                                   FPS32
                                   EARGB3,F
                 DECF
                 MOVF
                                   AEXP,W
                                   DEXP
                 MOVWF
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   DARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   DARGB2
                                   LOG24LQ,2,0
                 POLL124
                 MOVF
                                   AEXP,W
                 MOVWF
                                   CEXP
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   CARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   CARGB1
                 MOVF
                                   AARGB2,W
                                   CARGB2
                 MOVWF
                 MOVF
                                   DEXP,W
                 MOVWF
                                   AEXP
                                   DARGB0,W
                 MOVF
                 MOVWF
                                   AARGB0
                 MOVF
                                   DARGB1,W
                 MOVWF
                                   AARGB1
                 MOVF
                                   DARGB2,W
                 MOVWF
                                   AARGB2
                 POL24
                                   LOG24LP,1,0
                 MOVF
                                   CEXP,W
                 MOVWF
                                   BEXP
                 MOVF
                                   CARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   CARGB1,W
                 MOVWF
                                   BARGB1
                                   CARGB2,W
                 MOVF
                 MOVWF
                                   BARGB2
                 CALL
                                   FPD32
LOG1024OK
                 MOVF
                                   DEXP,W
                 MOVWF
                                   BEXP
                 MOVF
                                   DARGB0,W
                                   BARGB0
                 MOVWF
```

	MOVF	DARGB1,W				
	MOVWF	BARGB1				
	MOVF	DARGB2,W				
	MOVWF	BARGB2				
	CALL	FPM32				
	MOVF	AEXP,W				
	MOVWF	DEXP				
	MOVF	AARGB0,W				
	MOVWF	DARGB0				
	MOVF	AARGB1,W				
	MOVWF	DARGB1				
	MOVF	AARGB2,W				
	MOVWF	DARGB2				
	OI DE	AADCDO				
	CLRF	AARGB0				
	MOVF	EARGB3,W				
	MOVWF	AARGB1				
	BTFSC	AARGB1,MSB				
	COMF	AARGB0,F				
	CALL	FL01624				
	CLRF	AARGB2				
	MOVF	DEXP,W				
	MOVWF	BEXP				
	MOVF	DARGB0,W				
	MOVWF	BARGB0				
	MOVF	DARGB1,W				
	MOVWF	BARGB1				
	MOVE	DARGB2,W				
	MOVWF	BARGB2				
	CALL	FPA32				
; fixed po	oint multiplicati	on by log10(2)				
	MOVF	AARGB0,W				
	MOVWF	EARGB3				
	BSF	AARGB0,MSB				
	MOVLW	0x9A				
	MOVWF	BARGB0				
	MOVLW	0x20				
	MOVWF	BARGB1				
	MOVLW	0x9B				
	MOVWF	BARGB2				
	CATT	EVMO A O AII				
	CALL DECF	FXM2424U AEXP,F				
	DECF	AEAP, F				
	BTFSC	AARGB0,MSB				
	GOTO	LOG1024DONE				
	RLF	AARGB3,F				
	RLF	AARGB2,F				
	RLF	AARGB1,F				
	RLF DECF	AARGB0,F AEXP,F				
LOG1024DONE	BTFSS	EARGB3,MSB				
	BCF	AARGB0,MSB				
	DTECC	האם כם 2 האה				
	BTFSS	DARGB3,RND				
	RETLW	0x00				
	BSF	FPFLAGS, RND	;	restore	rounding	flag
					9	

	CALL RETLW	RND3224 0x00	
DOMERR24	BSF RETLW	FPFLAGS,DOM	; domain error
;			
		66i ni naka 6.00 1.00/	1) (
; minima	x rational coe	fficients for log2(	1+x)/x on [1/sqrt(2)-1,0]
LOG24HP0	EQU	0x81	; LOG24HP0 = .73551298732E+1
LOG24HP00	EQU	0x6B	
LOG24HP01 LOG24HP02	EQU	0x5D 0x39	
LOGZ4HPUZ	EQU	0X39	
LOG24HP1	EQU	0x81	; $LOG24HP1 = .40900513905E+1$
LOG24HP10	EQU	0x02	
LOG24HP11	EQU	0xE1	
LOG24HP12	EQU	0xB3	
LOG24HQ0	EQU	0x81	; LOG24HOO = .50982159260E+1
LOG24HQ00	EQU	0x23	/ LOG24nQ050962159200E+1
LOG24HO01	EQU	0x24	
LOG24HQ02	EQU	0x96	
LOG24HQ1	EQU	0x81	; LOG24HQ1 = .53849258895E+1
LOG24HQ10	EQU	0x2C	
LOG24HQ11	EQU	0x51 0x50	
LOG24HQ12	EQU	0.00	
LOG24HQ2	EQU	0x7F	; LOG24HQ2 = 1.0
LOG24HQ20	EQU	0x00	
LOG24HQ21	EQU	0x00	
LOG24HQ22	EQU	0x00	
,			
; minima	x rational coe	fficients for log2(	1+x)/x on $[0,sqrt(2)-1]$
LOG24LP0	EQU	0x82	; LOG24LP0 = .103115556038E+2
LOG24LP00	EQU	0x24	/ HOGZIHIO103113330030H·Z
LOG24LP01	EQU	0xFC	
LOG24LP02	EQU	0x22	
LOG24LP1	EQU	0x81	; LOG24LP1 = .457749066375E+1
LOG24LP10	EQU	0x12	
LOG24LP11	EQU	0x7A	
LOG24LP12	EQU	0xCE	
LOG24LQ0	EQU	0x81	; $LOG24LQ0 = .714746549793E+1$
LOG24LQ00	EQU	0x64	
LOG24LQ01	EQU	0xB8	
LOG24LQ02	EQU	0x0A	
LOG24LQ1	EQU	0x81	; LOG24LQ1 = .674551124538E+1
LOG24LQ10	EQU	0x57	/ LOG24LQ10/4551124556E+1
LOG24LQ11	EQU	0xDB	
LOG24LQ12	EQU	0x3A	
<b>.</b>	~ -		
LOG24LQ2	EQU	0x7F	; $LOG24LQ2 = 1.0$
LOG24LQ20	EQU	0x00	
LOG24LQ21	EQU	0x00	
LOG24LQ22	EQU	0x00	
; * * * * * * * * * * * *	******	******	**********

```
*****************************
        Evaluate log(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Use:
               CALL
                       LOG24
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- LOG( AARG )
        Testing on [MINNUM, MAXNUM] from 100000 trials:
               min
                       max
                               mean
        Timing: 1662
                       3555
                               3371.2 clks
               min
                        max
                               mean
                                        rms
                               -0.80
        Error: -0x02
                       0x00
                                       0.92
        This approximation of the natural log function is based upon the
        expansion
                log(x) = log(2) * log2(x) = log(2) * (n + log2(f))
        where .5 <= f < 1 and n is an integer. The additional transformation
                        2*f-1, f < 1/sqrt(2), n=n-1
                       f-1, otherwise
        produces a naturally segmented representation of log2(1+z) on the
        intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
        approximations.
LOG24
                                AARGB2
                CLRF
                               AARGB0,MSB
                BTFSC
                                               ; test for negative argument
                GOTO
                                DOMERR24
                MOVF
                               AEXP,W
                                                ; test for zero argument
                BTFSC
                                _{\rm Z}
                               DOMERR24
                GOTO
                MOVF
                                FPFLAGS, W
                                                ; save rounding flag
                MOVWF
                                DARGB3
                                FPFLAGS, RND
                BCF
                                               ; disable rounding
                MOVF
                                AEXP,W
                MOVWF
                                EARGB3
                                EXPBIAS-1
               MOVLW
                SUBWE
                                EARGB3,F
                MOVWF
                                AEXP
                MOVLW
                                0xF3
                                                ; .70710678118655 = 7E3504F3
                               AARGB2,W
                SUBWF
                MOVLW
                                0x04
                                TEMP
                MOVWF
                BTFSS
                              TEMP,W
                INCFSZ
                SUBWE
                                AARGB1,W
                                0x35
                MOVLW
                                TEMP
                MOVWF
                BTFSS
                                _C
                INCFSZ
                                TEMP,W
```

```
SUBWF
                                   AARGB0,W
                 BTFSS
                                   _C
                                   LOG24L
                 GOTO
        minimax rational approximation on [0,.sqrt(2)-1]
LOG24H
                                   0x7F
                 MOVLW
                 MOVWF
                                   BEXP
                                   BARGB0
                 CLRF
                 CLRF
                                   BARGB1
                 CLRF
                                   BARGB2
                                   FPS32
                 CALL
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   DARGB0
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                                   DARGB2
                 MOVWF
                 POLL124
                                   LOG24HQ,2,0
                 MOVF
                                   AEXP,W
                                   CEXP
                 MOVWF
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   CARGB0
                 MOVF
                                   AARGB1,W
                                   CARGB1
                 MOVWF
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   CARGB2
                                   DEXP,W
                 MOVF
                                   AEXP
                 MOVWF
                 MOVF
                                   DARGB0,W
                 MOVWF
                                   AARGB0
                 MOVF
                                   DARGB1,W
                 MOVWF
                                   AARGB1
                 MOVF
                                   DARGB2,W
                 MOVWF
                                   AARGB2
                                   LOG24HP,1,0
                 POL24
                                   CEXP,W
                 MOVF
                 MOVWF
                                   BEXP
                 MOVF
                                   CARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   CARGB1,W
                 MOVWF
                                   BARGB1
                 MOVF
                                   CARGB2,W
                 MOVWF
                                   BARGB2
                 CALL
                                   FPD32
                 GOTO
                                   LOG240K
        minimax rational approximation on [1/sqrt(2)-1,0]
LOG24L
                 INCF
                                   AEXP,F
                 MOVLW
                                   0x7F
                 MOVWF
                                   BEXP
```

CLRF	BARGB0
CLRF	BARGB1
CLRF	BARGB2
CALL	FPS32
CALL	11002
DECF	EARGB3,F
MOVF	AEXP,W
MOVWF	DEXP
MOVF	AARGB0,W
MOVWF	DARGB0
MOVF	AARGB1,W
MOVWF	DARGB1
MOVF	AARGB2,W
MOVWF	DARGB2
POLL124	LOG24LQ,2,0
	~, , ,
MOVF	AEXP,W
MOVWF	CEXP
MOVF	AARGB0,W
MOVWF	CARGB0
MOVF	AARGB1,W
MOVWF	CARGB1
MOVF	AARGB2,W
MOVWF	CARGB2
	<b>DD</b>
MOVF	DEXP,W
MOVWF	AEXP
MOVF	DARGB0,W
MOVWF	AARGB0
MOVF	DARGB1,W
MOVWF	AARGB1
MOVF	DARGB2,W
MOVWF	AARGB2
POL24	LOG24LP,1,0
MOVF	CEXP,W
MOVWF	BEXP
MOVF	CARGB0,W
MOVWF MOVF	BARGB0 CARGB1,W
	BARGB1
MOVE	
MOVE	CARGB2,W
MOVWF	BARGB2
CALL	FPD32
MOTTE	DEAD M
MOVE	DEXP,W
MOVWF	BEXP
MOVE	DARGBO,W
MOVWF	BARGB0
MOVF	DARGB1,W
MOVWF	BARGB1
MOVF	DARGB2,W
MOVWF	BARGB2
CALL	FPM32
MOVIE	יי מעיוג
MOVF	AEXP,W
MOVWF	DEXP
MOVF	AARGB0,W
MOVWF	DARGB0

LOG240K

	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	MOVF	AARGB2,W	
	MOVWF	DARGB2	
	CLRF	AARGB0	
	MOVF	EARGB3,W	
	MOVWF	AARGB1	
	BTFSC	AARGB1,MSB	
	COMF	AARGB0,F	
	CALL	FLO1624	
	CLRF	AARGB2	
	MOVF	DEXP,W	
	MOVWF	BEXP	
	MOVF	DARGB0,W	
	MOVWF	BARGB0	
	MOVF	DARGB1,W	
	MOVWF	BARGB1	
	MOVF	DARGB2,W	
	MOVWF	BARGB2	
	CALL	FPA32	
; fixed	l point multipl	ication by log(2)	
	MOVF	AEXP,W	
	BTFSC	_Z	
	RETLW	0x00	
	MOVF	AARGB0,W	
	MOVWF	EARGB3	
	BSF	AARGB0,MSB	
	MOVLW	0xB1	
	MOVWF	BARGB0	
	MOVLW	0x72	
	MOVWF	BARGB1	
	MOVLW	0x18	
	MOVWF	BARGB2	
	CALL	FXM2424U	
	BTFSC	AARGB0,MSB	
	GOTO	LOG24DONE	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
T 0 0 0 4 p	pm=6-5		
LOG24DONE	BTFSS	EARGB3,MSB	
	BCF	AARGB0,MSB	
	BTFSS	DARGB3,RND	
	RETLW	0x00	
	202		
	BSF	FPFLAGS, RND	
	GOTO	RND3224	
DOMERR24	BSF	FPFLAGS,DOM	; domain error
	RETLW	0xFF	
:			
,			

```
minimax rational coefficients for log2(1+x)/x on [1/sqrt(2)-1,0]
LOG24HP0
                EQU
                                0x81
                                                 ; LOG24HP0 = .73551298732E+1
LOG24HP00
                EQU
                                0x6B
                                0x5D
LOG24HP01
                EQU
LOG24HP02
                                0x39
                EQU
LOG24HP1
                EQU
                                0x81
                                                 ; LOG24HP1 = .40900513905E+1
LOG24HP10
                EQU
                                0x02
LOG24HP11
                                0xE1
                EQU
LOG24HP12
                EQU
                                0xB3
                                                 ; LOG24HQ0 = .50982159260E+1
LOG24HQ0
                EQU
                                0x81
LOG24HQ00
                EQU
                                0x23
                                0 \times 24
LOG24HO01
                EOU
LOG24HQ02
                EQU
                                0x96
LOG24HQ1
                EQU
                                0x81
                                                 ; LOG24HQ1 = .53849258895E+1
LOG24HQ10
                                0x2C
                EQU
LOG24HQ11
                EQU
                                0x51
LOG24HQ12
                EQU
                                0x50
LOG24HQ2
                EQU
                                0x7F
                                                 ; LOG24HQ2 = 1.0
LOG24HQ20
                                0x00
                EQU
LOG24HQ21
                EQU
                                0x00
LOG24HQ22
                                0x00
                EQU
       minimax rational coefficients for log2(1+x)/x on [0,sqrt(2)-1]
LOG24LP0
                EQU
                                0x82
                                                 ; LOG24LP0 = .103115556038E+2
LOG24LP00
                EQU
                                0 \times 24
LOG24LP01
                                0xFC
                EQU
LOG24LP02
                EOU
                                0x22
                                0x81
                                                 ; LOG24LP1 = .457749066375E+1
LOG24LP1
                EQU
LOG24LP10
                EQU
                                0x12
LOG24LP11
                EQU
                                0x7A
LOG24LP12
                                0xCE
                EQU
LOG24LQ0
                EQU
                                0x81
                                                 ; LOG24LQ0 = .714746549793E+1
                                0x64
LOG24LQ00
                EQU
LOG24LO01
                EQU
                                0xB8
LOG24LQ02
                                0x0A
                EQU
LOG24LQ1
                EQU
                                0x81
                                                 ; LOG24LQ1 = .674551124538E+1
                                0x57
LOG24LQ10
                EQU
LOG24LQ11
                EQU
                                0xDB
LOG24LQ12
                EQU
                                0x3A
                                0x7F
LOG24LQ2
                EQU
                                                 ; LOG24LQ2 = 1.0
                                0x00
LOG24LQ20
                EQU
LOG24LQ21
                EQU
                                0x00
LOG24LQ22
                EQU
                                0x00
Nearest neighbor rounding
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                CALL
                        RND3224
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
```

```
Result: AARG <-- RND( AARG )
        Testing on [MINNUM, MAXNUM] from 10000 trials:
                 min
                         max
                                  mean
        Timing: 3
                         17
                                          clks
                 min
                         max
                                  mean
        Error: 0
                         Ω
                                  Ω
                                          nsb
RND3224
                                                  ; is NSB < 0x80?
                                  AARGB2,MSB
                 BTFSS
                 RETLW
                                  0x00
                 BSF
                                  _C
                                                   ; set carry for rounding
                                  0x7F
                 MOVLW
                 ANDWF
                                  AARGB2,W
                                  _Z
                 BTFSC
                 RRF
                                  AARGB1,W
                                                   ; select even if NSB = 0x80
                                  AARGB0,W
                 MOVF
                 MOVWF
                                  SIGN
                                                   ; save sign
                                                   ; make MSB explicit
                                  AARGB0,MSB
                 BSF
                                  _{\rm Z}
                 BCF
                                  _C
                 BTFSC
                                                   ; round
                 INCF
                                  AARGB1,F
                 BTFSC
                                  _{\rm Z}
                 INCF
                                  AARGB0,F
                 BTFSS
                                                   ; has rounding caused carryout?
                                  _{\rm Z}
                                  RND32240K
                 GOTO
                 RRF
                                  AARGB0,F
                                                   ; if so, right shift
                                  AARGB1,F
                 RRF
                                  EXP,F
                 INCF
                                                   ; test for floating point overflow
                 BTFSC
                                  _{\rm Z}
                 GOTO
                                  SETFOV24
RND32240K
                                  SIGN, MSB
                 BTFSS
                 BCF
                                  AARGB0,MSB
                                                   ; clear sign bit if positive
                 RETLW
                                  0x00
        Evaluate cos(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                 CALL
                         COS24
        Use:
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- COS( AARG )
;
        Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 2736
                         4505
                                 4134.3 clks
```

```
min
                       max
                              mean
                                       rms
       Error:
               -0x56
                       0x13
                               -7.13
                                       20.90
                                              nsb
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x \mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
               sin(z) = z * p(z**2), cos(z) = q(z**2)
       where p and q are minimax polynomial approximations.
COS24
               MOVF
                               FPFLAGS,W
                                              ; save rounding flag
               MOVWF
                               DARGB3
               BCF
                               FPFLAGS, RND
                                              ; disable rounding
               CLRF
                               CARGB3
                                              ; initialize sign in CARGB3
               BCF
                               AARGB0,MSB
                                              ; use |x|
                               RRSINCOS24
               CALL
RRCOS240K
                               EARGB3,W
               RRF
               XORWF
                               EARGB3,W
               MOVWF
                               TEMPB0
               BTFSC
                               TEMPB0,LSB
               GOTO
                               COSZSIN24
               CALL
                               ZCOS24
               GOTO
                               COSSIGN24
COSZSIN24
               CALL
                               ZSIN24
COSSIGN24
               MOVLW
                               0x80
                               EARGB3,LSB+1
               BTFSC
               XORWF
                               CARGB3,F
               BTFSC
                               CARGB3, MSB
               XORWF
                               AARGB0,F
                               DARGB3, RND
               BTFSS
               RETLW
                               0x00
               BSF
                               FPFLAGS, RND
                                             ; restore rounding flag
               CALL
                               RND3224
                               0x00
               RETLW
Evaluate sin(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
               CALL
                       SIN24
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- SIN( AARG )
```

```
Testing on [-LOSSTHR, LOSSTHR] from 100000 trials:
               min
                               mean
                       max
       Timing: 2564
                               4134.5 clks
                       4494
               min
                       max
                               mean
                                       rms
       Error: -0x56
                               -7.12
                                       20.89
                       0x13
                                               nsb
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
;
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x \mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
               sin(z) = z * p(z**2), cos(z) = q(z**2)
       where p and q are minimax polynomial approximations.
SIN24
               MOVF
                               FPFLAGS,W
                                               ; save rounding flag
               MOVWF
                               DARGB3
               BCF
                               FPFLAGS, RND
                                               ; disable rounding
                                               ; initialize sign in CARGB3
               CLRF
                               CARGB3
                               AARGB0,MSB
                                               ; toggle sign if x < 0
               BTFSC
               BSF
                               CARGB3, MSB
               BCF
                               AARGB0,MSB
                                               ; use |x|
                               RRSINCOS24
               CALL
RRSIN240K
               RRF
                               EARGB3,W
               XORWF
                               EARGB3,W
               MOVWF
                               TEMPB0
                               TEMPB0,LSB
               BTFSC
               GOTO
                               SINZCOS24
               CALL
                               ZSIN24
               GOTO
                               SINSIGN24
SINZCOS24
               CALL
                               ZCOS24
SINSIGN24
               MOVLW
                               0x80
                               CARGB3,MSB
               BTFSC
               XORWF
                               AARGB0,F
               BTFSS
                               DARGB3, RND
                               0x00
               RETLW
               BSF
                               FPFLAGS, RND
                                               ; restore rounding flag
               CALL
                               RND3224
               RETLW
                               0x00
Evaluate sin(x) and cos(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
               CALL
                       SINCOS24
```

```
Output: 24 bit floating point numbers in AEXP, AARGBO, AARGB1 and
                 BEXP, BARGBO, BARGB1
        Result: AARG <-- COS( AARG )
                 BARG <-- SIN( AARG )
        Testing on [-LOSSTHR, LOSSTHR] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 4525
                         6478
                                  6032.2 clks
                 min
                         max
                                  mean
                                          rms
        Error:
                -0x56
                         0x13
                                  -7.12
                                          20.89
                                                   nsb
                                                           sine
                 -0x56
                         0x13
                                  -7.13
                                          20.90
                                                           cosine
        The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition z = x mod pi/4, with an additional variable j
        indicating the correct octant, leading to the appropriate call
        to either the sine or cosine approximations
                 sin(z) = z * p(z**2), cos(z) = q(z**2)
        where p and q are minimax polynomial approximations. In this case,
        only one range reduction is necessary.
SINCOS24
                 MOVF
                                  FPFLAGS, W
                                                   ; save rounding flag
                 MOVWF
                                  DARGB3
                                  FPFLAGS, RND
                 BCF
                                                   ; disable rounding
                MOVF
                                  AEXP,W
                                                   ; save x in EARG
                 MOVWF
                                  EEXP
                 MOVF
                                  AARGB0,W
                 MOVWF
                                  EARGB0
                 MOVF
                                  AARGB1,W
                                  EARGB1
                 MOVWF
                 CLRF
                                  EARGB2
                 BCF
                                  AARGB0,MSB
                                                   ; use |x|
                                  CARGB3
                 CLRF
                                                   ; initialize sign in CARGB3
                                  RRSINCOS24
                 CALL
                                                   ; range reduction
                 MOVF
                                  CARGB3,W
                                                   ; save sign from range reduction
                 MOVWF
                                  ZARGB3
                 MOVLW
                                  0x80
                 BTFSC
                                  EARGB0,MSB
                                                   ; toggle sign if x < 0
                 XORWF
                                  CARGB3,F
                 CALL
                                  RRSIN240K
                 BTFSC
                                  DARGB3, RND
                 CALL
                                  RND3224
                                                   ; save sin(x) in EARG
                 MOVF
                                  AEXP,W
                 MOVWF
                                  EEXP
                                  AARGB0,W
                 MOVF
                                  EARGB0
                 MOVWF
                 MOVF
                                  AARGB1,W
                 MOVWF
                                  EARGB1
```

```
MOVF
                                 AARGB2.W
                MOVWF
                                 EARGB2
                MOVF
                                 DEXP,W
                                                  ; restore z*z in AARG
                MOVWF
                                 AEXP
                MOVF
                                 DARGB0,W
                MOVWF
                                 AARGB0
                MOVF
                                 DARGB1,W
                MOVWF
                                 AARGB1
                                 DARGB2.W
                MOVF
                MOVWF
                                 AARGB2
                MOVF
                                 ZARGB3,W
                                                  ; restore sign from range reduction
                MOVWF
                                 CARGB3
                CALL
                                 RRCOS240K
                MOVF
                                 EEXP,W
                                                  ; restore sin(x) in BARG
                MOVWF
                                 BEXP
                MOVF
                                 EARGBO.W
                MOVWF
                                 BARGB0
                MOVF
                                 EARGB1,W
                MOVWF
                                 BARGB1
                                 EARGB2,W
                MOVF
                                 BARGB2
                MOVWF
                BTFSS
                                 DARGB3, RND
                RETLW
                                 0x00
                                 FPFLAGS, RND
                BSF
                                                  ; restore rounding flag
                                 RND3224
                CALL
                RETLW
                                 0x00
                ********************
        Range reduction routine for trigonometric functions
;
        The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition
                z = x \mod pi/4,
        produced by first evaluating y and j through the relations
                y = floor(x/(pi/4)), j = y - 8*[y/8].
;
        where j equals the correct octant. For j odd, adding one to j
;
        and y eliminates the odd octants. Additional logic on j and the
        sign of the result leads to appropriate use of the sine or cosine
        routine in each case.
        The calculation of z is then obtained through a pseudo extended
        precision method
;
                z = x \mod pi/4 = x - y*(pi/4) = ((x - p1*y)-p2*y)-p3*y
        where pi/4 = p1 + p2 + p3, with p1 close to pi/4 and p2 close to
;
        \mathrm{pi}/4 - \mathrm{pl}. The numbers \mathrm{pl} and \mathrm{p2} are chosen to have an exact
        machine representation with slightly more than the lower half of
;
        the mantissa bits zero, typically leading to no error in computing
;
        the terms in parenthesis. This calculation breaks down leading to
        a loss of precision for |x| > LOSSTHR = sqrt(2**24)*pi/4, or for |x|
        close to an integer multiple of pi/4. This loss threshold has been
        chosen based on the efficacy of this calculation, with a domain error
        reported if this threshold is exceeded.
```

```
RRSINCOS24
                  MOVF
                                   AEXP,W
                                                      ; loss threshold check
                  SUBLW
                                   LOSSTHR24EXP
                  BTFSS
                                   _C
                  GOTO
                                   DOMERR24
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                   RRSINCOS24ARGOK
                  MOVF
                                   AARGB0,W
                  SUBLW
                                   LOSSTHR24B0
                  BTFSS
                                   _C
                  GOTO
                                   DOMERR24
                  BTFSS
                                   _{\mathbf{Z}}
                                   RRSINCOS24ARGOK
                  GOTO
                 MOVF
                                   AARGB1,W
                  SUBLW
                                   LOSSTHR24B1
                  BTFSS
                                    _C
                                   DOMERR24
                  GOTO
RRSINCOS24ARGOK
                                   AEXP,W
                  MOVF
                                                      ; save |x| in CARG
                                   CEXP
                 MOVWF
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   CARGB0
                  MOVF
                                   AARGB1,W
                 MOVWF
                                   CARGB1
                                   CARGB2
                  CLRF
         fixed point multiplication by 4/pi
                  BSF
                                   AARGB0,MSB
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   AARGB1,W
                                   BARGB1
                 MOVWF
                 MOVLW
                                   0xA2
                                                     i = 1.27323954474
                  MOVWF
                                   AARGB0
                  MOVLW
                                   0xF9
                  MOVWF
                                   AARGB1
                 MOVLW
                                   0x83
                 MOVWF
                                   AARGB2
                                   FXM2416U
                  CALL
                  INCF
                                   AEXP,F
                  BTFSC
                                   AARGB0, MSB
                  GOTO
                                   RRSINCOS24YOK
                 RLF
                                   AARGB3,F
                                   AARGB2,F
                  RLF
                                   AARGB1,F
                  RLF
                  RLF
                                   AARGB0,F
                  DECF
                                   AEXP,F
RRSINCOS24YOK
                  BCF
                                   AARGB0,MSB
                                   INT3224
                                                     ; y = [|x| * (4/pi)]
                  CALL
                                   AARGB2,LSB
                  BTFSS
                  GOTO
                                   SAVEY24
                                   AARGB2,F
                  INCF
```

```
BTFSC
                                    _Z
                  INCF
                                   AARGB1,F
                  BTFSC
                                    _{\rm Z}
                                   AARGB0,F
                  INCF
SAVEY24
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   DARGB0
                                                      ; save y in DARG
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   DARGB1
                                   AARGB2,W
                 MOVF
                 MOVWF
                                   DARGB2
                                   0x07
                                                      ; j = y \mod 8
                 MOVLW
                 ANDWF
                                   AARGB2,F
                 MOVLW
                                   0x03
                  SUBWF
                                   AARGB2,W
                                   0x80
                 MOVLW
                 BTFSS
                                   _C
                                   JOK24
                  GOTO
                  XORWF
                                   CARGB3,F
                 MOVLW
                                   0x04
                                   AARGB2,F
                  SUBWF
JOK24
                 MOVF
                                   AARGB2,W
                                   EARGB3
                                                      ; save j in EARGB3
                 MOVWF
                 MOVF
                                   DARGB0,W
                 MOVWF
                                   AARGB0
                                                      ; restore y to AARG
                 MOVF
                                   DARGB1,W
                                   AARGB1
                 MOVWF
                                   DARGB2,W
                 MOVF
                                   AARGB2
                 MOVWF
                                   FLO2432
                  CALL
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                                      ; save y in DARG
                  BTFSC
                                    _{\rm Z}
                                   RRSINCOS24ZEQX
                  GOTO
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   DARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                                   DARGB2
                 MOVWF
;
         Cody-Waite extended precision calculation of |x| - y * pi/4 using
         fixed point multiplication. Since y >= 1, underflow is not possible
         in any of the products.
                 BSF
                                   AARGB0,MSB
                 MOVLW
                                   0xC9
                                                      ; - p1 = -.78515625
                                   BARGB0
                 MOVWF
                  CLRF
                                   BARGB1
                  CALL
                                   FXM2416U
                 BTFSC
                                   AARGB0,MSB
                                   RRSINCOS24Z1OK
                 GOTO
                 RLF
                                   AARGB3,F
                 RLF
                                   AARGB2,F
                                   AARGB1,F
                 RLF
```

	RLF	AARGB0,F	
	DECF	AEXP,F	
	DECI	110111 / 1	
RRSINCOS24Z1OK			
	MOVF	CEXP,W	; restore x to BARG
	MOVWF	BEXP	
	MOVF	CARGB0,W	
	MOVWF	BARGB0	
	MOVF	CARGB1,W	
	MOVWF	BARGB1	
	CLRF	BARGB2	
	CLRF	DARGDZ	
	CALL	FPA32	; z1 =  x  - y * (p1)
	CILLE	111132	, 21 -  X  1 (P1)
	MOVF	AEXP,W	
	MOVWF	CEXP	; save z1 in CARG
	MOVF	AARGB0,W	
	MOVWF	CARGB0	
	MOVF	AARGB1,W	
	MOVWF	CARGB1	
	MOVF	AARGB2,W	
	MOVWF		
	MOVWE	CARGB2	
	MOVF	DEAD M	
	MOVWF	DEXP,W	
		AEXP	
	MOVF	DARGBO,W	
	MOVWF	AARGB0	; restore y to AARG
	MOVF	DARGB1,W	
	MOVWF	AARGB1	
	MOVF	DARGB2,W	
	MOVWF	AARGB2	
	BSF	AARGB0,MSB	
	MOVIT M	Oven	; - p2 =00024187564849853515624
	MOVLW	0xFD	/ - pz =0002416/364649633313624
	MOVWF	BARGB0	
	MOVLW	0xA0	
	MOVWF	BARGB1	
	CALL	FXM2416U	
	CALL	FAM24100	
	MOVLW	0x0D - 1	
	NOVEW	OXOD I	
	BTFSC	AARGBO.MSR	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS24Z2OK	
	GOTO RLF	RRSINCOS24Z2OK AARGB3,F	
	GOTO RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F	
	GOTO RLF RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F	
	GOTO RLF RLF RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
	GOTO RLF RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F	
PPSTNCOS24720K	GOTO RLF RLF RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	; restore z1 to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF SUBWF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	; restore zl to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF SUBWF MOVF MOVWF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP	; restore zl to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W	; restore z1 to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVF MOVF MOVF MOVFF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0	; restore z1 to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVF MOVF MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W	; restore z1 to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVWF MOVF MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1	; restore zl to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W	; restore zl to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVWF MOVF MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1	; restore zl to BARG
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2	
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W	<pre>; restore z1 to BARG ; z2 = z1 - y * (p2)</pre>
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2 FPA32	
RRSINCOS24Z2OK	GOTO RLF RLF RLF RLF DECF  SUBWF  MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF	RRSINCOS24Z2OK AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  AEXP,F  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2	

```
CEXP
                                                     ; save z2 in CARG
                 MOVWF
                                   AARGB0,W
                 MOVF
                 MOVWF
                                   CARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   CARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   CARGB2
                 MOVF
                                   DEXP,W
                                   AEXP
                 MOVWF
                 MOVF
                                   DARGB0,W
                 MOVWF
                                   AARGB0
                                                     ; restore y to AARG
                 MOVF
                                   DARGB1,W
                 MOVWF
                                   AARGB1
                                   DARGB2,W
                 MOVF
                 MOVWF
                                   AARGB2
                 BSF
                                   AARGB0, MSB
                                                     i - p3 = -3.77489497744597636E-8
                 MOVLW
                                   0xA2
                                   BARGB0
                 MOVWF
                 MOVLW
                                   0x21
                 MOVWF
                                   BARGB1
                                   0x69
                 MOVLW
                 MOVWF
                                   BARGB2
                 CALL
                                   FXM2424U
                 MOVLW
                                   0x19 - 1
                 BTFSC
                                   AARGB0,MSB
                 GOTO
                                   RRSINCOS24Z3OK
                                   AARGB3,F
                 RLF
                 RLF
                                   AARGB2,F
                                   AARGB1,F
                 RLF
                 RLF
                                   AARGB0,F
                 DECF
                                   AEXP,F
RRSINCOS24Z3OK
                 SUBWF
                                   AEXP,F
                                                     ; restore z2 to BARG
                 MOVF
                                   CEXP,W
                                   BEXP
                 MOVWF
                 MOVF
                                   CARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   CARGB1,W
                 MOVWF
                                   BARGB1
                                   CARGB2,W
                 MOVF
                 MOVWF
                                   BARGB2
                 CALL
                                   FPA32
                                                     z = z2 - y * (p3)
                 MOVF
                                   AEXP,W
                                   CEXP
                                                     ; save z in CARG
                 MOVWF
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   CARGB0
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   CARGB1
                                   AARGB2,W
                 MOVF
                 MOVWF
                                   CARGB2
                                   AEXP,W
                 MOVF
                 MOVWF
                                   BEXP
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   BARGB0
                                   AARGB1,W
                 MOVF
```

```
MOVWF
                                   BARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   BARGB2
                                                     ; z * z
                 CALL
                                   FPM32
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                                     ; save z * z in DARG
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   DARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   DARGB2
                 RETLW
                                   0x00
RRSINCOS24ZEQX
                                   CEXP, W
                 MOVF
                 MOVWF
                                   AEXP
                                   CARGB0,W
                 MOVF
                 MOVWF
                                   AARGB0
                 MOVF
                                   CARGB1,W
                                   AARGB1
                 MOVWF
                 MOVF
                                   CARGB2,W
                 MOVWF
                                   AARGB2
                                   AEXP,W
                 MOVF
                 MOVWF
                                   BEXP
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   AARGB1,W
                                   BARGB1
                 MOVWF
                                   AARGB2,W
                 MOVF
                                   BARGB2
                 MOVWF
                                   FPM32
                                                     ; z * z
                 CALL
                 MOVF
                                   AEXP,W
                 MOVWF
                                   DEXP
                                                     ; save z * z in DARG
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   DARGB0
                                   AARGB1,W
                 MOVF
                 MOVWF
                                   DARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   DARGB2
                                   0x00
                 RETLW
DOMERR24
                 BSF
                                   FPFLAGS, DOM
                                                     ; domain error
                 RETLW
                                   0xFF
        minimax polynomial approximation p(x**2) on [0,pi/4]
ZCOS24
                 POL24
                                   COS24,3,0
                 RETLW
                                   0x00
        minimax polynomial approximation x*p(x**2) on [0,pi/4]
ZSIN24
                 POL24
                                   SIN24,2,0
```

```
MOVF
                               CEXP, W
               MOVWF
                               BEXP
               MOVF
                               CARGB0,W
               MOVWF
                               BARGB0
               MOVF
                               CARGB1,W
               MOVWE
                               BARGB1
                               CARGB2,W
               MOVF
               MOVWF
                               BARGB2
               CALL
                               FPM32
               RETLW
                               0x00
       minimax polynomial coefficients for \sin(z)/z = p(z^{**}2) on [0,pi/4]
                                               ; LP0 = .73551298732E+1*******
SIN240
               EQU
                               0x7E
SIN2400
                               0x7F
               EQU
SIN2401
                               0xFF
               EQU
SIN2402
               EQU
                               0xAC
SIN241
               EQU
                               0x7C
                                               ; LP1 = .40900513905E+1
SIN2410
                               0xAA
               EOU
SIN2411
               EQU
                               0x99
SIN2412
                               0x9D
               EQU
STN242
                               0x78
                                               ; LQ0 = .50982159260E+1
               EQU
SIN2420
               EQU
                               0x05
SIN2421
                               0x10
               EQU
SIN2422
               EQU
                               0x48
       minimax polynomial coefficients for cos(z) = q(z^{**}2) on [0,pi/4]
       with COS240 constrained to be 1.
                                              ; LP0 = .73551298732E+1*****
COS 240
               EQU
                               0x7F
COS2400
               EQU
                               0x00
COS2401
               EQU
                               0x00
COS2402
               EQU
                               0x00
COS241
               EOU
                               0x7D
                                               ; LP1 = .40900513905E+1
COS2410
                               0xFF
               EQU
COS2411
               EQU
                               0xFF
COS2412
               EQU
                               0xD0
COS242
               EQU
                               0x7A
                                               ; LQ0 = .50982159260E+1
COS2420
               EQU
                               0x2A
COS2421
               EQU
                               0x9E
COS2422
                               0x76
               EQU
COS243
                               0x75
                                               ; LQ1 = .53849258895E+1
               EQU
COS2430
               EQU
                               0xB2
COS2431
               EQU
                               0x12
COS2432
                               0xBF
               EOU
                        *****
Evaluate sqrt(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
               CALL
                       SQRT24
```

```
Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- SQRT( AARG )
        Testing on [0,MAXNUM] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 7
                         2968
                                  2517.5 clks
                 min
                         max
                                  mean
                                           rms
                         0x08
                                           3.60
        Error:
                -0x0b
                                  -1.35
                                                   nsb
        Range reduction for the square root function is naturally produced by
        the floating point representation,
                 x = f * 2**e, where 1 <= f < 2,
        leading to the expression
                             | sqrt(f) * 2**(e/2),e even
                               sqrt(f) * sqrt(2) * 2**(e/2),e odd
        The function \operatorname{sqrt}(f) is then approximated by a segmented fourth degree
        minimax polynomial on the intervals [1,1.5] and [1.5,2].
SORT24
                                  AARGB0,MSB
                 BTFSC
                                                   ; test for negative argument
                 GOTO
                                  DOMERR24
                 CLRF
                                  AARGB2
                                                   ; return if argument zero
                                  AEXP,W
                 MOVF
                 BTFSC
                                  Z
                 RETLW
                                  0x00
                 MOVF
                                  AEXP,W
                                                   ; save exponent in CEXP
                 MOVWF
                                  CEXP
                 MOVF
                                  FPFLAGS,W
                                                   ; save RND flag in DARGB3
                 MOVWF
                                  DARGB3
                 BCF
                                  FPFLAGS, RND
                                                   ; disable rounding
                 MOVLW
                                  EXPBIAS
                                                   ; compute z
                 MOVWF
                                  AEXP
                 MOVF
                                  AEXP,W
                                                   ; save z in DARG
                 MOVWF
                                  DEXP
                 MOVF
                                  AARGB0,W
                                  DARGB0
                 MOVWF
                 MOVF
                                  AARGB1,W
                 MOVWF
                                  DARGB1
                 CLRF
                                  DARGB2
                                  AARGB0,MSB-1
                 BTFSS
                 GOTO
                                  SQRT24L
SQRT24H
                 POL24
                                  SQRT24H,4,0
                                                   ; minimax approximation on [1.5,2]
                 GOTO
                                  SQRT240K
SQRT24L
                 POL24
                                  SQRT24L,4,0
                                                   ; minimax approximation on [1,1.5]
```

```
SQRT240K
                                   CEXP, LSB
                 BTFSC
                                                     ; is CEXP even or odd?
                 GOTO
                                   RRSQRTOK24
        fixed point multiplication by sqrt(2)
                 BSF
                                   AARGB0, MSB
                 MOVLW
                                   0xB5
                                                     ; sqrt(2) = 1.41421356237
                                   BARGB0
                 MOVWF
                 MOVLW
                                   0x04
                 MOVWF
                                   BARGB1
                 MOVLW
                                   0xF3
                 MOVWF
                                   BARGB2
                 CALL
                                   FXM2424U
                 INCF
                                   AEXP,F
                 BTFSC
                                   AARGB0,MSB
                                   RRSQRTOK24
                 GOTO
                 RLF
                                   AARGB3,F
                 RLF
                                   AARGB2,F
                                   AARGB1,F
                 RLF
                                   AARGB0,F
                 RLF
                 DECF
                                   AEXP,F
RRSQRTOK24
                 BCF
                                   AARGB0,MSB
                                                     ; make MSB implicit
                 MOVLW
                                   EXPBIAS
                                                     ; divide exponent by two
                 ADDWF
                                   CEXP,F
                                   CEXP,W
                 RRF
                                   AEXP
                 MOVWE
                 BTFSS
                                   DARGB3, RND
                 RETLW
                                   0x00
                                   FPFLAGS, RND
                 BSF
                 CALL
                                   RND3224
                 RETLW
                                   0x00
DOMERR24
                 BSF
                                   FPFLAGS,DOM
                                                     ; domain error
                 RETLW
                                   OxFF
        fourth degree minimax polynomial coefficients for sqrt(x) on [1.5,2]
SQRT24H0
                 EQU
                                   0x7D
                                                     ; SQRT24H0 = 3.5963132863E-1
SQRT24H00
                 EQU
                                   0x38
SQRT24H01
                 EQU
                                   0x21
SQRT24H02
                                   0x99
                 EQU
SQRT24H1
                                   0x7E
                                                     ; SQRT24H1 = 8.3106978456E-1
                 EQU
SQRT24H10
                 EQU
                                   0x54
SQRT24H11
                 EQU
                                   0xC0
SQRT24H12
                                   0xFD
                 EQU
SQRT24H2
                 EQU
                                   0x7C
                                                     ; SQRT24H2 = -2.3944355047E-1
SQRT24H20
                 EQU
                                   0xF5
SQRT24H21
                                   0x30
                 EQU
SQRT24H22
                 EQU
                                   0xB1
SQRT24H3
                 EQU
                                   0x7A
                                                     ; SQRT24H3 = 5.5047377031E-2
SQRT24H30
                 EQU
                                   0x61
SQRT24H31
                 EQU
                                   0x79
```

## **AN660**

```
SQRT24H32
                EQU
                                0x5C
SQRT24H4
                EQU
                                0x77
                                                ; SQRT24H4 = -5.6351436252E-3
SQRT24H40
                EQU
                                0xB8
SQRT24H41
                EQU
                                0xA7
SQRT24H42
                EQU
                                0x03
        fourth degree minimax polynomial coefficients for sqrt(x) on [1,1.5]
                EQU
                                0x7D
                                                ; SQRT24L0 = 3.0221977303E-1
SORT24L0
SQRT24L00
                EOU
                                0x1A
SQRT24L01
                EQU
                                0xBC
SQRT24L02
                EQU
                                0x8B
SQRT24L1
                                0x7E
                                                ; SQRT24L1 = 9.8831235597E-1
                EQU
SQRT24L10
                EQU
                                0x7D
SQRT24L11
                EQU
                                0x02
SQRT24L12
                EQU
                                0x0A
SQRT24L2
                EQU
                                0x7D
                                                ; SQRT24L2 = -4.0192034196E-1
                                0xCD
SQRT24L20
                EQU
SQRT24L21
                EQU
                                0xC8
SQRT24L22
                EQU
                                0x81
SQRT24L3
                EQU
                                0x7C
                                                ; SQRT24L3 = 1.3009144111E-1
SQRT24L30
                                0x05
                EQU
SQRT24L31
                EQU
                                0x36
SQRT24L32
                                0xB1
                EQU
                                0x79
SQRT24L4
                EQU
                                                ; SQRT24L4 = -1.8702682470E-2
SQRT24L40
                EQU
                                0x99
SQRT24L41
                EQU
                                0x36
SQRT24L42
                EQU
                                0x36
Floating Point Relation A < B
                24 bit floating point number in AEXP, AARGBO, AARGB1
                24 bit floating point number in BEXP, BARGBO, BARGB1
        Use:
                CALL
                        TALTB24
        Output: logical result in W
        Result: if A < B TRUE, W = 0x01
                if A < B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                min
                        max
                                mean
                        28
                                14.6
        Timing: 9
                                        clks
TALTB24
                MOVF
                                AARGB0,W
                                                ; test if signs opposite
                XORWF
                                BARGB0,W
                                TEMPB()
                MOVWF
                BTFSC
                                TEMPB0, MSB
                GOTO
                                TALTB240
                BTFSC
                                AARGB0,MSB
                GOTO
                                TALTB24N
TALTB24P
                MOVF
                                AEXP,W
                                                ; compare positive arguments
                SUBWF
                                BEXP,W
                BTFSS
                                _C
```

```
RETLW
                                   0x00
                 BTFSS
                                   _Z
                 RETLW
                                   0x01
                 MOVF
                                   AARGB0,W
                 SUBWF
                                   BARGB0,W
                 BTFSS
                                   _C
                 RETLW
                                   0x00
                 BTFSS
                                    _{\rm Z}
                                   0x01
                 RETLW
                 MOVF
                                   AARGB1,W
                  SUBWF
                                   BARGB1,W
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 BTFSS
                                   _Z
                 RETLW
                                   0x01
                 RETLW
                                   0x00
TALTB24N
                 MOVF
                                   BEXP,W
                                                     ; compare negative arguments
                                   AEXP,W
                  SUBWF
                  BTFSS
                                    _C
                 RETLW
                                   0x00
                                    _{\rm Z}
                 BTFSS
                 RETLW
                                   0x01
                 MOVF
                                   BARGB0,W
                                   AARGB0,W
                 SUBWF
                                   _C
                 BTFSS
                                   0x00
                 RETLW
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
                 MOVF
                                   BARGB1,W
                                   AARGB1,W
                 SUBWF
                 BTFSS
                                    _C
                                   0x00
                 RETLW
                 BTFSS
                                    _Z
                                   0x01
                 RETLW
                 RETLW
                                   0x00
TALTB240
                                   BARGB0, MSB
                 BTFSS
                                   0x01
                 RETLW
                 RETLW
                                   0x00
         Floating Point Relation A <= B
                 24 bit floating point number in AEXP, AARGBO, AARGB1
                  24 bit floating point number in BEXP, BARGBO, BARGB1
         Use:
                 CALL
                          TALEB24
;
         Output: logical result in W
         Result: if A <= B TRUE, W = 0x01
;
                 if A <= B FALSE, W = 0x00
         Testing on [-MAXNUM, MAXNUM] from 100000 trials:
;
                 min
                          max
                                   mean
         Timing: 9
                          26
                                   14.4
                                            clks
```

## **AN660**

```
TALEB24
                  MOVF
                                    AARGB0,W
                                                       ; test if signs opposite
                                    BARGB0,W
                  XORWF
                  MOVWF
                                    TEMPB0
                  BTFSC
                                    TEMPB0,MSB
                  GOTO
                                    TALEB240
                  BTFSC
                                    AARGB0, MSB
                  GOTO
                                    TALEB24N
TALEB24P
                  MOVF
                                    AEXP,W
                                                       ; compare positive arguments
                  SUBWF
                                    BEXP,W
                  BTFSS
                                     _C
                                    0x00
                  RETLW
                  BTFSS
                                     _{\rm Z}
                                    0x01
                  RETLW
                  MOVF
                                    AARGB0,W
                  SUBWF
                                    BARGB0,W
                  BTFSS
                                     _C
                  RETLW
                                    0x00
                  BTFSS
                                     _Z
                  RETLW
                                    0x01
                                    AARGB1,W
                  MOVF
                  SUBWF
                                    BARGB1,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                                    0x01
                  RETLW
TALEB24N
                                    BEXP,W
                  MOVF
                                                       ; compare negative arguments
                  SUBWF
                                    AEXP,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                     _{\rm Z}
                                    0x01
                  RETLW
                                    BARGB0,W
                  MOVF
                                    AARGB0,W
                  SUBWF
                  BTFSS
                                     _C
                  RETLW
                                    0x00
                  BTFSS
                                     _{\rm Z}
                                    0 \times 01
                  RETLW
                  MOVF
                                    BARGB1,W
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                                    0x01
                  RETLW
TALEB240
                  BTFSS
                                    BARGB0, MSB
                  RETLW
                                    0x01
                                    0x00
                  RETLW
         Floating Point Relation A > B
                  24 bit floating point number in AEXP, AARGBO, AARGB1
                  24 bit floating point number in BEXP, BARGBO, BARGB1
         Use:
                  CALL
                           TAGTB24
         Output: logical result in {\tt W}
```

```
Result: if A > B TRUE, W = 0x01
;
                if A > B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                min
                        max
                                mean
        Timing: 9
                        28
                                14.6
                                        clks
                MOVF
                                BARGB0,W
                                                ; test if signs opposite
TAGTB24
                XORWF
                                AARGB0,W
                MOVWF
                                TEMPB0
                BTFSC
                                TEMPB0,MSB
                                TAGTB240
                GOTO
                BTFSC
                                BARGB0, MSB
                GOTO
                                TAGTB24N
                                BEXP,W
TAGTB24P
                MOVF
                                                ; compare positive arguments
                               AEXP,W
                SUBWF
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                _Z
                RETLW
                                0x01
                MOVF
                                BARGB0,W
                SUBWF
                                AARGB0,W
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                _Z
                                0x01
                RETLW
                                BARGB1,W
                MOVF
                                AARGB1,W
                SUBWF
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                _Z
                                0x01
                RETLW
                RETLW
                                0x00
TAGTB24N
                MOVF
                                AEXP,W
                                                ; compare negative arguments
                SUBWF
                                BEXP,W
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                _z
                RETLW
                                0x01
                                AARGB0,W
                MOVF
                                BARGB0,W
                SUBWF
                BTFSS
                                _C
                RETLW
                                0x00
                BTFSS
                                _{\rm Z}
                                0x01
                RETLW
                MOVF
                                AARGB1,W
                SUBWF
                                BARGB1,W
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                _{\rm Z}
                RETLW
                                0 \times 01
                RETLW
                                0x00
                                AARGB0,MSB
TAGTB240
                BTFSS
                                0x01
                RETLW
                RETLW
                                0x00
```

```
*******************************
        Floating Point Relation A >= B
                24 bit floating point number in AEXP, AARGBO, AARGB1
                24 bit floating point number in BEXP, BARGBO, BARGB1
        Use:
                CALL
                         TAGEB24
        Output: logical result in W
        Result: if A >= B TRUE, W = 0x01
                if A >= B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                min
                         max
                                 mean
        Timing: 9
                         26
                                 14.4
                                          clks
                MOVF
TAGEB24
                                 BARGB0,W
                                                  ; test if signs opposite
                                 AARGB0,W
                XORWF
                MOVWF
                                 TEMPB0
                BTFSC
                                 TEMPB0,MSB
                                 TAGEB240
                GOTO
                BTFSC
                                 BARGB0, MSB
                GOTO
                                 TAGEB24N
TAGEB24P
                MOVF
                                 BEXP, W
                                                  ; compare positive arguments
                SUBWF
                                 AEXP,W
                BTFSS
                                 _C
                RETLW
                                 0x00
                BTFSS
                                 _Z
                                 0x01
                RETIM
                MOVF
                                 BARGB0,W
                                 AARGB0,W
                SUBWF
                BTFSS
                                 _C
                RETLW
                                 0x00
                BTFSS
                                 _{\rm Z}
                RETLW
                                 0x01
                MOVF
                                 BARGB1,W
                SUBWF
                                 AARGB1,W
                BTFSS
                                 _C
                                 0x00
                RETLW
                RETLW
                                 0x01
TAGEB24N
                MOVF
                                 AEXP,W
                                                  ; compare negative arguments
                SUBWF
                                 BEXP,W
                BTFSS
                                 _C
                                 0x00
                RETLW
                BTFSS
                                 _{\rm Z}
                                 0x01
                RETLW
                MOVF
                                 AARGB0,W
                                 BARGB0,W
                SUBWF
                BTFSS
                                 _C
                RETLW
                                 0x00
                BTFSS
                                 _Z
                                 0x01
                RETLW
                MOVF
                                 AARGB1,W
                SUBWF
                                 BARGB1,W
                BTFSS
                                 _C
                                 0x00
                RETLW
```

```
RETLW
                            0x01
TAGEB240
              BTFSS
                            AARGB0,MSB
              RETLW
                             0x01
              RETLW
                             0x00
Floating Point Relation A == B
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
       Use:
              CALL
                     TAEQB24
       Output: logical result in W
       Result: if A == B TRUE, W = 0 \times 01
;
              if A == B FALSE, W = 0x00
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
              min
                     max
                            mean
       Timing: 5
                     14
                            6.9
                                    clks
TAEQB24
              MOVF
                            BEXP,W
              SUBWF
                            AEXP,W
              BTFSS
                             _{\rm Z}
                             0x00
              RETLW
                            BARGB0,W
              MOVF
                            AARGB0,W
              SUBWF
              BTFSS
                             _{\rm Z}
                            0x00
              RETLW
                            BARGB1,W
              MOVF
                            AARGB1,W
              SUBWF
              BTFSS
                             _{\rm Z}
              RETLW
                             0x00
              RETLW
                             0x01
       Floating Point Relation A =! B
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
              CALL
                     TANEB24
       Use:
       Output: logical result in W
       Result: if A =! B TRUE, W = 0x01
              if A =! B FALSE, W = 0x00
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
              min
                     max
                            mean
       Timing: 5
                     14
                            6.9
                                    clks
TANEB24
              MOVF
                            BEXP,W
              SUBWF
                            AEXP,W
              BTFSS
                            _{\rm Z}
```

## **AN660**

	RETLW	0x01
	MOVF	BARGBO,W
	SUBWF	AARGB0,W
	BTFSS	_Z
	RETLW	0x01
	MOVF	BARGB1,W
	SUBWF	AARGB1,W
	BTFSS	_Z
	RETLW	0x01
	RETLW	0x00
; * * * * * * * * * * * *	******	**************

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: PIC16CXXX 32-BIT ELEMENTARY FUNCTION LIBRARY

```
RCS Header $Id: exp32.a16 1.4 1997/02/25 14:23:57 F.J.Testa Exp $
        $Revision: 1.4 $
        Evaluate expl0(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                         EXP1032
        Use:
                CALL
;
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- EXP10( AARG )
        Testing on [MINLOG10, MAXLOG10] from 100000 trials:
;
                min
                         max
                                 mean
        Timing: 3515
                         5384
                                 5001.5 clks
                min
                         max
                                 mean
                                          rms
                -0xB3
                         0x14E
                                  25.78
                                          65.54
        Error:
        This approximation of the exponential function is based upon the
        expansion
                expl0(x) = 10**x = 10**(z + n*log10(2)) = 10**z * 2**n,
;
        where -\log 10(2)/2 \ll z \ll \log 10(2)/2 and n is an integer, evaluated during
;
        range reduction. Segmented fifth degree minimax polynomial approximations
        are used to estimate 10**z on the intervals [-log10(2)/2,0] and [0,log10(2)/2].
EXP1032
                MOVLW
                                 0x5C
                                                   ; test for |x| < 2**(-32)/(2*LOG(10))
                SUBWF
                                 EXP,W
                MOVWF
                                 TEMPB0
                BTFSC
                                 TEMPB0,MSB
                                 EXP1032ONE
                 GOTO
                                                   ; return e^*x = 1
                                 AARGB0,MSB
                                                   ; determine sign
                BTFSC
                                 TNEXP1032
                 GOTO
TPEXP1032
                MOVF
                                 AEXP,W
                                                   ; positive domain check
                SUBLW
                                 MAXLOG1032EXP
                BTFSS
                                  _C
                                 DOMERR32
                GOTO
                 BTFSS
                                  Z
                 GOTO
                                 EXP1032ARGOK
                MOVF
                                 AARGB0,W
                SUBLW
                                 MAXLOG1032B0
                BTFSS
                                  _C
                 GOTO
                                 DOMERR32
                 BTFSS
                                  _{\rm Z}
                                  EXP1032ARGOK
                 COTO
```

	MOVF	AARGB1,W	
	SUBLW	MAXLOG1032B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
	0010	Em 1052mcook	
	MOVF	AARGB2,W	
	SUBLW	MAXLOG1032B2	
	BTFSS	_C	
	GOTO	DOMERR32	
	GOTO	EXP1032ARGOK	
	G010	EXFIUSZARGOR	
TNEXP1032			
111111111111111111111111111111111111111	MOVF	AEXP,W	; negative domain check
	SUBLW	MINLOG1032EXP	, negative admarii oncom
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS		
		_Z	
	GOTO	EXP1032ARGOK	
	MOVF	AARGB0,W	
	SUBLW	MINLOG1032B0	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
	MOVF	AARGB1,W	
	SUBLW	MINLOG1032B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
		33DGD0 11	
	MOVF	AARGB2,W	
	SUBLW	MINLOG1032B2	
	BTFSS	_C	
	GOTO	DOMERR32	
EVD10203DGOV			
EXP1032ARGOK	MOTTE	EDEL AGG M	A THE PART OF THE
	MOVF	FPFLAGS,W	; save RND flag
	MOVWF	DARGB3	
	DCE	EDELYCC DND	· onable rounding
	BSF	FPFLAGS, RND	; enable rounding
	CALL	RREXP1032	
	BTFSC	DARGB0,MSB	
	GOTO	EXP1032L	
	0010	EMITOSZE	
EXP1032H			
1111 103211	POL32	EXP1032H,5,4	; minimax approximation on [0,log10(2)/2]
	GOTO	EXP1032OK	
EXP1032L			
	POL32	EXP1032L,5,4	; minimax approximation on [-log10(2)/2,0]
EXP1032OK			
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	RETLW	0x00	
EXP1032ONE	MOVLW	EXPBIAS	; return 10**x = 1.0
	MOVWF	AEXP	
	CLRF	AARGB0	

```
CLRF
                               AARGB1
                CLRF
                               AARGB2
                CLRF
                                AARGB3
               RETLW
                               0x00
                               FPFLAGS,DOM
DOMERR32
               BSF
                                               ; domain error
                               0xFF
               RETLW
Range reduction routine for the exponential function
        The evaluation of z and n through the decomposition
               x = z + n*log10(2)
        is performed by first evaluating n through the relation
               n = floor(x*log2(10) + .5)
;
        The calculation of z is then obtained through a pseudo extended
        precision method
                z = x - n*log10(2) = (x - n*c1) - n*c2
        where c1 is close to log10(2) and has an exact machine representation,
        typically leading to no error in computing the term in parenthesis.
RREXP1032
               MOVF
                               AEXP,W
               MOVWF
                               CEXP
                                                ; save x in CARG
               MOVF
                               AARGB0,W
               MOVWF
                               CARGB0
                               AARGB1,W
               MOVF
               MOVWF
                               CARGB1
               MOVF
                               AARGB2,W
               MOVWF
                               CARGB2
                               AARGB0,MSB
                BSF
               MOVF
                               AARGB0,W
               MOVWF
                               BARGB0
                               AARGB1,W
               MOVF
               MOVWF
                               BARGB1
                               AARGB2,W
               MOVF
               MOVWF
                               BARGB2
                                               i = 1/\log(10) = 3.32192809489
                               0xD4
               M-TVOM
               MOVWF
                               AARGB0
               MOVLW
                                0x9A
               MOVWF
                               AARGB1
                               0x78
               MOVLW
                               AARGB2
               MOVWF
                               0x47
               MOVLW
               MOVWF
                               AARGB3
                               FXM3224U
                                               ; x * (1/log10(2))
                CALL
                INCF
                               AEXP,F
                INCF
                               AEXP,F
               BTFSC
                               AARGB0, MSB
               GOTO
                               RREXP1032YOK
               RLF
                               AARGB3,F
                RLF
                               AARGB2,F
                               AARGB1,F
               RLF
```

	RLF	AARGB0,F	
	DECF	AEXP,F	
		•	
RREXP1032YOK	DTTCC	CADCDO MCD	
KKEAP103210K	BTFSS	CARGBO, MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	MOVLW	0x7E	; k = [x / log10(2) + .5]
	MOVWF	BEXP	
	CLRF	BARGB0	
	CLRF	BARGB1	
	CLRF	BARGB2	
	CALL	FPA32	
	CALL	FLOOR32	
	CALL	I HOORSZ	
	MOVF	AEXP,W	
	MOVWF	EEXP	; save float k in EARG
	BTFSC	_Z	
	GOTO	RREXP1032FEQX	
	MOVF	AARGB0,W	
	MOVWF	EARGB0	
	MOVF	AARGB1,W	
	MOVWF	EARGB1	
	MOVF	AARGB2,W	
	MOVWF	EARGB2	
	MOVLW	0x7D	
		BEXP	
	MOVWF		. 1 201005200605
	MOVLW	0x9A	; c1 =301025390625
	MOVWF	BARGB0	
	MOVLW	0x20	
	MOVWF	BARGB1	
	CLRF	BARGB2	
	02111	2111022	
	CATT	EDM20	
	CALL	FPM32	
	MOVF	CEXP,W	
	MOVWF	BEXP	
	MOVF	CARGB0,W	
	MOVWF	BARGB0	
	MOVF	CARGB1,W	
	MOVWF	BARGB1	
	MOVF	CARGB2,W	
	MOVWF	BARGB2	
	CALL	FPA32	
	MOVF	AEXP,W	
			; save f1 in DARG
	MOVWF	DEXP	, save II III DARG
	MOVF	AARGB0,W	
	MOVWF	DARGB0	
	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	MOVF	AARGB2,W	
	MOVWF	DARGB2	
		0	
	MOVLW	0x6D	
	MOVWF	BEXP	
	MOVLW	0x9A	; c2 = 4.6050389811952113E-6
	MOVWF	BARGB0	
	MOVLW	0x84	
	MOVWF	BARGB1	
	MOVLW	0xFC	

```
MOVWF
                 BARGB2
MOVF
                 EEXP,W
MOVWF
                 AEXP
                 EARGB0,W
MOVF
MOVWF
                 AARGB0
MOVF
                 EARGB1,W
MOVWF
                 AARGB1
MOVF
                 EARGB2,W
                 AARGB2
MOVWF
CALL
                 FPM32
MOVF
                 DEXP,W
                 BEXP
MOVWF
MOVF
                 DARGB0,W
                 BARGB0
MOVWF
MOVF
                 DARGB1,W
                 BARGB1
MOVWF
                 DARGB2,W
MOVF
MOVWF
                 BARGB2
CALL
                 FPA32
MOVF
                 AEXP,W
MOVWF
                 DEXP
                                   ; save f in DARG
MOVF
                 AARGB0,W
                 DARGB0
MOVWF
MOVF
                 AARGB1,W
                 DARGB1
MOVWF
MOVF
                 AARGB2,W
MOVWF
                 DARGB2
MOVF
                 EEXP,W
MOVWF
                 AEXP
MOVF
                 EARGB0,W
MOVWF
                 AARGB0
                 EARGB1,W
MOVF
MOVWF
                 AARGB1
BCF
                 FPFLAGS, RND
                                   ; k = [x / log10(2) + .5]
CALL
                 INT2416
BSF
                 FPFLAGS, RND
MOVF
                 AARGB1,W
MOVWF
                 EARGB3
                                    ; save integer k in EARGB3
                 DEXP,W
MOVF
MOVWF
                 AEXP
                                   ; restore f in AARG
MOVF
                 DARGB0,W
MOVWF
                 AARGB0
                 DARGB1,W
MOVF
MOVWF
                 AARGB1
                 DARGB2,W
MOVF
MOVWF
                 AARGB2
RETLW
                 0x00
MOVF
                 CEXP,W
                 DEXP
MOVWF
MOVWF
                 AEXP
                                   ; save f = x in DARG, AARG
MOVF
                 CARGB0,W
MOVWF
                 DARGB0
MOVWF
                 AARGB0
                 CARGB1,W
MOVF
```

RREXP1032FEQX

	MOVWF	DARGB1	
	MOVWF	AARGB1	
	MOVF	CARGB2,W	
	MOVWF	DARGB2	
	MOVWF	AARGB2	
	110 7 112	1111022	
	CLRF	EARGB3	
	RETLW	0x00	
	112 1 2 11	01100	
;			
; fifth de	egree minimax pol	ynomial coefficie	nts for 10**(x) on [0,(log10(2))/2]
EXP1032H0	EQU	0x7F	; EXP1032H0 = 1.0
EXP1032H00	EQU	0x00	
EXP1032H01	EQU	0x00	
EXP1032H02	EQU	0x00	
	- 2 -		
EXP1032H1	EQU	0x80	; EXP1032H1 = 2.302585504840E0
EXP1032H10	EQU	0x13	
EXP1032H11	EQU	0x5D	
EXP1032H12	EQU	0x90	
EXP1032H2	EQU	0x80	; EXP1032H2 = 2.650909138708E0
EXP1032H20	EQU	0x29	
EXP1032H21	EQU	0xA8	
EXP1032H22	EQU	0x7F	
EXD1020112	EOH	00.0	• EVD1020112 0 02500000000000
EXP1032H3	EQU	0x80	; EXP1032H3 = 2.035920309947E0
EXP1032H30	EQU	0x02 0x4C	
EXP1032H31 EXP1032H32	EQU EQU	0x4C 0x85	
EXFIUSZIISZ	FÕO	0203	
EXP1032H4	EQU	0x7F	; EXP1032H4 = 1.154596329197E0
EXP1032H40	EQU	0x13	
EXP1032H41	EQU	0xC9	
EXP1032H42	EQU	0xD0	
EXP1032H5	EQU	0x7E	; EXP1032H5 = 6.388992868121E-1
EXP1032H50	EQU	0x23	
EXP1032H51	EQU	0x8E	
EXP1032H52	EQU	0xE7	
; fifth de	aree minimay nol	gnomial goefficie	nts for $10**(x)$ on $[-(log10(2))/2,0]$
, IIIcii ac	egice minimax por	ynomiai cocilicic	nes for 10 (x) on [ (log10(2///2,0]
EXP1032L0	EQU	0x7F	; EXP1032L0 = 1.0
EXP1032L00	EQU	0x00	
EXP1032L01	EQU	0x00	
EXP1032L02	EQU	0x00	
EXP1032L1	EQU	0x80	; EXP1032L1 = 2.302584716116E0
EXP1032L10	EQU	0x13	
EXP1032L11	EQU	0x5D	
EXP1032L12	EQU	0x8C	
EXP1032L2	EQU	0x80	; EXP1032L2 = 2.650914554552E0
EXP1032L20	EQU	0x29	, <u> </u>
EXP1032L21	EQU	0xA8	
EXP1032L22	EQU	0x96	
	- × ·	¥	
EXP1032L3	EQU	0x80	; EXP1032L3 = 2.033640565225E0
EXP1032L30	EQU	0x02	
EXP1032L31	EQU	0x27	
EXP1032L32	EQU	0x2B	

```
EXP1032L4
                 EOU
                                  0x7F
                                                   ; EXP1032L4 = 1.157459289066E0
EXP1032L40
                 EOU
                                  0x14
EXP1032L41
                 EQU
                                  0x27
EXP1032L42
                 EQU
                                  0xA0
EXP1032L5
                 EQU
                                  0x7D
                                                   ; EXP1032L5 = 4.544952589676E-1
EXP1032L50
                                  0x68
                 EQU
EXP1032L51
                 EQU
                                  0xB3
EXP1032L52
                 EQU
                                  0x9A
        Evaluate exp(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                 CALL
                         EXP32
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
;
        Result: AARG <-- EXP( AARG )
        Testing on [MINLOG, MAXLOG] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 16
                         5411
                               4985.0 clks
;
                 min
                         max
                                  mean
                                           rms
        Error: -0xD2
                         0xF7
                                  2.50
                                           63.99
                                                  nsb
        This approximation of the exponential function is based upon the
        expansion
                 \exp(x) = e^{*x} = e^{*x}(z + n*\log(2)) = e^{*x}z * 2**n,
;
        where -\log(2)/2 \ll z \ll \log(2)/2 and n is an integer, evaluated during
;
        range reduction. Segmented fifth degree minimax polynomial approximations
        are used to estimate e^*z on the intervals [-\log(2)/2,0] and [0,\log(2)/2].
EXP32
                 MOVLW
                                  0x5E
                                                   ; test for |x| < 2**(-32)/2
                 SUBWF
                                  EXP,W
                 MOVWF
                                  TEMPB0
                 BTFSC
                                  TEMPB0,MSB
                                  EXP32ONE
                                                   ; return e^*x = 1
                 GOTO
                 BTFSC
                                  AARGB0,MSB
                 GOTO
                                  TNEXP32
TPEXP32
                 MOVF
                                  AEXP,W
                 SUBLW
                                  MAXLOG32EXP
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
                 BTFSS
                                  _{\rm Z}
                 GOTO
                                  EXP32ARGOK
                 MOVF
                                  AARGB0,W
                                  MAXLOG32B0
                 SUBLW
                 BTFSS
                                  C
                                  DOMERR32
                 GOTO
                 BTFSS
                                  _{\rm Z}
                 GOTO
                                  EXP32ARGOK
```

	MOVF	AARGB1,W	
	SUBLW	MAXLOG32B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	0010	Bill Shintoon	
	MOVF	AARGB2,W	
	SUBLW	MAXLOG32B2	
	BTFSS	_C	
	GOTO	DOMERR32	
	GOTO	EXP32ARGOK	
מאזנייאט ט			
TNEXP32	MOVIE	YEAD M	
	MOVF	AEXP,W	
	SUBLW	MINLOG32EXP	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	MOVF	AARGB0,W	
	SUBLW	MINLOG32B0	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	G010	EXF JZAKGOK	
	MOVF	AARGB1,W	
	SUBLW	MINLOG32B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	MOVF	AARGB2,W	
	SUBLW	MINLOG32B2	
	BTFSS	_C	
	GOTO	DOMERR32	
EXP32ARGOK			
	MOVF	FPFLAGS,W	
	MOVWF	DARGB3	; save rounding flag
	BCF	FPFLAGS, RND	; disable rounding
	CALL	RREXP32	
	BTFSC	DARGB0,MSB	
	GOTO	EXP32L	
EXP32H			
	POL32	EXP32H,5,0	
		, ,	
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	RETLW	0x00	
	T(T) T T144	OAUU	
EXP32L			
1111 J211	DOI:32	EXD301 E O	
	POL32	EXP32L,5,0	
EVD220V			
EXP32OK	MOTTE	E3DGD2 ::	
	MOVF	EARGB3,W	
	ADDWF	AEXP,F	
	BTFSS	DARGB3,RND	
	RETLW	0x00	

```
BSF
                                  FPFLAGS, RND
                                                   ; restore rounding flag
                 GOTO
                                  RND4032
EXP32ONE
                                  EXPBIAS
                                                    ; return e^*x = 1.0
                 MOVLW
                 MOVWF
                                  AEXP
                 CLRF
                                  AARGB0
                 CLRF
                                  AARGB1
                 CLRF
                                  AARGB2
                 CLRF
                                  AARGB3
                 RETLW
                                  0x00
DOMERR32
                 BSF
                                  FPFLAGS, DOM
                                                   ; domain error
                 RETLW
                                  0xFF
        Range reduction routine for the exponential function
        The evaluation of z and n through the decomposition
                 x = z + n*log(2)
        is performed by first evaluating n through the relation
                 n = floor(x*log2(e) + .5)
;
        The calculation of z is then obtained through a pseudo extended
        precision method
                 z = x - n*log(2) = (x - n*c1) + n*c2
;
        where c1 is close to log(2) and has an exact machine representation,
        typically leading to no error in computing the term in parenthesis.
RREXP32
                 MOVF
                                  AEXP,W
                 MOVWF
                                  CEXP
                                                    ; save x in CARG
                 MOVF
                                  AARGB0,W
                 MOVWE
                                  CARGB0
                 MOVF
                                  AARGB1,W
                 MOVWF
                                  CARGB1
                 MOVF
                                  AARGB2,W
                 MOVWF
                                  CARGB2
                 BSF
                                  AARGB0, MSB
                 MOVF
                                  AARGB0,W
                                  BARGB0
                 MOVWF
                 MOVF
                                  AARGB1,W
                 MOVWF
                                  BARGB1
                 MOVF
                                  AARGB2,W
                                  BARGB2
                 MOVWF
                                  0xB8
                                                   i = 1.44269504089
                 MOVLW
                 MOVWF
                                  AARGB0
                 MOVLW
                                  0xAA
                                  AARGB1
                 MOVWF
                 MOVLW
                                  0x3B
                                  AARGB2
                 MOVWF
                 MOVLW
                                  0x29
                                  AARGB3
                 MOVWF
                                                   ; x * (1/ln2)
                 CALL
                                  FXM3224U
                 INCF
                                  AEXP,F
```

	DIRECO		
	BTFSC	AARGB0,MSB	
	GOTO	RREXP32YOK	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
RREXP32YOK	BTFSS	CARGB0,MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	CALL	1000 1002	
		0. 55	. 1 . 5 . / 1 . 5 . 1
	MOVLW	0x7E	; k = [x / ln2 + .5]
	MOVWF	BEXP	
	CLRF	BARGB0	
	CLRF	BARGB1	
	CLRF	BARGB2	
	CALL	FPA32	
	CALL	FFAJZ	
	CALL	FLOOR32	
	MOVF	AEXP,W	
	MOVWF	EEXP	; save float k in EARG
	BTFSC	_Z	
		RREXP32FEQX	
	GOTO		
	MOVF	AARGB0,W	
	MOVWF	EARGB0	
	MOVF	AARGB1,W	
	MOVWF	EARGB1	
	MOVF	AARGB2,W	
	MOVWF	EARGB2	
	110 V W1	Erinoba	
		0.85	
	MOVLW	0x7E	
	MOVWF	BEXP	
	MOVLW	0xB1	; c1 = .693359375
	MOVLW MOVWF	0xB1 BARGB0	; c1 = .693359375
	MOVWF	BARGB0	; c1 = .693359375
	MOVWF MOVLW	BARGB0 0x80	; c1 = .693359375
	MOVWF MOVLW MOVWF	BARGB0 0x80 BARGB1	; c1 = .693359375
	MOVWF MOVLW	BARGB0 0x80	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF	BARGB0 0x80 BARGB1 BARGB2	; c1 = .693359375
	MOVWF MOVLW MOVWF	BARGB0 0x80 BARGB1	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF	BARGB0 0x80 BARGB1 BARGB2	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF	BARGB0 0x80 BARGB1 BARGB2 FPM32	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF CALL MOVF	BARGB0 0x80 BARGB1 BARGB2 FPM32	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF CALL MOVF MOVWF	BARGB0 0x80 BARGB1 BARGB2 FPM32 CEXP,W BEXP	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF	BARGB0 0x80 BARGB1 BARGB2 FPM32 CEXP,W BEXP CARGB0,W	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF CALL MOVF MOVWF MOVF MOVF	BARGB0 0x80 BARGB1 BARGB2 FPM32 CEXP,W BEXP CARGB0,W BARGB0	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVWF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVWF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVFF MOVFF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W BARGB1 CARGB2, W BARGB2	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVFF MOVFF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W BARGB1 CARGB2, W BARGB2	; c1 = .693359375
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W BARGB1 CARGB2, W BARGB2  FPA32  AEXP, W	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP	; c1 = .693359375 ; save f1 in DARG
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP, W BEXP CARGB0, W BARGB0 CARGB1, W BARGB1 CARGB2, W BARGB2  FPA32  AEXP, W	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1 AARGB2,W DARGB1	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1	
	MOVWF MOVLW MOVWF CLRF  CALL  MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BARGB0 0x80 BARGB1 BARGB2  FPM32  CEXP,W BEXP CARGB0,W BARGB0 CARGB1,W BARGB1 CARGB2,W BARGB2  FPA32  AEXP,W DEXP AARGB0,W DARGB0 AARGB1,W DARGB1 AARGB2,W DARGB1	

```
MOVLW
                 0x5E
                                   ; c2 = .00021219444005
MOVWF
                 BARGB0
MOVLW
                 0x80
                 BARGB1
MOVWF
                 0x83
MOVLW
MOVWF
                 BARGB2
MOVF
                 EEXP,W
MOVWF
                 AEXP
                 EARGB0,W
MOVF
MOVWF
                 AARGB0
MOVF
                 EARGB1,W
MOVWF
                 AARGB1
MOVF
                 EARGB2,W
                 AARGB2
MOVWF
                 FPM32
CALL
MOVF
                 DEXP,W
MOVWF
                 BEXP
                 DARGB0,W
MOVF
MOVWF
                 BARGB0
MOVF
                 DARGB1,W
MOVWF
                 BARGB1
                 DARGB2,W
MOVF
MOVWF
                 BARGB2
CALL
                 FPA32
                 RND4032
CALL
MOVF
                 AEXP,W
                                   ; save f in DARG
                 DEXP
MOVWF
MOVF
                 AARGB0,W
MOVWF
                 DARGB0
MOVF
                 AARGB1,W
MOVWF
                 DARGB1
                 AARGB2,W
MOVF
                 DARGB2
MOVWF
MOVF
                 EEXP,W
MOVWF
                 AEXP
MOVF
                 EARGB0,W
MOVWF
                 AARGB0
MOVF
                 EARGB1,W
MOVWF
                 AARGB1
                 INT2416
                                   ; k = [x / ln2 + .5]
CALL
MOVF
                 AARGB1,W
MOVWF
                 EARGB3
                                   ; save integer k in EARGB3
MOVF
                 DEXP,W
MOVWF
                 AEXP
                                   ; restore f in AARG
MOVF
                 DARGB0,W
MOVWF
                 AARGB0
                 DARGB1,W
MOVF
MOVWF
                 AARGB1
MOVF
                 DARGB2,W
MOVWF
                 AARGB2
RETLW
                 0x00
MOVF
                 CEXP,W
                 DEXP
MOVWF
```

RREXP32FEQX

```
AEXP
                                                     ; save f = x in DARG, AARG
                  MOVWF
                                   CARGB0,W
                  MOVF
                  MOVWF
                                   DARGB0
                  MOVWF
                                   AARGB0
                  MOVF
                                   CARGB1,W
                 MOVWF
                                   DARGB1
                                   AARGB1
                 MOVWF
                  MOVF
                                   CARGB2,W
                  MOVWF
                                   DARGB2
                 MOVWF
                                   AARGB2
                  CLRF
                                   EARGB3
                  RETLW
                                   0x00
         fifth degree minimax polynomial coefficients for e^{**}(x) on [0,(ln2)/2]
EXP32H0
                  EQU
                                   0x7F
                                                     ; EXP32H0 = 1.0
                                   0x00
EXP32H00
                  EQU
EXP32H01
                  EQU
                                   0x00
EXP32H02
                  EQU
                                   0x00
EXP32H1
                  EQU
                                   0x7F
                                                     ; EXP32H1 = 1.00000025499
EXP32H10
                                   0x00
                  EQU
EXP32H11
                  EQU
                                   0x00
EXP32H12
                                   0x02
                 EQU
                                   0x7D
                                                     ; EXP32H2 = .499991163105
EXP32H2
                  EQU
EXP32H20
                  EQU
                                   0x7F
EXP32H21
                  EQU
                                   0xFE
EXP32H22
                  EQU
                                   0xD7
                                   0x7C
                                                     ; EXP32H3 = .166777360103
EXP32H3
                 EOU
EXP32H30
                 EQU
                                   0x2A
EXP32H31
                  EQU
                                   0xC7
EXP32H32
                  EQU
                                   0xAF
EXP32H4
                  EQU
                                   0x7A
                                                     ; EXP32H4 = .410473706887E-1
EXP32H40
                  EQU
                                   0x28
EXP32H41
                  EQU
                                   0x21
EXP32H42
                                   0x4A
                  EQU
EXP32H5
                  EQU
                                   0x78
                                                     ; EXP32H5 = .989943653774E-2
EXP32H50
                  EQU
                                   0x22
EXP32H51
                  EQU
                                   0x31
EXP32H52
                                   0x3F
                  EQU
        fifth degree minimax polynomial coefficients for e^{**}(x) on [-(ln2)/2,0]
EXP32L0
                                   0x7F
                                                     ; EXP32L0 = 1.0
                 EQU
EXP32L00
                  EQU
                                   0x00
EXP32L01
                                   0x00
                  EQU
EXP32L02
                 EQU
                                   0x00
EXP32L1
                                   0x7E
                                                     ; EXP32L1 = .999999766814
                 EQU
EXP32L10
                  EQU
                                   0x7F
EXP32L11
                                   0xFF
                  EQU
EXP32L12
                  EQU
                                   0xFC
EXP32L2
                  EQU
                                   0x7D
                                                     ; EXP32L2 = .499992371926
EXP32L20
                  EQU
                                   0x7F
EXP32L21
                  EQU
                                   0xFF
EXP32L22
                  EQU
                                   0x00
```

```
EXP32L3
              EOU
                             0x7C
                                            ; EXP32L3 = .166574299807
EXP32L30
              EOU
                             0x2A
EXP32L31
              EQU
                             0 \times 92
EXP32L32
              EQU
                             0x74
EXP32L4
              EQU
                             0x7A
                                            ; EXP32L4 = .411548782678E-1
EXP32L40
                             0x28
              EQU
EXP32L41
              EQU
                             0x92
EXP32L42
              EQU
                             0x05
EXP32L5
              EQU
                             0x77
                                            ; EXP32L5 = .699995870637E-2
EXP32L50
                             0x65
              EQU
EXP32L51
              EQU
                             0x5F
EXP32L52
              EQU
                             0xE9
Evaluate floor(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
;
       Use:
              CALL
                     FLOOR32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- FLOOR( AARG )
;
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                             mean
                      max
       Timing: 41
                      61
                             47.32
                                    clks
;
              min
                      max
                             mean
                                    rms
       Error: 0x00
                      0x00
                             0.0
                                    0.0
                                            nsb
       floor(x) evaluates the largest integer, as a float, not greater than x.
FLOOR32
              CLRF
                             AARGB3
                                            ; test for zero argument
              MOVF
                             AEXP,W
              BTFSC
                             _Z
                             0x00
              RETLW
              MOVF
                             AARGB0,W
                             AARGB4
              MOVWF
                                            ; save mantissa
              MOVF
                             AARGB1,W
              MOVWF
                             AARGB5
              MOVF
                             AARGB2,W
                             AARGB6
              MOVWF
                             EXPBIAS
              MOVLW
              SUBWF
                             AEXP,W
              MOVWF
                             TEMPB1
                             TEMPB1, MSB
              BTFSC
              GOTO
                             FLOOR32ZERO
              SUBLW
                             0x18-1
                                            ; save number of zero bits in TEMPBO
              MOVWF
                             TEMPB0
              MOVWF
                             TEMPB1
              BTFSC
                             TEMPB1,LSB+1+3 ; divide by eight
              GOTO
                             FLOOR32MASKH
              BTFSC
                             TEMPB1,LSB+3
```

	GOTO	FLOOR 3 2MASKM
FLOOR32MASKL		
LOOKSZMAGKE	MOVLW	0x07 ; get remainder for mask pointer
	ANDWF	TEMPBO, F
	MOVLW	LOW FLOOR32MASKTABLE
	ADDWF	TEMPBO,F
	MOVLW	HIGH FLOOR32MASKTABLE
	BTFSC	_C
	ADDLW	0x01
	MOVWF	PCLATH
	INCF	TEMPBO,W
	CALL	FLOOR32MASKTABLE; access table for mask
	ANDWF	AARGB2,F
	BTFSS	AARGBO,MSB ; if negative, round down
	RETLW	0x00
	MOVWF	AARGB7
	MOVF	AARGB6,W
	SUBWF	AARGB2,W
	BTFSS	_Z
	GOTO	FLOOR32RNDL
	RETLW	0x00
FLOOR32RNDL		
	COMF	AARGB7,W
	MOVWF	TEMPB1
	INCF	TEMPB1,W
	ADDWF	AARGB2,F
	BTFSC	_Z
	INCF	AARGB1, F
	BTFSC	_Z
	INCF	AARGBO, F
	BTFSS	_Z ; has rounding caused carryout?
	RETLW	0x00
	RRF	AARGBO,F
	RRF	AARGB1,F
	RRF	AARGB2,F
	INCFSZ	AEXP,F ; check for overflow
	RETLW	0x00
	GOTO	SETFOV32
FLOOR32MASKM		
	MOVLW	0x07 ; get remainder for mask pointer
	ANDWF	TEMPBO, F
	MOVLW	LOW FLOOR32MASKTABLE
	ADDWF	TEMPBO, F
	MOVLW	HIGH FLOOR32MASKTABLE
	BTFSC	_C
	ADDLW	0x01
	MOVWF	PCLATH
	INCF	TEMPBO,W
	CALL	FLOOR32MASKTABLE; access table for mask
	ANDWF	AARGB1,F
	CLRF	AARGB2
	BTFSS	AARGBO,MSB ; if negative, round down
	RETLW	0x00
	MOVWF	AARGB7
	MOVF	AARGB6,W
	SUBWF	AARGB2,W
	BTFSS	_Z

```
GOTO
                                    FLOOR 32RNDM
                                    AARGB5,W
                  MOVF
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    FLOOR32RNDM
                  GOTO
                  RETLW
                                    0x00
FLOOR32RNDM
                  COMF
                                    AARGB7,W
                                    TEMPB1
                  MOVWF
                  INCF
                                    TEMPB1,W
                  ADDWF
                                    AARGB1,F
                  BTFSC
                                    _{\rm Z}
                  INCF
                                    AARGB0,F
                                                      ; has rounding caused carryout?
                  BTFSS
                                    _Z
                  RETLW
                                    0x00
                  RRF
                                    AARGB0,F
                  RRF
                                    AARGB1,F
                                   AARGB2,F
                  RRF
                                    AEXP,F
                                                      ; check for overflow
                  INCFSZ
                                    0x00
                  RETLW
                  GOTO
                                    SETFOV32
FLOOR32MASKH
                  MOVLW
                                    0x07
                                                      ; get remainder for mask pointer
                  ANDWF
                                    TEMPB0,F
                  MOVLW
                                    LOW FLOOR32MASKTABLE
                  ADDWF
                                    TEMPB0,F
                  MOVLW
                                    HIGH FLOOR32MASKTABLE
                                    _C
                  BTFSC
                  ADDLW
                                    0x01
                  MOVWF
                                    PCLATH
                                    TEMPB0,W
                  INCF
                                    FLOOR32MASKTABLE; access table for mask
                  CALL
                                    AARGB0,F
                  ANDWF
                  CLRF
                                    AARGB1
                  CLRF
                                    AARGB2
                  BTFSS
                                    AARGB0,MSB
                                                      ; if negative, round down
                  RETLW
                                    0x00
                  MOVWF
                                    AARGB7
                  MOVF
                                    AARGB6,W
                  SUBWF
                                    AARGB2,W
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    FLOOR32RNDH
                                    AARGB5,W
                  MOVF
                  SUBWF
                                    AARGB1,W
                                    _Z
                  BTFSS
                  GOTO
                                    FLOOR32RNDH
                                   AARGB4,W
                  MOVF
                                    AARGB0,W
                  SUBWF
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    FLOOR32RNDH
                  RETLW
                                    0x00
FLOOR32RNDH
                                    AARGB7,W
                  COMF
                  MOVWF
                                    TEMPB1
                                    TEMPB1,W
                  INCF
                  ADDWF
                                    AARGB0,F
                  BTFSS
                                                      ; has rounding caused carryout?
                                    C
                  RETLW
                                    0x00
                  RRF
                                    AARGB0,F
                                    AARGB1,F
                  RRF
```

```
INCFSZ
                          AEXP,F
                                        ; check for overflow
             RETLW
                           0x00
             GOTO
                           SETFOV32
FLOOR32ZERO
             BTFSC
                          AARGB0,MSB
             GOTO
                          FLOOR32MINUSONE
             CLRF
                           AEXP
             CLRF
                           AARGB0
             CLRF
                           AARGB1
             CLRF
                           AARGB2
             RETLW
                           0x00
FLOOR32MINUSONE
                           0x7F
             MOVLW
             MOVWF
                          AEXP
             MOVLW
                           0x80
             MOVWF
                           AARGB0
             CLRF
                           AARGB1
                           AARGB2
             CLRF
             RETLW
                           0x00
FLOOR32MASKTABLE
             MOVWF
                           PCL
             RETLW
                           0xFF
             RETLW
                           0xFE
                           0xFC
             RETLW
             RETLW
                           0xF8
             RETLW
                           0xF0
             RETLW
                           0xE0
                           0xC0
             RETLW
                           0x80
             RETIW
             RETLW
                           0x00
Evaluate log10(x)
      Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Use:
             CALL
                   LOG1032
      Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- LOG10( AARG )
      Testing on (0,MAXNUM] from 100000 trials:
             min
                    max
                          mean
      Timing: 5208
                    5949
                          5605.7 clks
             min
                    max
                           mean
      Error: -0x96
                    0xAC
                           59.20
                                 87.50
LOG1032
             MOVF
                           FPFLAGS,W
             MOVWF
                           ZARGB0
             BSF
                           FPFLAGS, RND
             CALL
                           LOG32
                           0x7D
             MOVLW
```

```
BEXP
              MOVWF
                                            i \log 10(e) = .43429448190325
              MOVLW
                             0x5E
              MOVWF
                             BARGB0
              MOVLW
                             0x5B
              MOVWF
                             BARGB1
              MOVLW
                             0xD9
              MOVWF
                             BARGB2
                             ZARGB0, RND
              BTFSS
              BCF
                             FPFLAGS, RND
               CALL
                             FPM32
              RETLW
                             0x00
Evaluate log(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
              CALL
                     LOG32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- LOG( AARG )
       Testing on (0,MAXNUM] from 100000 trials:
                      max
                             mean
       Timing: 4797
                      5406
                             5090.6 clks
              min
                      max
                             mean
                                     rms
       Error: -0xF0
                      0x02
                             0.57
                                     1.12
                                            nsb
       This approximation of the natural log function is based upon the
       expansion
               log(x) = log(f) + log(2**n) = log(f) + n*log(2)
       where .5 \le f \le 1 and n is an integer. The additional transformation
                      2*f-1, f < 1/sqrt(2), n=n-1
;
;
                      f-1, otherwise
       produces a naturally segmented representation of log(1+z) on the
       intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
       approximations. The final evaluation of
              log(1+z) + n*log(2) = (log(1+z) - n*c2) + n*c1
       is performed in pseudo extended precision where {\tt cl} is {\tt close} to {\tt log(2)}
       and has an exact machine representation.
LOG32
               CLRF
                             AARGB3
              BTFSC
                             AARGB0,MSB
                                            ; test for negative argument
               GOTO
                             DOMERR32
              MOVF
                             AEXP,W
                                            ; test for zero argument
               BTFSC
                             _{\rm Z}
               GOTO
                             DOMERR32
```

	MOVF MOVWF	FPFLAGS,W DARGB3	; save rounding flag
	BCF	FPFLAGS,RND	; disable rounding
	MOVF	AEXP,W	
	MOVWF	EARGB3	
	MOVLW	EXPBIAS-1	
	SUBWF	EARGB3,F	
	MOVWF	AEXP	
	MOVLW	0xF3	; .70710678118655 = 7E3504F3
	SUBWF	AARGB2,W 0x04	
	MOVLW MOVWF	TEMP	
	BTFSS	_C	
	INCFSZ	TEMP,W	
	SUBWF	AARGB1,W	
	MOVLW	0x35	
	MOVWF	TEMP	
	BTFSS	_C	
	INCFSZ	TEMP,W	
	SUBWF	AARGB0,W	
		·	
	BTFSS	_C	
	GOTO	LOG32FLOW	
LOG32FHIGH	MOVLW	0x7F	
	MOVWF	BEXP	
	CLRF	BARGB0	
	CLRF	BARGB1	
	CLRF	BARGB2	
	CALL	FPS32	
	GOTO	LOGZ32OK	
LOG32FLOW	INCF	AEXP,F	
	MOVLW	0x7F	
	MOVWF	BEXP	
	CLRF	BARGB0	
	CLRF	BARGB1	
	CLRF	BARGB2	
	CALL	FPS32	
	DECF	EARGB3,F	
LOGZ320K			
<del></del>	MOVF	AEXP,W	; save z
	MOVWF	DEXP	
	MOVF	AARGB0,W	
	MOVWF	DARGB0	
	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	MOVF	AARGB2,W	
	MOVWF	DARGB2	
	POLL132	LOG32Q,2,0	; Q(z)
	MOVF	AEXP,W	
	MOVWF	CEXP	
	MOVF	AARGB0,W	
	MOVWF	CARGB0	
	MOVF	AARGB1,W	
	MOVWF	CARGB1	
	MOVF	AARGB2,W	

MOVWF	CARGB2	
MOVF	DEXP,W	
	•	
MOVWF	AEXP	
MOVF	DARGB0,W	
MOVWF	AARGB0	
MOVF	DARGB1,W	
MOVWF	AARGB1	
MOVF	DARGB2,W	
MOVWF	AARGB2	
POL32	LOG32P,1,0	; P(z)
MOVF	CEXP,W	
MOVWF	BEXP	
MOVF	CARGB0,W	
MOVWF	BARGB0	
MOVF	CARGB1,W	
MOVWF	BARGB1	
MOVF	CARGB2,W	
MOVWF	BARGB2	
CALL	FPD32	; P(z)/Q(z)
MOVF	AEXP,W	; save in CARG
MOVWF	CEXP	
MOVF	AARGB0,W	
MOVWF	CARGB0	
MOVF	AARGB1,W	
MOVWF	CARGB1	
MOVF	AARGB2,W	
MOVWF	CARGB2	
MOVF	DEXP,W	
MOVWF	BEXP	
MOVF	DARGB0,W	
MOVWF	BARGB0	
MOVF	DARGB1,W	
MOVWF	BARGB1	
MOVF	DARGB2,W	
MOVWF	BARGB2	
MOVWE	BARGBZ	
MOVF	DEXP,W	
MOVWF	AEXP	
MOVF	DARGB0,W	
MOVWF	AARGB0	
MOVF	DARGB1,W	
MOVWF	AARGB1	
MOVF	DARGB2,W	
MOVWF	AARGB2	
CALL	FPM32	; z*z
MOVF	AEXP,W	; save in EARG
MOVWF	EEXP	
MOVF	AARGB0,W	
MOVWF	EARGB0	
MOVE	AARGB1,W	
MOVWF	EARGB1	
MOVF	AARGB2,W	
MOVWF	EARGB2	
MOVF	CEXP,W	; z*z*P(z)/Q(z)
MOVWF	BEXP	. 2 2 1 (2)/2(2)
MOVE	CARGBO,W	
MOVWF	BARGB0	

```
MOVF
                  CARGB1,W
MOVWF
                  BARGB1
MOVF
                  CARGB2,W
                  BARGB2
MOVWF
CALL
                  FPM32
MOVF
                  DEXP,W
                                    ; z*(z*z*P(z)/Q(z))
MOVWF
                  BEXP
                  DARGB0,W
MOVF
MOVWF
                  BARGB0
MOVF
                  DARGB1,W
MOVWF
                  BARGB1
MOVF
                  DARGB2,W
                  BARGB2
MOVWF
                  FPM32
CALL
                  EARGB0,W
MOVF
MOVWF
                  BARGB0
                  EARGB1,W
MOVF
MOVWF
                  BARGB1
MOVF
                  EARGB2,W
MOVWF
                  BARGB2
MOVF
                  EEXP,W
                                    ; -.5*z*z + z*(z*z*P(z)/Q(z))
MOVWF
                  BEXP
BTFSS
                  _Z
DECF
                  BEXP,F
CALL
                  FPS32
CALL
                  RND4032
MOVF
                  DEXP,W
                                    z = .5*z*z + z*(z*z*P(z)/Q(z))
MOVWF
                  BEXP
MOVF
                  DARGB0,W
                  BARGB0
MOVWF
                  DARGB1,W
MOVF
                  BARGB1
MOVWF
MOVF
                  DARGB2,W
MOVWF
                  BARGB2
CALL
                  FPA32
BTFSC
                  DARGB3, RND
                  RND4032
CALL
                  EARGB3,W
MOVF
BTFSS
                  _Z
GOTO
                  ADJLOG32
RETLW
                  0x00
CALL
                  RND4032
MOVF
                  AEXP,W
                                    ; save in EARG
MOVWF
                  EEXP
MOVF
                  AARGB0,W
MOVWF
                  EARGB0
MOVF
                  AARGB1,W
                  EARGB1
MOVWF
MOVF
                  AARGB2,W
                  EARGB2
MOVWF
CLRF
                  AARGB0
                  EARGB3,W
MOVF
```

ADJLOG32

	MOVWF	AARGB1	
	BTFSC	AARGB1,MSB	
	COMF	AARGB0,F	
	CALL	FLO1624	
	CLRF	AARGB2	
	MOVF	AEXP,W	; save k in DARG
	MOVWF	DEXP	
	MOVF	AARGB0,W	
	MOVWF	DARGB0	
	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	MOVE	AARGB2,W	
	MOVWF	DARGB2	
	BSF	AARGB0,MSB	
	MOVLW	0x0D-1	; .000212194440055
	SUBWF	AEXP,F	
	MOVLW	0xDE	
	MOVWF	BARGB0	
	MOVLW	0x80	
	MOVWF	BARGB1	
	MOVLW	0x83	
	MOVWF	BARGB2	
	PIO V WI	DARODZ	
	CALL	FXM2424U	
	BTFSC	AARGB0,MSB	
	GOTO	LOG32F1OK	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
LOG32F1OK			
	BTFSC	DARGB0,MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
			7 (4 ) 1 (7 (6)
	MOVF	EEXP,W	; log(1+z) + k*log(2)
	MOVWF	BEXP	; log(1+z) + k*log(2)
		BEXP EARGB0,W	; log(1+z) + k*log(2)
	MOVWF	BEXP	; log(1+z) + k*log(2)
	MOVWF MOVF	BEXP EARGB0,W	; log(1+z) + k*log(2)
	MOVWF MOVF MOVWF	BEXP EARGB0,W BARGB0	; log(1+z) + k*log(2)
	MOVWF MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W	; log(1+z) + k*log(2)
	MOVWF MOVWF MOVWF MOVWF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1	; log(1+z) + k*log(2)
	MOVWF MOVF MOVWF MOVWF MOVWF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W	; log(1+z) + k*log(2)
	MOVWF MOVF MOVF MOVWF MOVF MOVF MOVWF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2	; log(1+z) + k*log(2)
	MOVWF MOVF MOVF MOVF MOVF MOVF CALL	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2	
	MOVWF MOVF MOVF MOVF MOVF MOVF CALL CALL MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032	; log(1+z) + k*log(2)  ; save in EARG
	MOVWF MOVF MOVF MOVF MOVF CALL CALL MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP	
	MOVWF MOVF MOVF MOVF MOVF CALL CALL MOVF MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W	
	MOVWF MOVF MOVF MOVF MOVF CALL CALL MOVF MOVF MOVF MOVF MOVWF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0	
	MOVWF MOVF MOVF MOVF MOVF CALL CALL MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W	
	MOVWF MOVF MOVF MOVF MOVF MOVF MOVWF CALL CALL MOVF MOVF MOVWF MOVF MOVF MOVWF MOVF MOVF MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W EARGB1	
	MOVWF MOVF MOVF MOVF MOVF MOVF MOVWF CALL CALL MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W EARGB1 AARGB2,W	
	MOVWF MOVF MOVF MOVF MOVF MOVF MOVWF CALL CALL MOVF MOVF MOVWF MOVF MOVF MOVWF MOVF MOVF MOVF MOVF	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W EARGB1	
	MOVWF MOVF MOVF MOVF MOVF MOVF MOVWF CALL CALL MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W EARGB1 AARGB2,W	
	MOVWF MOVF MOVF MOVF MOVF MOVF MOVWF CALL CALL MOVF MOVWF MOVWF MOVF MOVWF MOVF MOVF MOVF MOVF MOVF MOVF MOVF MOV	BEXP EARGB0,W BARGB0 EARGB1,W BARGB1 EARGB2,W BARGB2 FPA32 RND4032 AEXP,W EEXP AARGB0,W EARGB0 AARGB1,W EARGB1 AARGB2,W EARGB1 AARGB2,W EARGB2	; save in EARG

	MOVLW	$0 \times 80$	
	MOVWF	BARGB1	
	MOVF	DEXP,W	
	MOVWF	AEXP	
	MOVF	DARGBO,W	
	MOVWF	AARGB0	
	MOVF	DARGB1,W	
	MOVWF	AARGB1	
	MOVF	DARGB2,W	
	MOVWF	AARGB2	
	BSF	ANDCDO MCD	
	DOF	AARGB0,MSB	
	CALL	FXM2416U	
	BTFSC	AARGB0,MSB	
	GOTO	LOG32FOK	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
LOG32FOK			
	BTFSS	DARGB0,MSB	
	BCF	AARGB0,MSB	
	DCI	THINGDO / TIOD	
	MOTTE	DDVD M	. 7 / 1 \ . 1-47 / 0 \
	MOVF	EEXP,W	; $log(1+z) + k*log(2)$
	MOVWF	BEXP	
	MOVF	EARGB0,W	
	MOVWF	BARGB0	
	MOVF	EARGB1,W	
	MOVWF	BARGB1	
	MOVF	EARGB2,W	
	MOVWF	BARGB2	
	CALL	FPA32	
		DARGB3,RND	
	BTFSC	DARGB3,RND	
		DARGB3,RND RND4032	
DOMIND 10	BTFSC GOTO	RND4032	
DOMERR32	BTFSC GOTO BSF	RND4032 FPFLAGS,DOM	; domain error
DOMERR32	BTFSC GOTO	RND4032	; domain error
DOMERR32	BTFSC GOTO BSF	RND4032 FPFLAGS,DOM	; domain error
DOMERR32	BTFSC GOTO BSF	RND4032 FPFLAGS,DOM	; domain error
DOMERR32	BTFSC GOTO BSF	RND4032 FPFLAGS,DOM	; domain error
;	BTFSC GOTO BSF RETLW	RND4032  FPFLAGS,DOM  0xFF	
;	BTFSC GOTO BSF RETLW	RND4032  FPFLAGS,DOM  0xFF	; domain error +z*(z*z*P(z)/Q(z))
;; ; minimax	BTFSC GOTO  BSF RETLW  rational approx	RND4032  FPFLAGS,DOM 0xFF  imation z5*z*z	+z*(z*z*P(z)/Q(z))
;; minimax	BTFSC GOTO  BSF RETLW  rational approx:	RND4032  FPFLAGS,DOM 0xFF  imation z5*z*z	
;; ; minimax LOG32P0 LOG32P00	BTFSC GOTO  BSF RETLW  rational approx:	RND4032  FPFLAGS,DOM 0xFF  imation z5*z*z 0x7E 0x55	+z*(z*z*P(z)/Q(z))
;; minimax	BTFSC GOTO  BSF RETLW  rational approx:	RND4032  FPFLAGS,DOM 0xFF  imation z5*z*z	+z*(z*z*P(z)/Q(z))
;; ; minimax LOG32P0 LOG32P00	BTFSC GOTO  BSF RETLW  rational approx:	RND4032  FPFLAGS,DOM 0xFF  imation z5*z*z 0x7E 0x55	+z*(z*z*P(z)/Q(z))
;; ; minimax LOG32P0 LOG32P00 LOG32P01	BTFSC GOTO  BSF RETLW  rational approx:	RND4032  FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46	+z*(z*z*P(z)/Q(z))
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P01	BTFSC GOTO  BSF RETLW  rational approx: EQU EQU EQU EQU EQU	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1	BTFSC GOTO  BSF RETLW  rational approx: EQU EQU EQU EQU EQU EQU	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6 0x7D	+z*(z*z*P(z)/Q(z))
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P1	BTFSC GOTO  BSF RETLW  rational approx: EQU EQU EQU EQU EQU EQU EQU	RND4032  FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P10	BTFSC GOTO  BSF RETLW  rational approx: EQU	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P1	BTFSC GOTO  BSF RETLW  rational approx: EQU EQU EQU EQU EQU EQU EQU	RND4032  FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P10	BTFSC GOTO  BSF RETLW  rational approx: EQU	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P10	BTFSC GOTO  BSF RETLW  rational approx: EQU	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12	+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452
; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P11 LOG32P11	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	RND4032  FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A	+z*(z*z*P(z)/Q(z))  ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P11 LOG32P12 LOG32Q0 LOG32Q0	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F	+z*(z*z*P(z)/Q(z))  ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P1 LOG32P10 LOG32P12 LOG32Q0 LOG32Q00 LOG32Q00 LOG32Q01	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F 0xF5	+z*(z*z*P(z)/Q(z))  ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P1 LOG32P10 LOG32P11 LOG32P12 LOG32Q0 LOG32Q0	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F	+z*(z*z*P(z)/Q(z))  ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P10 LOG32P11 LOG32P12 LOG32Q0 LOG32Q00 LOG32Q00 LOG32Q00 LOG32Q001 LOG32Q002	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F 0xF5 0xC6	<pre>+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294  ; LOG32Q0 = .24993759223E1</pre>
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P10 LOG32P11 LOG32P12 LOG32Q0 LOG32Q00 LOG32Q00 LOG32Q00 LOG32Q01 LOG32Q02 LOG32Q02	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F 0xF5 0xC6	+z*(z*z*P(z)/Q(z))  ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294
;; minimax LOG32P0 LOG32P00 LOG32P01 LOG32P02 LOG32P10 LOG32P11 LOG32P12 LOG32Q0 LOG32Q00 LOG32Q00 LOG32Q00 LOG32Q001 LOG32Q002	BTFSC GOTO  BSF RETLW  rational approx:  EQU EQU EQU EQU EQU EQU EQU EQU EQU EQ	FPFLAGS, DOM 0xFF  imation z5*z*z  0x7E 0x55 0x46 0xF6  0x7D 0x79 0x12 0x8A  0x80 0x1F 0xF5 0xC6	<pre>+z*(z*z*P(z)/Q(z)) ; LOG32P0 = .83311400452  ; LOG32P1 = .48646956294  ; LOG32Q0 = .24993759223E1</pre>

```
LOG32011
                 EOU
                                   0x5F
LOG32012
                 EOU
                                   0x72
LOG32Q2
                 EQU
                                                    i LOG32Q2 = 1.0
                                   0x7F
LOG32Q20
                 EQU
                                   0x00
LOG32Q21
                 EQU
                                   0x00
LOG32Q22
                 EQU
                                   0x00
        Evaluate rand(x)
        Input: 32 bit initial integer seed in AARGBO, AARGB1, AARGB2, AARGB3
        Use:
                 CALL
                          RAND32
        Output: 32 bit random integer in AARGB0, AARGB1, AARGB2, AARGB3
        Result: AARG <-- RAND32( AARG )
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                 min
                          max
                                  mean
        Timing: 487
                          487
                                   487
                                           clks
                 min
                                   mean
                          max
        Error: 0x00
                          0x00
                                   0x00
                                           nsb
        Linear congruential random number generator
                 X \leftarrow (a * X + c) \mod m
        The calculation is performed exactly, with multiplier a, increment c, and
        modulus m, selected to achieve high ratings from standard spectral tests.
RAND32
                 MOVF
                                  RANDB0,W
                 MOVWF
                                   AARGB0
                 MOVF
                                  RANDB1,W
                                  AARGB1
                 MOVWF
                 MOVF
                                  RANDB2,W
                                  AARGB2
                 MOVWF
                 MOVF
                                  RANDB3,W
                 MOVWF
                                  AARGB3
                 MOVLW
                                   0x0D
                                                    ; multiplier a = 1664525
                 MOVWF
                                   BARGB2
                 MOVLW
                                   0x66
                                  BARGB1
                 MOVWF
                                   0x19
                 MOVLW
                 MOVWF
                                  BARGB0
                                  FXM3224U
                 CALL
                 INCF
                                  AARGB6,F
                                                    ; c = 1
                 BTFSC
                                   _{\mathsf{Z}}
                 INCF
                                   AARGB5,F
                 BTFSC
                 INCF
                                   AARGB4,F
                 BTFSC
                                   Z
                                   AARGB3,F
                 INCF
                 BTFSC
                                   _Z
                                   AARGB2,F
                 INCF
```

```
BTFSC
                                   _{\rm Z}
                                   AARGB1,F
                 INCF
                 BTFSC
                                   _{\rm Z}
                                   AARGB0,F
                 INCF
                 MOVF
                                   AARGB3,W
                                   RANDB0
                                                     i m = 2**32
                 MOVWF
                 MOVF
                                   AARGB4,W
                                   RANDB1
                 MOVWF
                                   AARGB5,W
                 MOVF
                 MOVWF
                                   RANDB2
                 MOVF
                                   AARGB6,W
                 MOVWF
                                   RANDB3
                                   0x00
                 RETLW
        Nearest neighbor rounding
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                 CALL
                          RND3224
        Use:
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- RND( AARG )
        Testing on [MINNUM, MAXNUM] from 10000 trials:
                 min
                          max
                                   mean
        Timing: 3
                                            clks
                          17
                 min
                          max
                                   mean
        Error: 0
                          0
                                            nsb
RND3224
                 BTFSS
                                   AARGB2,MSB
                                                     ; is NSB < 0x80?
                 RETLW
                                   0x00
                 BSF
                                   С
                                                     ; set carry for rounding
                                   0x7F
                 MOVLW
                 ANDWF
                                   AARGB2,W
                 BTFSC
                                   _{\rm Z}
                                   AARGB1,W
                 RRF
                                                     ; select even if NSB = 0x80
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   SIGN
                                                     ; save sign
                                   AARGB0,MSB
                                                     ; make MSB explicit
                 BSF
                                   _z
                 BCF
                 BTFSC
                                   _C
                                                     ; round
                                   AARGB1,F
                 INCF
                 BTFSC
                                   _{\rm Z}
                 INCF
                                   AARGB0,F
                 BTFSS
                                   _{\rm Z}
                                                     ; has rounding caused carryout?
                                   RND32240K
                 GOTO
                 RRF
                                   AARGB0,F
                                                     ; if so, right shift
                                   AARGB1,F
                 RRF
                 INCF
                                   EXP,F
                                                     ; test for floating point overflow
                 BTFSC
                                   _{\rm Z}
                                   SETFOV24
                 GOTO
```

```
RND32240K
                 BTFSS
                                   SIGN, MSB
                 BCF
                                   AARGB0,MSB
                                                   ; clear sign bit if positive
                 RETLW
                                   0x00
        Nearest neighbor rounding
        Input: 40 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2, AARGB3
        Use:
                 CALL
                          RND4032
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- RND( AARG )
        Testing on [MINNUM, MAXNUM] from 10000 trials:
                 min
                          max
                                  mean
        Timing: 3
                          17
                                           clks
                 min
                          max
                                   mean
        Error: 0
                                           nsb
                          0
RND4032
                                  AARGB3,MSB
                                                  ; is NSB < 0x80?
                 BTFSS
                 RETLW
                                   0x00
                                   _C
                                                    ; set carry for rounding
                 BSF
                                   0x7F
                 M.TVOM
                                  AARGB3,W
                 ANDWF
                 BTFSC
                                   _{\rm Z}
                                                    ; select even if NSB = 0x80
                 RRF
                                  AARGB2,W
                 MOVF
                                  AARGB0,W
                                                    ; save sign
                 MOVWF
                                   SIGN
                 BSF
                                  AARGB0,MSB
                                                    ; make MSB explicit
                 BCF
                                   _Z
                 BTFSC
                                   С
                                                    ; round
                                  AARGB2,F
                 INCF
                 BTFSC
                                   _{\rm Z}
                                  AARGB1,F
                 INCF
                 BTFSC
                                   _{\rm Z}
                 INCF
                                  AARGB0,F
                 BTFSS
                                   _{\rm Z}
                                                    ; has rounding caused carryout?
                                  RND40320K
                 GOTO
                 RRF
                                  AARGB0,F
                                                    ; if so, right shift
                 RRF
                                  AARGB1,F
                 RRF
                                  AARGB2,F
                 INCF
                                  EXP,F
                                                    ; test for floating point overflow
                 BTFSC
                                   _{\rm Z}
                 GOTO
                                   SETFOV32
RND40320K
                 BTFSS
                                   SIGN, MSB
                                                    ; clear sign bit if positive
                 BCF
                                  AARGB0,MSB
                 RETLW
                                   0x00
```

```
Evaluate cos(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                CALL
                         COS32
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- COS( AARG )
        Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
                min
                         max
                                 mean
        Timing: 3568
                         6098
                               5545.9 clks
                min
                         max
                                 mean
                                          rms
        Error: -0x225 0x1E5
                                -10.42 98.36
                                                  nsb
        The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition z = x \mod pi/4, with an additional variable j
        indicating the correct octant, leading to the appropriate call
        to either the sine or cosine approximations
        \sin(z) = z * (z**2) * p(z**2), \cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
        where p and q are minimax polynomial approximations.
COS32
                MOVF
                                 FPFLAGS, W
                                                  ; save rounding flag
                MOVWF
                                 DARGB3
                BSF
                                 FPFLAGS, RND
                                                  ; enable rounding
                 CLRF
                                 CARGB3
                                                  ; initialize sign in CARGB3
                 BCF
                                 AARGB0,MSB
                                                  ; use |x|
                                 RRSINCOS32
                 CALL
                                                  ; range reduction
RRCOS320K
                 RRF
                                 EARGB3,W
                 XORWF
                                 EARGB3,W
                                 TEMPB0
                MOVWF
                 BTFSC
                                 TEMPB0,LSB
                 GOTO
                                 COSZSIN32
                 CALL
                                 ZCOS32
                 GOTO
                                 COSSIGN32
COSZSIN32
                 CALL
                                 ZSIN32
COSSIGN32
                 MOVLW
                                 0x80
                 BTFSC
                                 EARGB3,LSB+1
                 XORWF
                                 CARGB3,F
                 BTFSC
                                 CARGB3, MSB
                                 AARGB0,F
                 XORWF
                                 DARGB3, RND
                 BTFSS
```

```
RETLW
                               0x00
               BSF
                               FPFLAGS, RND
                                              ; restore rounding flag
               CALL
                               RND4032
               RETLW
                               0x00
Evaluate sin(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Use:
               CALL
                       SIN32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- SIN( AARG )
       Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
               min
                       max
                               mean
       Timing: 4030
                       6121
                               5545.7 clks
               min
                       max
                               mean
                                       rms
       Error: -0x22D 0x1F1
                               -9.55
                                       97.87
                                             nsb
;
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
;
       the definition z = x \mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
       \sin(z) = z * (z**2) * p(z**2), \cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
       where p and q are minimax polynomial approximations.
SIN32
                               FPFLAGS,W
               MOVF
                                               ; save rounding flag
               MOVWF
                               DARGB3
               BSF
                               FPFLAGS, RND
                                               ; enable rounding
               CLRF
                               CARGB3
                                               ; initialize sign in CARGB3
               BTFSC
                               AARGB0, MSB
               BSF
                               CARGB3, MSB
               BCF
                               AARGB0,MSB
                                               ; use |x|
               CALL
                               RRSINCOS32
                                               ; range reduction
RRSIN320K
               RRF
                               EARGB3,W
               XORWF
                               EARGB3,W
               MOVWF
                               TEMPB0
                               TEMPB0, LSB
               BTFSC
               GOTO
                               SINZCOS32
               CALL
                               ZSIN32
               GOTO
                               SINSIGN32
SINZCOS32
               CALL
                               ZCOS32
```

```
SINSIGN32
               MOVLW
                               0x80
                               CARGB3,MSB
               BTFSC
               XORWF
                               AARGB0,F
               BTFSS
                               DARGB3, RND
               RETLW
                               0x00
               BSF
                               FPFLAGS, RND
                                             ; restore rounding flag
               CALL
                               RND4032
               RETLW
                               0x00
Evaluate sin(x) and cos(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
               CALL
                     SINCOS32
       Use:
       Output: 32 bit floating point cos(x) in AEXP, AARGBO, AARGB1, AARGB2 and
               sin(x) BEXP, BARGBO, BARGB1, BARGB2
       Result: AARG <-- COS( AARG )
               BARG <-- SIN( AARG )
       Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
               min
                       max
                               mean
       Timing: 6611
                       8858
                              8382.6 clks
                       max
                               mean
                                       rms
       Error: -0x225 0x1E5
                               -10.42 98.36
                                               nsb
                                                      cos(x)
                               -9.55
               -0x22D
                       0x1F1
                                       97.87
                                                      sin(x)
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x \mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
       sin(z) = z * (z**2) * p(z**2), cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
       where p and q are minimax polynomial approximations. In this case,
       only one range reduction is necessary.
SINCOS32
               MOVF
                               FPFLAGS, W
                                              ; save rounding flag
               MOVWF
                               DARGB3
               BSF
                               FPFLAGS, RND
                                              ; enable rounding
               MOVF
                               AEXP,W
                                              ; save x in EARG
               MOVWF
                               EEXP
               MOVF
                               AARGB0,W
                               EARGB0
               MOVWF
               MOVF
                               AARGB1,W
               MOVWF
                               EARGB1
               MOVF
                               AARGB2,W
               MOVWF
                               EARGB2
               BCF
                               AARGB0,MSB
                                               ; use |x|
               CLRF
                               CARGB3
                                               ; initialize sign in CARGB3
```

```
CALL
                          RRSINCOS32
                                           ; range reduction
        MOVF
                          CARGB3,W
                                           ; save sign from range reduction
        MOVWF
                          ZARGB2
        MOVLW
                          0x80
        BTFSC
                          EARGB0, MSB
                                           ; toggle sign if x < 0
        XORWF
                          CARGB3,F
                          RRSIN320K
        CALL
        MOVF
                          AEXP,W
                                           ; save sin(x) in EARG
        MOVWF
                          EEXP
        MOVF
                          AARGB0,W
        MOVWF
                          EARGB0
        MOVF
                          AARGB1,W
        MOVWF
                          EARGB1
        MOVF
                          AARGB2,W
                          EARGB2
        MOVWF
                          AARGB3,W
        MOVF
                          ZARGB3
        MOVWF
        BSF
                          FPFLAGS, RND
                                           ; enable rounding
        MOVF
                          DEXP,W
                                           ; restore z*z in AARG
                          AEXP
        MOVWF
        MOVF
                          DARGB0,W
                          AARGB0
        MOVWF
        MOVF
                          DARGB1,W
                          AARGB1
        MOVWF
        MOVF
                          DARGB2,W
        MOVWF
                          AARGB2
        MOVF
                          ZARGB2,W
                                           ; restore sign from range reduction
        MOVWF
                          CARGB3
                          RRCOS320K
        CALL
                          EEXP,W
        MOVF
                                           ; restore sin(x) in BARG
                          BEXP
        MOVWF
        MOVF
                          EARGB0,W
        MOVWF
                          BARGB0
                          EARGB1,W
        MOVF
        MOVWF
                          BARGB1
                          EARGB2,W
        MOVF
        MOVWF
                          BARGB2
        MOVF
                          ZARGB3,W
                          BARGB3
        MOVWF
        RETLW
                          0x00
Range reduction routine for trigonometric functions
The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
alternative trigonometric argument z on [-pi/4,pi/4], through
the definition
        z = x \mod pi/4,
produced by first evaluating y and j through the relations
```

y = floor(x/(pi/4)), j = y - 8\*[y/8].

;

;

```
where j equals the correct octant. For j odd, adding one to j
        and y eliminates the odd octants. Additional logic on j and the
        sign of the result leads to appropriate use of the sine or cosine
        routine in each case.
        The calculation of z is then obtained through a pseudo extended
        precision method
        z = x \mod pi/4 = x - y*(pi/4) = (((x - p1*y)-p2*y)-p3*y)-p4*y
        where pi/4 = p1 + p2 + p3 + p4, with p1 close to pi/4, p2 close to
        pi/4 - p1, and p3 close to pi/4 - p1 - p2. The numbers p1, p2 and p3
        are chosen to have an exact machine representation with slightly more
        than the lower half of the mantissa bits zero, typically leading to no
        error in computing the terms in parenthesis. This calculation breaks
        down leading to a loss of precision for |x| > LOSSTHR = sqrt(2**24)*pi/4,
        or for |x| close to an integer multiple of pi/4. This loss threshold has
        been chosen based on the efficacy of this calculation, with a domain error
        reported if this threshold is exceeded.
RRSINCOS32
                 MOVF
                                  AEXP,W
                                                    ; loss threshold check
                 SUBLW
                                  LOSSTHR32EXP
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
                 BTFSS
                                  _{\rm Z}
                 GOTO
                                  RRSINCOS32ARGOK
                 MOVF
                                  AARGBO.W
                 SUBLW
                                  LOSSTHR32B0
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
                 BTFSS
                                  _{\rm Z}
                                  RRSINCOS32ARGOK
                 COTO
                 MOVF
                                  AARGB1,W
                 SUBLW
                                  LOSSTHR32B1
                 BTFSS
                                  C
                 GOTO
                                  DOMERR32
                 BTFSS
                                  Z
                 GOTO
                                  RRSINCOS32ARGOK
                                  AARGB2,W
                 MOVF
                 SUBLW
                                  LOSSTHR32B2
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
RRSINCOS32ARGOK
                 MOVF
                                  AEXP,W
                 MOVWF
                                  CEXP
                                                    ; save |x| in CARG
                 MOVF
                                  AARGB0,W
                                  CARGB0
                 MOVWF
                 MOVE
                                  AARGB1.W
                 MOVWF
                                  CARGB1
                 MOVF
                                  AARGB2,W
                 MOVWF
                                  CARGB2
        fixed point multiplication by 4/pi
                 BSF
                                  AARGB0, MSB
                 MOVF
                                  AARGB0,W
                 MOVWF
                                  BARGB0
                                  AARGB1,W
                 MOVF
                                  BARGB1
                 MOVWF
                 MOVF
                                  AARGB2,W
                                  BARGB2
                 MOVWF
```

	MOVLW	0xA2	; 4/pi = 1.27323954474
	MOVWF	AARGB0	
	MOVLW	0xF9	
	MOVWF	AARGB1	
	MOVLW	0x83	
	MOVWF	AARGB2	
	MOVLW	0x6E	
	MOVWF	AARGB3	
	110 V W1	THRODS	
	CALL	FXM3224U	
	INCF	AEXP,F	
	11101	111111 / 1	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS32YOK	
	RLF	AARGB3,F	
	RLF	AARGB2,F	
	RLF	AARGB1,F	
	RLF	AARGB0,F	
	DECF	AEXP,F	
	DECI	ABAL , L	
RRSINCOS32YOK			
	BCF	AARGB0,MSB	
	BCF	FPFLAGS, RND	
	CALL	INT3224	; y = [ x  * (4/pi)]
			, À - [  X  (4/bī) ]
	BSF	FPFLAGS,RND	
	BTFSS	AARGB2,LSB	
	GOTO	SAVEY32	
	0010	011111111111111111111111111111111111111	
	TMOR	77DCD0 E	
	INCF	AARGB2,F	
	BTFSC	_Z	
	INCF	AARGB1,F	
	BTFSC	_Z	
	INCF	AARGB0,F	
SAVEY32	MOVF	AARGB0,W	
	MOVWF	DARGB0	; save y in DARG
	MOVF	AARGB1,W	
		DARGB1	
	MOVWF		
	MOVE	AARGB2,W	
	MOVWF	DARGB2	
	MOVLW	0x07	$; j = y \mod 8$
	ANDWF	AARGB2,F	
	MOVLW	0x03	
	SUBWF	AARGB2,W	
	MOVII W	0.200	
	MOVLW	0x80	
	BTFSS	_C	
	GOTO	JOK32	
	XORWF	CARGB3,F	
	MOVLW	$0 \times 04$	
	SUBWF	AARGB2,F	
ЈОК32	MOVF	AARGB2,W	
	MOVWF	EARGB3	; save j in EARGB3
			-
	MOVF	DARGB0,W	
	MOVWF	AARGB0	; restore y to AARG
	MOVF	DARGB1,W	
	MOVWF	AARGB1	
	MOVF	DARGB2,W	

```
MOVWF
                                    AARGB2
                  CALL
                                    FLO2432
                  MOVF
                                    AEXP,W
                 MOVWF
                                    DEXP
                                                      ; save y in DARG
                  BTFSC
                                    _{\rm Z}
                  GOTO
                                    RRSINCOS32ZEQX
                  MOVF
                                    AARGB0,W
                  MOVWF
                                    DARGB0
                  MOVF
                                    AARGB1,W
                  MOVWF
                                    DARGB1
                  MOVF
                                    AARGB2,W
                  MOVWF
                                    DARGB2
        Cody-Waite extended precision calculation of |x| - y * pi/4 using
        fixed point multiplication. Since y >= 1, underflow is not possible
        in any of the products.
                  BSF
                                    AARGB0,MSB
                  MOVLW
                                    0xC9
                                                      ; - p1 = -.78515625
                  MOVWF
                                    BARGB0
                                    BARGB1
                  CLRF
                  CALL
                                    FXM2416U
                                    AARGB0,MSB
                  BTFSC
                  GOTO
                                    RRSINCOS32Z1OK
                  RLF
                                    AARGB3,F
                  RLF
                                    AARGB2,F
                  RLF
                                    AARGB1,F
                                    AARGB0,F
                  RLF
                  DECF
                                    AEXP,F
RRSINCOS32Z1OK
                                    CEXP,W
                  MOVF
                                                      ; restore \mathbf{x} to BARG
                  MOVWF
                                    BEXP
                  MOVF
                                    CARGB0,W
                  MOVWF
                                    BARGB0
                  MOVF
                                    CARGB1,W
                  MOVWF
                                    BARGB1
                 MOVF
                                    CARGB2,W
                 MOVWF
                                    BARGB2
                                                      ; z1 = |x| - y * (p1)
                  CALL
                                    FPA32
                  MOVF
                                    AEXP,W
                  MOVWF
                                    CEXP
                                                      ; save z1 in CARG
                  MOVF
                                    AARGB0,W
                  MOVWF
                                    CARGB0
                                    AARGB1,W
                 MOVF
                  MOVWF
                                    CARGB1
                                    AARGB2,W
                 MOVF
                  MOVWF
                                    CARGB2
                 MOVF
                                    DEXP,W
                  MOVWF
                                    AEXP
                  MOVF
                                    DARGB0,W
                  MOVWF
                                    AARGB0
                                                      ; restore y to AARG
                                    DARGB1,W
                 MOVF
                  MOVWF
                                    AARGB1
                 MOVF
                                    DARGB2,W
                  MOVWF
                                    AARGB2
                                    AARGB0,MSB
                  BSF
```

```
0xFD
                                                     ; - p2 = -.00024187564849853515624
                 MOVLW
                 MOVWF
                                   BARGB0
                                   0xA0
                 MOVLW
                                   BARGB1
                 MOVWF
                 CALL
                                   FXM2416U
                                   0x0D - 1
                 MOVLW
                 BTFSC
                                   AARGB0,MSB
                 GOTO
                                   RRSINCOS32Z2OK
                 RLF
                                   AARGB3,F
                                   AARGB2,F
                 RLF
                                   AARGB1,F
                 RLF
                 RLF
                                   AARGB0,F
                                   AEXP,F
                 DECF
RRSINCOS32Z2OK
                 SUBWF
                                   AEXP,F
                 MOVF
                                   CEXP,W
                                                     ; restore z1 to BARG
                 MOVWF
                                   BEXP
                                   CARGB0,W
                 MOVF
                 MOVWF
                                   BARGB0
                 MOVF
                                   CARGB1,W
                 MOVWF
                                   BARGB1
                                   CARGB2,W
                 MOVF
                 MOVWF
                                   BARGB2
                 CALL
                                   FPA32
                                                     ; z2 = z1 - y * (p2)
                                   AEXP,W
                 MOVF
                                   CEXP
                                                     ; save z2 in CARG
                 MOVWE
                 MOVF
                                   AARGB0,W
                 MOVWF
                                   CARGB0
                 MOVF
                                   AARGB1,W
                 MOVWF
                                   CARGB1
                 MOVF
                                   AARGB2,W
                 MOVWF
                                   CARGB2
                                   DEXP,W
                 MOVF
                                   AEXP
                 MOVWF
                 MOVF
                                   DARGB0,W
                 MOVWF
                                   AARGB0
                                                     ; restore y to AARG
                 MOVF
                                   DARGB1,W
                 MOVWF
                                   AARGB1
                                   DARGB2,W
                 MOVF
                 MOVWF
                                   AARGB2
                 BSF
                                   AARGB0, MSB
                 MOVLW
                                   0xA2
                                                     i - p3 = -3.7747668102383613583E-8
                                   BARGB0
                 MOVWF
                 MOVLW
                                   0x20
                 MOVWF
                                   BARGB1
                 CALL
                                   FXM2416U
                 MOVLW
                                   0x19 - 1
                 BTFSC
                                   AARGB0,MSB
                                   RRSINCOS32Z3OK
                 GOTO
                 RLF
                                   AARGB3,F
                 RLF
                                   AARGB2,F
                                   AARGB1,F
                 RLF
```

	RLF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32Z3OK			
RK51NCO53223OK	GT-D-T-		
	SUBWF	AEXP,F	
	MOVF	CEXP,W	; restore z2 to BARG
	MOVWF	BEXP	
	MOVF	CARGB0,W	
	MOVWF	BARGB0	
	MOVF	CARGB1,W	
	MOVWF	BARGB1	
	MOVF	CARGB2,W	
	MOVWF	BARGB2	
	110 V W1	Dincoba	
	G3.7.7		. 2 0 1 (2)
	CALL	FPA32	; z3 = z2 - y * (p3)
	MOVF	AEXP,W	
	MOVWF	CEXP	; save z3 in CARG
	MOVF	AARGB0,W	
	MOVWF	CARGB0	
	MOVF	AARGB1,W	
	MOVWF	CARGB1	
	MOVF	AARGB2,W	
	MOVWF	CARGB2	
	MOVWE	CARGBZ	
	MOVF	DEXP,W	
	MOVWF	AEXP	
	MOVF	DARGB0,W	
	MOVWF	BARGB0	; restore y to BARG
			/ Testore y to bard
	MOVF	DARGB1,W	
	MOVWF	BARGB1	
	MOVF	DARGB2,W	
	MOVWF	BARGB2	
	BSF	BARGB0,MSB	
	BSF	BARGBO, MBB	
		0. 7.4	. 4 2 554004055445056265
	MOVLW	0xB4	i - p4 = -3.77489497744597636E-8
	MOVWF	AARGB0	
	MOVLW	0x61	
	MOVWF	AARGB1	
	MOVLW	0x1A	
	MOVWF	AARGB2	
	MOVLW	0x63	
	MOVWF	AARGB3	
	CALL	FXM3224U	
		- <del>-</del>	
	MONT M	020 1	
	MOVLW	0x28 - 1	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS32Z4OK	
	RLF	AARGB4,F	
	RLF	AARGB4,F	
	RLF RLF	AARGB3,F	
	RLF RLF RLF	AARGB3,F AARGB2,F	
	RLF RLF	AARGB3,F AARGB2,F AARGB1,F	
	RLF RLF RLF	AARGB3,F AARGB2,F	
	RLF RLF RLF RLF RLF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
	RLF RLF RLF	AARGB3,F AARGB2,F AARGB1,F	
RRSINCOS32740K	RLF RLF RLF RLF RLF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
RRSINCOS32Z40K	RLF RLF RLF RLF DECF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	
RRSINCOS32Z40K	RLF RLF RLF RLF RLF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F	
RRSINCOS32Z40K	RLF RLF RLF RLF DECF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	
RRSINCOS32Z4OK	RLF RLF RLF RLF DECF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	
RRSINCOS32Z4OK	RLF RLF RLF RLF DECF SUBWF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  AEXP,F	
RRSINCOS32Z4OK	RLF RLF RLF RLF DECF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F	; restore z3 to BARG
RRSINCOS32Z4OK	RLF RLF RLF RLF DECF SUBWF	AARGB3,F AARGB2,F AARGB1,F AARGB0,F AEXP,F  AEXP,F	; restore z3 to BARG

	MOVF	CARGB0,W	
	MOVWF	BARGB0	
	MOVF	CARGB1,W	
		BARGB1	
	MOVWF		
	MOVF	CARGB2,W	
	MOVWF	BARGB2	
	CALL	FPA32	z = z3 - y * (p4)
RRSINCOS320K			
	MOVF	AEXP,W	
			; save z in CARG
	MOVWF	CEXP	, save z In CARG
	MOVF	AARGB0,W	
	MOVWF	CARGB0	
	MOVF	AARGB1,W	
	MOVWF	CARGB1	
	MOVF	AARGB2,W	
	MOVWF	CARGB2	
	MOVWE	CARGDZ	
	MOTTE	A PARD A	
	MOVF	AEXP,W	
	MOVWF	BEXP	
	MOVF	AARGB0,W	
	MOVWF	BARGB0	
	MOVF	AARGB1,W	
	MOVWF		
		BARGB1	
	MOVF	AARGB2,W	
	MOVWF	BARGB2	
	CALL	FPM32	
	MOVF	AEXP,W	
			; save z * z in DARG
	MOVWF	DEXP	/ Save Z Z III DARG
	MOVF	AARGB0,W	
	MOVWF	DARGB0	
	MOVF	AARGB1,W	
	MOVWF	DARGB1	
	MOVF	AARGB2,W	
	MOVWF	DARGB2	
	RETLW	0x00	
RRSINCOS32ZEQX			
	MOVF	CEXP,W	
	MOVWF	AEXP	
	MOVE	CARGBO,W	
	MOVWF	AARGB0	
	MOVF	CARGB1,W	
	MOVWF	AARGB1	
	MOVF	CARGB2,W	
	MOVWF	AARGB2	
	MOVF	AEXP,W	
	MOVWF	BEXP	
	MOVF	AARGB0,W	
	MOVWF	BARGB0	
	MOVF	AARGB1,W	
	MOVWF	BARGB1	
	MOVF	AARGB2,W	
	MOVWF	BARGB2	
	CATT	EDM20	+ _
	CALL	FPM32	; z * z
	MOTTE	3 EUO ***	
	MOVF	AEXP,W	
	MOVWF	DEXP	; save z * z in DARG
	MOVF	AARGB0,W	
	MOVWF	DARGB0	

	MOVF	AARGB1,W
	MOVWF	DARGB1
	MOVF	AARGB2,W
	MOVWF	DARGB2
	RETLW	0x00
DOMERR32	BSF	FPFLAGS,DOM ; domain error
DOMERICOZ	RETLW	0xFF
	KBIBW	ONLI
; * * * * * * * * * * * * * * *	******	****************
ZCOS32	POL32	COS32D,2,1
	MOVF	DEXP, W
	MOVWF	BEXP
	MOVF	DARGBO,W
	MOVWF	BARGB0
	MOVF	DARGB1,W
	MOVWF	BARGB1
	MOVF	
		DARGB2,W
	MOVWF	BARGB2
	CALL	FPM32
	MOVIE	מעקת ע
	MOVF	DEXP, W
	MOVWF	BEXP
	MOVF	DARGBO, W
	MOVWF	BARGB0
	MOVF	DARGB1,W
	MOVWF	BARGB1
	MOVF	DARGB2,W
	MOVWF	BARGB2
	CALL	FPM32
	MOVF	DEXP, W
	MOVWF	BEXP
	MOVF	DARGBO, W
	MOVWF	BARGBO
	MOVF	DARGB1,W
	MOVWF	BARGB1
	MOVF	DARGB2,W
	MOVWF	BARGB2
	DECF	BEXP,F
	CALL	FPS32
	MOVLW	EXPBIAS
	MOVWF	BEXP
	CLRF	BARGB0
	CLRF	BARGB1
	CLRF	BARGB2
	BCF	FPFLAGS, RND
	CALL	FPA32
	- <del></del>	
	RETLW	0x00
ZSIN32		
70TIN27	POL32	SIN32D,3,1
	. 01102	01M389   0   1
	MOVF	DEXP, W
	MOVWF	BEXP
	MOVF	DARGBO, W
	MOVWF	BARGBO

```
MOVF
                                   DARGR1.W
                 MOVWF
                                   BARGB1
                 MOVF
                                   DARGB2,W
                 MOVWF
                                   BARGB2
                 CALL
                                   FPM32
                 MOVF
                                   CEXP, W
                 MOVWF
                                   BEXP
                                   CARGB0,W
                 MOVF
                                   BARGB0
                 MOVWF
                                   CARGB1,W
                 MOVF
                 MOVWF
                                   BARGB1
                 MOVF
                                   CARGB2,W
                                   BARGB2
                 MOVWF
                 CALL
                                   FPM32
                 MOVF
                                   CEXP, W
                                   BEXP
                 MOVWF
                 MOVF
                                   CARGB0,W
                 MOVWF
                                   BARGB0
                 MOVF
                                   CARGB1,W
                                   BARGB1
                 MOVWF
                                   CARGB2,W
                 MOVF
                 MOVWF
                                   BARGB2
                 BCF
                                   FPFLAGS, RND
                 CALL
                                   FPA32
                 RETLW
                                   0x00
        minimax polynomial coefficients for sin(z) = z+z*(z**2)*p(z**2) on [-pi/4,pi/4]
SIN32D0
                 EQU
                                   0x7C
                                                    ; SIN32D0 = -1.66666664079712E-1
SIN32D00
                 EQU
                                   OxAA
SIN32D01
                                   0xAA
                 EQU
SIN32D02
                 EQU
                                   0xAB
SIN32D1
                 EQU
                                   0x78
                                                    ; SIN32D1 = 8.333329304850749E-3
SIN32D10
                 EQU
                                   0x08
SIN32D11
                                   0x88
                 EQU
SIN32D12
                 EQU
                                   0x84
                                                    ; SIN32D2 = -1.983931227180460E-4
SIN32D2
                                   0x72
                 EQU
SIN32D20
                 EQU
                                   0xD0
SIN32D21
                 EQU
                                   0x07
SIN32D22
                 EQU
                                   0xC0
SIN32D3
                 EQU
                                   0x6C
                                                    ; SIN32D3 = 2.718121647219611E-6
SIN32D30
                 EQU
                                   0x36
SIN32D31
                 EQU
                                   0x68
SIN32D32
                 EQU
                                   0xF9
        minimax polynomial coefficients for cos(z) = 1 - .5*z**2 + z**4*q(z**2)
        on [-pi/4,pi/4]
COS32D0
                                   0x7A
                                                    ; COS32D0 = 4.166664568297614E-2
                 EQU
COS32D00
                                   0x2A
                 EQU
COS32D01
                 EQU
                                   0xAA
COS32D02
                 EQU
                                   0xA5
```

```
0x75
COS32D1
                EOU
                                                  ; COS32D1 = -1.388731625438419E-3
COS32D10
                EQU
                                 0xB6
COS32D11
                EQU
                                 0x06
COS32D12
                EQU
                                 0x1A
COS32D2
                EQU
                                 0x6F
                                                 ; COS32D2 = 2.443315706066392E-5
COS32D20
                EQU
                                 0x4C
COS32D21
                EQU
                                 0xF5
COS32D22
                                 OxCE
                EOU
        Evaluate sqrt(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                CALL
                        SQRT32
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- SQRT( AARG )
        Testing on [0,MAXNUM] from 100000 trials:
                min
                        max
                                mean
        Timing: 7
                        4966
                              4290.2 clks
                min
                        max
                                 mean
                                         rms
        Error: -0xC7
                        0xDF
                                 -15.18 37.95
                                                nsb
        Range reduction for the square root function is naturally produced by
        the floating point representation,
                x = f * 2**e, where 1 <= f < 2,
        leading to the expression
                            sqrt(f) * 2**(e/2),e even
                             | sqrt(f) * sqrt(2) * 2**(e/2),e odd
        With f=1+z, the function sqrt(1+z) is then approximated by a
        minimax rational function on the interval [0,1].
SQRT32
                BTFSC
                                 AARGB0,MSB
                                               ; test for negative argument
                GOTO
                                 DOMERR32
                                 AARGB3
                CLRF
                                                  ; return if argument zero
                MOVF
                                 AEXP,W
                BTFSC
                                 _{\rm Z}
                RETLW
                                 0x00
                                                 ; save exponent in CEXP
                MOVF
                                 AEXP,W
                MOVWF
                                 CEXP
                MOVF
                                 FPFLAGS,W
                                                  ; save RND flag in DARGB3
                MOVWF
                                 DARGB3
                                 FPFLAGS, RND
                                                 ; disable rounding
                BCF
                MOVLW
                                 EXPBIAS
                                                  ; compute z
                MOVWF
                                 AEXP
```

```
MOVWF
                  BEXP
CLRF
                  BARGB0
CLRF
                  BARGB1
CLRF
                  BARGB2
CALL
                  FPS32
MOVF
                  AEXP,W
                                    ; save z in DARG
MOVWF
                  DEXP
                  AARGB0,W
MOVF
MOVWF
                  DARGB0
MOVF
                  AARGB1,W
MOVWF
                  DARGB1
MOVF
                  AARGB2,W
                  DARGB2
MOVWF
POLL132
                  SQRT32Q,3,0
                                   ; Q(z)
                                    ; save Q(z) in EARG
MOVF
                  AEXP,W
MOVWF
                  EEXP
                  AARGB0,W
MOVF
MOVWF
                  EARGB0
MOVF
                  AARGB1,W
                  EARGB1
MOVWF
MOVF
                  AARGB2,W
MOVWF
                  EARGB2
                  DEXP,W
                                    ; restore z
MOVF
MOVWF
                  AEXP
                  DARGB0,W
MOVF
MOVWF
                  AARGB0
MOVF
                  DARGB1,W
                  AARGB1
MOVWF
MOVF
                  DARGB2,W
MOVWF
                  AARGB2
                  SQRT32P,2,0
POL32
                                    ; P(z)
MOVF
                  EEXP,W
MOVWF
                  BEXP
MOVF
                  EARGB0,W
                  BARGB0
MOVWF
                  EARGB1,W
MOVF
MOVWF
                  BARGB1
MOVF
                  EARGB2,W
MOVWF
                  BARGB2
                  FPD32
                                    ; P(z)/Q(z)
CALL
MOVF
                  DEXP,W
                                    ; restore z
MOVWF
                  BEXP
MOVF
                  DARGB0,W
MOVWF
                  BARGB0
                  DARGB1,W
MOVF
MOVWF
                  BARGB1
MOVF
                  DARGB2,W
MOVWF
                  BARGB2
CALL
                  FPM32
                                    z*P(z)/Q(z)
MOVLW
                  EXPBIAS
MOVWF
                  BEXP
CLRF
                  BARGB0
                  BARGB1
CLRF
CLRF
                  BARGB2
CALL
                  FPA32
                                    ; sqrt(1+z)=1+z*P(z)/Q(z)
```

```
SORT320K
                 BTFSC
                                   CEXP, LSB
                                                    ; is CEXP even or odd?
                 GOTO
                                   RRSQRTOK32
        fixed point multiplication by sqrt(2)
                 BSF
                                   AARGB0,MSB
                                   0xB5
                                                    ; sqrt(2) = 1.41421356237
                 M.TVOM
                 MOVWF
                                   BARGB0
                                   0x04
                 MOVLW
                 MOVWF
                                   BARGB1
                 MOVLW
                                   0xF3
                                   BARGB2
                 MOVWF
                 MOVLW
                                   0x33
                 MOVWF
                                   BARGB3
                                   FXM3232U
                 CALL
                                   AEXP,F
                 INCF
                 BTFSC
                                   AARGB0,MSB
                                   RRSQRTOK32
                 GOTO
                                   AARGB4,F
                 RLF
                                   AARGB3,F
                 RLF
                 RLF
                                   AARGB2,F
                 RLF
                                   AARGB1,F
                 RLF
                                   AARGB0,F
                 DECF
                                   AEXP,F
RRSQRTOK32
                 BCF
                                   AARGB0,MSB
                                                    ; make MSB implicit
                 MOVLW
                                   EXPBIAS
                                                    ; divide exponent by two
                 ADDWF
                                   CEXP,F
                 RRF
                                   CEXP,W
                 MOVWF
                                   AEXP
                 BTFSS
                                   DARGB3, RND
                 RETLW
                                   0x00
                                   FPFLAGS, RND
                 BSF
                                   RND4032
                 CALL
                 RETLW
                                   0x00
DOMERR32
                 BSF
                                   FPFLAGS,DOM
                                                    ; domain error
                 RETLW
                                   0xFF
        minimax rational coefficients for (sqrt(1+z)-1)/z on [0,1]
SQRT32P0
                                   0x84
                                                    ; SQRT32P0 = 6.054736157E1
                 EQU
                                   0x72
SQRT32P00
                 EQU
SQRT32P01
                 EQU
                                   0x30
SQRT32P02
                 EQU
                                   0x80
SQRT32P1
                 EQU
                                   0x84
                                                    ; SQRT32P1 = 5.154073142E1
                 EQU
                                   0x4E
SQRT32P10
SQRT32P11
                 EQU
                                   0x29
SQRT32P12
                 EQU
                                   0xB5
                                                    ; SQRT32P2 = 7.370062896E0
SQRT32P2
                 EQU
                                   0x81
SQRT32P20
                 EQU
                                   0x6B
SQRT32P21
                 EQU
                                   0xD7
SQRT32P22
                 EQU
                                   0x8E
```

```
; SQRT32Q0 = 1.210947497E2
SORT3200
                EOU
                                0x85
SQRT32Q00
                EQU
                                0x72
SQRT32Q01
                EQU
                                0x30
SQRT32Q02
                EQU
                                0x83
SQRT32Q1
                EQU
                                0x86
                                                ; SQRT32Q1 = 1.333554439E2
SQRT32Q10
                EQU
                                0x05
SQRT32Q11
                EQU
                                0x5A
SQRT32Q12
                                0xBC
                EQU
SQRT32Q2
                EQU
                                0x84
                                                ; SQRT32Q2 = 3.294831307E1
SQRT32Q20
                EQU
                                0x03
SQRT32Q21
                EQU
                                0xCB
SQRT32Q22
                                0x13
                EQU
SQRT32Q3
                EQU
                                0x7F
                                                ; SQRT32Q3 = 1.0
SQRT32Q30
                EQU
                                0x00
SQRT32Q31
                                0x00
                EQU
SQRT32Q32
                                0x00
                EQU
Floating Point Relation A < B
                32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
                       TALTB32
        Use:
                CALL
;
        Output: logical result in W
        Result: if A < B TRUE, W = 0x01
                if A < B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
;
                min
;
                       max
                               mean
        Timing: 59
                                15.4
                                       clks
                MOVF
                                AARGB0,W
TALTB32
                XORWF
                                BARGB0,W
                MOVWF
                               TEMPB0
                BTFSC
                                TEMPB0, MSB
                GOTO
                               TALTB320
                               AARGB0,MSB
                BTFSC
                GOTO
                                TALTB32N
TALTB32P
                MOVF
                                AEXP, W
                               BEXP, W
                SUBWF
                                _C
                BTFSS
                                0x00
                RETLW
                BTFSS
                                _Z
                RETLW
                                0x01
                MOVF
                                AARGB0,W
                                BARGB0,W
                SUBWF
                BTFSS
                                _C
                                0x00
                RETLW
                BTFSS
                                Z
                                0x01
                RETLW
                MOVF
                                AARGB1,W
                                BARGB1,W
                SUBWF
```

```
BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _Z
                                    0x01
                  RETLW
                  MOVF
                                    AARGB2,W
                  SUBWF
                                    BARGB2,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  RETLW
                                    0x00
TALTB32N
                  MOVF
                                    BEXP,W
                                    AEXP,W
                  SUBWF
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                                    0x01
                  RETLW
                                    BARGB0,W
                  MOVF
                  SUBWF
                                    AARGB0,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                                    _Z
                  BTFSS
                  RETLW
                                    0 \times 01
                  MOVF
                                    BARGB1,W
                                    AARGB1,W
                  SUBWF
                                    _C
                  BTFSS
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                                    0x01
                  RETLW
                  MOVF
                                    BARGB2,W
                  SUBWF
                                    AARGB2,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0 \times 01
                  RETLW
                                    0x00
TALTB320
                  BTFSS
                                    BARGB0,MSB
                  RETLW
                                    0x01
                  RETLW
                                    0x00
         Floating Point Relation A <= B
                  32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
         Input:
                  32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
         Use:
                  CALL
                           TALEB32
         Output: logical result in W
         Result: if A <= B TRUE, W = 0x01
                  if A <= B FALSE, W = 0x00
         Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                  min
                           max
                                    mean
        Timing: 9
                           32
                                    15.1
                                             clks
```

TALEB32	MOVF	AARGB0,W
11100000	XORWF	BARGB0,W
	MOVWF	TEMPB0
	BTFSC	TEMPB0,MSB
	GOTO	TALEB320
	DTECC	AADCDO MCD
	BTFSC	AARGB0,MSB
	GOTO	TALEB32N
TALEB32P	MOVF	AEXP,W
TABEDJZI		BEXP,W
	SUBWF	
	BTFSS	_C
	RETLW	0x00
	BTFSS	_Z
	RETLW	0x01
	MOME	AADCDO M
	MOVF	AARGB0,W
	SUBWF	BARGB0,W
	BTFSS	_C
	RETLW	0x00
	BTFSS	_Z
	RETLW	0x01
	MOVF	AARGB1,W
	SUBWF	BARGB1,W
	BTFSS	_C
	RETLW	0x00
	BTFSS	_Z
	RETLW	0x01
	KBIBN	01101
	MOVF	AARGB2,W
	SUBWF	BARGB2,W
	BTFSS	_C
	RETLW	0x00
	RETLW	0x01
TALEB32N	MOVF	BEXP,W
	SUBWF	AEXP,W
	BTFSS	_C
	RETLW	0x00
	BTFSS	_Z
		<del>_</del>
	RETLW	0x01
	MOVF	BARGB0,W
	SUBWF	AARGB0,W
	BTFSS	_C
	RETLW	0x00
	BTFSS	_Z
	RETLW	0x01
	MOVF	BARGB1,W
	SUBWF	AARGB1,W
	BTFSS	
	RETLW	_C 0x00
	BTFSS	_Z
	RETLW	0x01
	MOVF	BARGB2,W
	SUBWF	AARGB2,W
	BTFSS	_C
		<del>_</del>
	RETLW	0x00
	RETLW	0x01

```
TALEB320
                  BTFSS
                                    BARGB0, MSB
                  RETLW
                                    0x01
                  RETLW
                                    0x00
         Floating Point Relation A > B
         Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                  32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
         Use:
                  CALL
                          TAGTB32
         Output: logical result in W
         Result: if A > B TRUE, W = 0x01
                 if A > B FALSE, W = 0x00
         Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                  min
                           max
                                    mean
         Timing: 5
                                    34
                                            15.4
                                                      clks
TAGTB32
                 MOVF
                                    BARGB0,W
                                    AARGB0,W
                  XORWF
                  MOVWF
                                    TEMPB0
                                    TEMPB0,MSB
                  BTFSC
                  GOTO
                                    TAGTB320
                  BTFSC
                                    BARGB0,MSB
                  GOTO
                                    TAGTB32N
TAGTB32P
                 MOVF
                                    BEXP,W
                                    AEXP,W
                  SUBWF
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  MOVF
                                    BARGB0,W
                  SUBWF
                                    AARGB0,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                                    BARGB1,W
                  MOVF
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  MOVF
                                    BARGB2,W
                  SUBWF
                                    AARGB2,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  RETLW
                                    0x00
TAGTB32N
                  MOVF
                                    AEXP,W
                  SUBWF
                                    BEXP,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
```

```
_Z
                 BTFSS
                 RETLW
                                   0x01
                                   AARGB0,W
                 MOVF
                                   BARGB0,W
                 SUBWF
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 BTFSS
                                   _{\rm Z}
                                   0x01
                 RETLW
                 MOVF
                                   AARGB1,W
                 SUBWF
                                   BARGB1,W
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                                   _Z
                 BTFSS
                 RETLW
                                   0x01
                                   AARGB2,W
                 MOVF
                                   BARGB2,W
                 SUBWF
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 BTFSS
                                   _Z
                 RETLW
                                   0x01
                                   0x00
                 RETLW
TAGTB320
                 BTFSS
                                   AARGB0,MSB
                 RETLW
                                   0x01
                 RETLW
                                   0x00
        Floating Point Relation A >= B
                 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Input:
                 32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
;
        Use:
                 CALL
                          TAGEB32
        Output: logical result in W
;
        Result: if A >= B TRUE, W = 0x01
;
                 if A >= B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
;
                 min
                          max
                                   mean
        Timing: 5
                          32
                                   15.1
                                            clks
TAGEB32
                 MOVF
                                   BARGB0,W
                 XORWF
                                   AARGB0,W
                 MOVWF
                                   TEMPB0
                 BTFSC
                                   TEMPB0,MSB
                                   TAGEB320
                 GOTO
                 BTFSC
                                   BARGB0, MSB
                                   TAGEB32N
                 GOTO
TAGEB32P
                 MOVF
                                   BEXP,W
                 SUBWF
                                   AEXP,W
                 BTFSS
                                   _C
                 RETLW
                                   0x00
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
```

```
MOVF
                               BARGB0,W
               SUBWF
                               AARGB0,W
               BTFSS
                               _C
                               0x00
               RETLW
               BTFSS
                               _Z
               RETLW
                               0x01
               MOVF
                               BARGB1,W
                               AARGB1,W
               SUBWF
               BTFSS
                               _C
               RETLW
                               0x00
               BTFSS
                                _Z
               RETLW
                               0x01
                               BARGB2,W
               MOVF
               SUBWF
                               AARGB2,W
               BTFSS
                               _C
                               0x00
               RETLW
                               0x01
               RETLW
TAGEB32N
                               AEXP,W
               MOVF
               SUBWF
                               BEXP,W
               BTFSS
                               _C
                               0x00
               RETLW
                               _Z
               BTFSS
               RETLW
                               0 \times 01
               MOVF
                               AARGB0,W
                               BARGB0,W
               SUBWF
               BTFSS
                               _C
               RETLW
                               0x00
               BTFSS
                               _{\rm Z}
                               0x01
               RETLW
               MOVF
                               AARGB1,W
               SUBWF
                               BARGB1,W
               BTFSS
                               _C
                               0x00
               RETLW
               BTFSS
                               _Z
               RETLW
                               0x01
                               AARGB2,W
               MOVF
                               BARGB2,W
               SUBWF
               BTFSS
                               _C
               RETLW
                               0x00
                               0x01
               RETLW
TAGEB320
                               AARGB0,MSB
               BTFSS
               RETLW
                               0x01
               RETLW
                               0x00
Floating Point Relation A == B
               32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Input:
               32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
       Use:
               CALL
                       TAEQB32
       Output: logical result in W
       Result: if A == B TRUE, W = 0x01
               if A == B FALSE, W = 0x00
```

```
Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                 min
                          max
                                   mean
        Timing: 5
                                   18
                                            7.4
                                                    clks
                          5
TAEQB32
                 MOVF
                                   BEXP,W
                 SUBWF
                                   AEXP,W
                 BTFSS
                                   _{\rm Z}
                                   0x00
                 RETLW
                 MOVF
                                   BARGB0,W
                 SUBWF
                                   AARGB0,W
                 BTFSS
                                   _{\rm Z}
                                   0x00
                 RETLW
                 MOVF
                                   BARGB1,W
                 SUBWF
                                   AARGB1,W
                 BTFSS
                                   _{\rm Z}
                                   0x00
                 RETLW
                 MOVF
                                   BARGB2,W
                 SUBWF
                                   AARGB2,W
                 BTFSS
                                   0x00
                 RETLW
                                   0x01
                 RETLW
        Floating Point Relation A =! B
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                 32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
        Use:
                 CALL
                          TANEB32
;
        Output: logical result in W
        Result: if A =! B TRUE, W = 0x01
;
                 if A =! B FALSE, W = 0x00
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                 min
;
                          max
                                   mean
        Timing: 5
                          18
                                   7.4
                                            clks
                                   BEXP,W
TANEB32
                 MOVF
                 SUBWF
                                   AEXP,W
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
                                   BARGB0,W
                 MOVF
                                   AARGB0,W
                 SUBWF
                 BTFSS
                                   _Z
                                   0x01
                 RETLW
                 MOVF
                                   BARGB1,W
                                   AARGB1,W
                 SUBWF
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
```

## **AN660**

1	MOVF	BARGB2,W
:	SUBWF	AARGB2,W
Ţ	BTFSS	_Z
]	RETLW	0x01
]	RETLW	0x00
; * * * * * * * * * * * * * * *	* * * * * * * * * * * * * * *	********************
; * * * * * * * * * * * * * * *	******	****************

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: PIC17CXXX 24-BIT ELEMENTARY FUNCTION LIBRARY

```
RCS Header $Id: ef24.a17 1.55 1997/02/25 14:32:22 F.J.Testa Exp $
       $Revision: 1.55 $
       PIC17 24-BIT ELEMENTARY FUNCTION LIBRARY
       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
       for an error condition specified in FPFLAGS.
       Test statistics are typically from 100000 trials, with timing in cycles
;
       and error in the next significant byte. In all cases, the floating point
       routines satisfy a half unit in the last position (.5*ulp) accuracy
;
       requirement, resulting in |nsb error| <= 0x7F. The integer and logical
;
       routines are exact.
;
      Routine Function
                             Timing in cycles
                                                    Error in nsb
                             min
                                    max
                                           mean
                                                    min
                                                                   mean
                                                                           rms
       SQRT24
              24 bit sqrt(x) 6
                                     327
                                             292.7
                                                    -0x10
                                                            0x05
                                                                   -3.56
                                                                           5.20
       EXP24
               24 bit exp(x) 645
                                     999
                                             859.3
                                                    -0x6E
                                                            0x69
                                                                   -0.97
                                                                           35.75
       EXP1024 24 bit exp10(x) 646
                                    1002
                                          859.5
                                                    -0x75
                                                            0x77
                                                                   -0.94
                                                                          40.34
       LOG24
               24 bit log(x) 12
                                    1442
                                          1316.5 - 0 \times 02
                                                            0x00
                                                                   -0.81
                                                                         0.92
       LOG1024 24 bit log10(x) 12
                                           1317.7 -0 \times 01
                                                            0x00
                                                                          0.57
                                    1457
                                                                   -0.32
       SIN24
               24 bit sin(x) 834
                                     1625
                                            1465.7 -0x56
                                                            0x13
                                                                   -7.12
                                                                           20.89
                                                                   -7.13
       COS24
               24 bit cos(x) 942
                                     1637
                                           1465.7 -0x56
                                                                           20.90
                                                            0x13
       SINCOS24 24 bit sin(x), cos(x) 15162248
                                             2128.2 -0x56
                                                            0x13
                                                                   -7.12
                                                                           20.89
                                                    -0x56
                                                            0x13
                                                                   -7.13
                                                                           20.90
       POW24 24 bit pow(x,y)=x**y
       FLOOR24 24 bit floor(x) 18 39
                                           30.11
                                                    0x00
                                                            0x00
                                                                   0.0
                                                                          0.0
      TALTB24 24 bit A < B 8
                                     27
                                            11.5
       TALEB24 24 bit A <= B 8
                                     25
                                            11.5
       TAGTB24 24 bit A > B
                           8
                                     2.7
                                            11.5
       TAGEB24 24 bit A >= B
                                     25
                                            11.5
       TAEQB24 24 bit A == B
                                     11
                                             6.0
       TANEB24 24 bit A != B
                                             6.0
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
polynomial evaluation macros
POLL124 macro
                     COF, N, ROUND
       32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
       with leading coefficient of one, and where AARG is assumed have been saved
       in DARG when N > 1. The result is in AARG.
       ROUND = Ono rounding is enabled; can be previously enabled
       ROUND = 1rounding is enabled
       ROUND = 2rounding is enabled then disabled before last add
       ROUND = 3rounding is assumed disabled then enabled before last add
       ROUND = 4rounding is assumed enabled and then disabled before last
             add if DARGB3,RND is clear
       local
            i,j
       variablei = N, j = 0
       variablei = i - 1
       if
              ROUND == 1 | ROUND == 2
              BSF
                             FPFLAGS, RND
       endif
                             COF#v(i)
              MOVLW
              MOVWF
                             BEXP
       variable j = 0
       while
              j <= 2
              MOVLW
                             COF#v(i)#v(j)
              MOVWF
                             BARGB#v(j)
       variable j = j + 1
       endw
              CALL
                             FPA32
       variable i = i - 1
       while
             i >= 0
```

```
MOVFP
                           DEXP, WREG
         MOVPF
                            WREG, BEXP
         MOVFP
                            DARGB0, WREG
                            WREG,BARGB0
         {\tt MOVPF}
                           DARGB1,WREG
         MOVFP
                           WREG, BARGB1
         MOVPF
         MOVFP
                           DARGB2, WREG
         MOVPF
                            WREG, BARGB2
         CALL
                           FPM32
         MOVLW
                            COF#v(i)
                            BEXP
         MOVWF
variable j = 0
while
         j <= 2
         MOVLW
                           COF#v(i)#v(j)
         MOVWF
                           BARGB#v(j)
variable j = j + 1
endw
         i == 0
if
         if
                  ROUND == 2
         BCF
                           FPFLAGS, RND
         \verb"endif"
         if
                  ROUND == 3
         BSF
                           FPFLAGS, RND
         \verb"endif"
         if
                  ROUND == 4
                           DARGB3,RND
         BTFSS
         BCF
                           FPFLAGS, RND
         endif
                  ROUND == 5
         if
         BTFSC
                           DARGB3,RND
         BSF
                           FPFLAGS, RND
         endif
endif
         CALL
                           FPA32
variablei = i - 1
endw
{\tt endm}
                  COF, N, ROUND
macro
```

POL24

```
32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
and where AARG is assumed have been be saved in DARG when N > 1.
The result is in AARG.
ROUND = Ono rounding is enabled; can be previously enabled
ROUND = 1rounding is enabled
ROUND = 2rounding is enabled then disabled before last add
ROUND = 3rounding is assumed disabled then enabled before last add
ROUND = 4rounding is assumed enabled and then disabled before last
       add if DARGB3,RND is clear
ROUND = 5rounding is assumed disabled and then enabled before last
       add if DARGB3,RND is set
local
      i,j
variablei = N, j = 0
if
        ROUND == 1 | ROUND == 2
        BSF
                         FPFLAGS, RND
endif
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
while
        j <= 2
        MOVLW
                         COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variable j = j + 1
endw
        CALL
                         FPM32
variablei = i - 1
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variable j = 0
while
        j <= 2
        MOVLW
                         COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variable j = j + 1
endw
        CALL
                         FPA32
variable i = i - 1
while
       i >= 0
        MOVFP
                         DEXP, WREG
        MOVPF
                         WREG, BEXP
                         DARGB0, WREG
        MOVFP
        MOVPF
                         WREG, BARGB0
                         DARGB1, WREG
        MOVFP
                         WREG, BARGB1
        MOVPF
        MOVFP
                         DARGB2, WREG
                         WREG, BARGB2
        MOVPF
```

```
CALL
                         FPM32
        MOVLW
                         COF#v(i)
        MOVWF
                         BEXP
variable j = 0
      j <= 2
while
                         COF#v(i)#v(j)
        MOVLW
        MOVWF
                         BARGB#v(j)
variablej = j + 1
endw
        i == 0
if
                ROUND == 2
        if
        BCF
                        FPFLAGS, RND
        endif
        if
                 ROUND == 3
        BSF
                        FPFLAGS, RND
        endif
        if
                 ROUND == 4
        BTFSS
                         DARGB3, RND
        BCF
                         FPFLAGS, RND
        {\tt endif}
                 ROUND == 5
        if
                         DARGB3, RND
        BTFSC
        BSF
                         FPFLAGS, RND
        endif
endif
        CALL
                        FPA32
variable i = i - 1
endw
endm
Evaluate exp(x)
Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                EXP24
Use:
        CALL
Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
```

```
Result: AARG <-- EXP( AARG )
        Testing on [MINLOG, MAXLOG] from 100000 trials:
                  min
                          max
                                    mean
        Timing: 645
                           999
                                    859.3
                                            clks
                  min
                          max
                                    mean
                                            rms
        Error:
                 -0x6E
                           0x69
                                    -0.97
                                             35.75
        This approximation of the exponential function is based upon the
         expansion
                  exp(x) = e^{*x} = 2^{**}(x/log(2)) = 2^{**}z * 2^{**}n,
                          x/\log(2) = z + n,
        where 0 <= z < 1 and n is an integer, evaluated during range reduction.
        Segmented third degree minimax polynomial approximations are used to
        estimate 2**z on the intervals [0,.25], [.25,.5], [.5,.75] and [.75,1].
EXP24
                 MOVLW
                                    0x66
                                                      ; test for |x| < 2**(-24)/2
                  CPFSGT
                                    EXP
                  GOTO
                                    EXP24ONE
                                                      ; return e^*x = 1
                  BTFSC
                                    AARGBO, MSB
                                                     ; determine sign
                  GOTO
                                    TNEXP24
TPEXP24
                  MOVFP
                                    AEXP, WREG
                                                      ; positive domain check
                  SUBLW
                                    MAXLOG24EXP
                  BTFSS
                                    _C
                                    DOMERR24
                  GOTO
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    EXP24ARGOK
                  MOVED
                                    AARGB0, WREG
                                    MAXLOG24B0
                  SUBLW
                  BTFSS
                  GOTO
                                    DOMERR24
                  BTFSS
                                    _{\rm Z}
                                    EXP24ARGOK
                  GOTO
                  MOVFP
                                    AARGB1, WREG
                  SUBLW
                                    MAXLOG24B1
                                    _C
                  BTFSS
                  GOTO
                                    DOMERR24
                  GOTO
                                    EXP24ARGOK
TNEXP24
                 MOVFP
                                                     ; negative domain check
                                    AEXP, WREG
                  SUBLW
                                    MINLOG24EXP
                  BTFSS
                                    _C
                  GOTO
                                    DOMERR24
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    EXP24ARGOK
                  MOVFP
                                    AARGB0, WREG
                                    MINLOG24B0
                  SUBLW
                  BTFSS
                                    _C
                                    DOMERR24
                  GOTO
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    EXP24ARGOK
```

	MOVFP	AARGB1,WREG	
	SUBLW	MINLOG24B1	
	BTFSS	_C	
	GOTO	DOMERR24	
EXP24ARGOK	MOMED	EDELACE MDEC	
	MOVEP	FPFLAGS, WREG	· gave wounding flog
	MOVWF	DARGB3	; save rounding flag
	BCF	FPFLAGS,RND	; disable rounding
	DCI	III EMOS / MAS	, dipapie rounding
	CALL	RREXP24	; range reduction
	MOVLW	0x7E	
	CPFSEQ	AEXP	
	GOTO	EXP24L	
EXP24H	BTFSS	AARGB0,MSB-1	
	GOTO	EXP24HL	
	POL24	EXP24HH,3,0	; minimax approximation on [.75,1]
	GOTO	EXP24OK	
EXP24HL	POL24	EXP24HL,3,0	; minimax approximation on [.5,.75]
	GOTO	EXP24OK	
EXP 0.41	MOLITE	075	
EXP24L	MOVLW	0x7D	
	CPFSEQ	AEXP	
	GOTO	EXP24LL	
	POL24	EXP24LH,3,0	; minimax approximation on [.25,.5]
	10221	2111 2 1211 / 5 / 6	manaman appronamation on [125,15]
	GOTO	EXP24OK	
EXP24LL			
	POL24	EXP24LL,3,0	; minimax approximation on [0,.25]
EXP24OK			
	MOVFP	EARGB3,WREG	
	ADDWF	AEXP,F	
	BTFSS	האחמה אחה	
	RETLW	DARGB3,RND 0x00	
	KEILW	0.000	
	BSF	FPFLAGS.RND	; restore rounding flag
	CALL	RND3224	
	RETLW	0x00	
EXP24ONE	MOVLW	EXPBIAS	; return e**x = 1.0
	MOVWF	AEXP	
	CLRF	AARGB0,F	
	CLRF	AARGB1,F	
	CLRF	AARGB2,F	
	RETLW	0x00	
DOMERR24	BSF	FPFLAGS,DOM	; domain error
	RETLW	0xFF	
	***		
;**************************************			
; Range r	eduction routine	for the evnonent	ial function
. Range I	Caucaton Loucine	101 CITC CAPOTICITY.	141 2411001011
;	$x/\log(2) = z + 1$	n	
	, <u>j</u> , , – , ,		

## **AN660**

RREXP24			
ICCDITE 2 I	MOVPF	AARGB0,DARGB0	; save sign
	BSF		; make MSB explicit
	201	111102071102	, maile figs empirers
	MOVPF	AARGB0,BARGB0	
	MOVPF	AARGB1,BARGB1	
		,	
	MOVLW	0xB8	$i(1/\ln(2) = 1.44269504089$
	MOVPF	WREG, AARGB0	
	MOVLW	0xAA	
	MOVPF	WREG, AARGB1	
	MOVLW	0x3B	
	MOVPF	WREG, AARGB2	
	CALL	FXM2416U	; x * (1/ln2)
	INCF	AEXP,F	
	BTFSC	AARGB0,MSB	
	GOTO	RREXP24YOK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RREXP24YOK	BTFSS	DARGB0,MSB	; restore sign
	BCF	AARGB0,MSB	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, DEXP	; save x/ln2 in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	CALL	FLOOR24	
	MOVFP	AEXP, WREG	
	MOVPF		; save float(n) in BARG
	BTFSC	_Z	1
	GOTO	RREXP24ZOK	i done if $n = 0$
	MOVPF	AARGB0,BARGB0	
	MOVPF	AARGB1,BARGB1	
	CLRF	BARGB2,F	
	CATT	TMT 2/16	[ (1/1,2) ]
	CALL	INT2416	; $n = [x * (1/ln2)]$
	MOVPF	77DCD1 F7DCD2	; save n in EARG
	MOVPF	AARGDI, LARGDS	, save ii iii EARG
	MOVFP	DEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	DARGB0, AARGB0	
	MOVFP	DARGB1, AARGB1	
	MOVFP	DARGB2,AARGB2	
	110 11 1	DintoD2 / InntoD2	
	CALL	FPS32	
	MOVFP	AEXP, WREG	
	MOVPF		; save z in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	RETLW	0x00	

```
RREXP24ZOK
                 MOVFP
                                   DEXP, WREG
                                   AEXP
                 MOVWF
                                   DARGB0, AARGB0
                 MOVFP
                 MOVFP
                                   DARGB1, AARGB1
                 MOVED
                                   DARGB2, AARGB2
                 CLRF
                                   EARGB3,F
                                   0x00
                 RETIW
        third degree minimax polynomial coefficients for 2**(x) on [.75,1]
EXP24HH0
                 EQU
                                   0x7E
                                                    ; EXP24HH0 = .99103284632
EXP24HH00
                 EQU
                                   0x7D
EXP24HH01
                 EQU
                                   0xB4
EXP24HH02
                                   0x54
                 EQU
EXP24HH1
                 EQU
                                   0x7E
                                                     ; EXP24HH1 = .73346850266
EXP24HH10
                 EQU
                                   0x3B
EXP24HH11
                 EQU
                                   0xC4
EXP24HH12
                 EQU
                                   0x97
EXP24HH2
                 EQU
                                   0x7C
                                                    ; EXP24HH2 = .17374128273
EXP24HH20
                 EQU
                                   0x31
EXP24HH21
                                   0xE9
                 EQU
EXP24HH22
                 EQU
                                   0x3C
EXP24HH3
                 EQU
                                   0x7B
                                                     ; EXP24HH3 = .10175678143
EXP24HH30
                 EQU
                                   0x50
EXP24HH31
                                   0x65
                 EQU
EXP24HH32
                                   0xDC
                 EQU
        third degree minimax polynomial coefficients for 2**(x) on [.5,.75]
EXP24HL0
                                                    ; EXP24HL0 = .99801686089
                 EQU
                                   0x7E
EXP24HL00
                 EQU
                                   0x7F
EXP24HL01
                                   0x7E
                 EQU
EXP24HL02
                 EQU
                                   0x08
EXP24HI.1
                 EQU
                                   0x7E
                                                    ; EXP24HL1 = .70586404164
EXP24HL10
                 EQU
                                   0x34
EXP24HL11
                 EQU
                                   0xB3
EXP24HL12
                 EQU
                                   0x81
EXP24HL2
                                   0x7C
                                                     ; EXP24HL2 = .21027360637
                 EQU
EXP24HL20
                 EQU
                                   0x57
EXP24HL21
                 EQU
                                   0x51
EXP24HL22
                 EQU
                                   0xF7
EXP24HL3
                 EQU
                                   0x7B
                                                    ; EXP24HL3 = .85566912730E-1
EXP24HL30
                 EQU
                                   0x2F
EXP24HL31
                 EQU
                                   0x3D
EXP24HL32
                 EQU
                                   0xB5
        third degree minimax polynomial coefficients for 2**(x) on [.25,.5]
EXP24LH0
                 EQU
                                   0x7E
                                                    ; EXP24LH0 = .99979384559
                                   0x7F
EXP24LH00
                 EQU
EXP24LH01
                 EQU
                                   0xF2
EXP24LH02
                                   0x7D
                 EQU
EXP24LH1
                 EQU
                                   0x7E
                                                     ; EXP24LH1 = .69545887384
EXP24LH10
                 EQU
                                   0x32
```

```
EXP24LH11
               EOU
                             0x09
EXP24LH12
                             0x98
              EOU
EXP24LH2
                             0x7C
                                            ; EXP24LH2 = .23078300446
               EQU
EXP24LH20
              EQU
                             0x6C
EXP24LH21
              EQU
                             0x52
EXP24LH22
                             0x61
              EQU
                             0x7B
                                            ; EXP24LH3 = .71952910179E-1
EXP24LH3
              EQU
EXP24TH30
                             0x13
              EQU
EXP24LH31
               EQU
                             0x5C
EXP24LH32
                             0x0C
              EQU
       third degree minimax polynomial coefficients for 2**(x) on [0,.25]
EXP24LL0
              EQU
                             0x7E
                                            ; EXP24LL0 = .99999970657
EXP24LL00
              EQU
                             0x7F
EXP24LL01
              EQU
                             0xFF
EXP24LL02
                             0xFB
              EQU
EXP24LL1
                             0x7E
                                            ; EXP24LL1 = .69318585159
              EQU
EXP24LL10
              EQU
                             0x31
EXP24LL11
              EQU
                             0x74
EXP24LL12
                             0xA1
              EQU
EXP24LL2
              EQU
                             0x7C
                                            ; EXP24LL2 = .23944330933
EXP24LL20
              EQU
                             0x75
EXP24LL21
                             0x30
              EQU
EXP24LL22
              EQU
                             0xA0
EXP24LL3
              EQU
                             0x7A
                                            ; EXP24LL3 = .60504944237E-1
EXP24LL30
               EQU
                             0x77
EXP241.1.31
                             0xD4
              EQU
EXP24LL32
                             0x08
              EQU
Evaluate expl0(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
              CALL
                      EXP1024
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- EXP10( AARG )
       Testing on [MINLOG10, MAXLOG10] from 10000 trials:
               min
                      max
                             mean
       Timing: 646
                      1002
                             859.5
                                     clks
              min
                      max
                             mean
                                     rms
       Error: -0x75
                      0x77
                             -0.94
                                     40.34
                                            nsb
;------
       This approximation of the base 10 exponential function is based upon the
       expansion
               expl0(x) = 10**x = 2**(x/log10(2)) = 2**z * 2**n
                      x/\log 10(2) = z + n,
```

```
where 0 <= z < 1 and n is an integer, evaluated during range reduction.
;
        Segmented third degree minimax polynomial approximations are used to
        estimate 2^{**z} on the intervals [0,.25], [.25,.5], [.5,.75] and [.75,1].
EXP1024
                 MOVT W
                                    0x66
                                                      ; test for |x| < 2**(-24)/2
                  CPFSGT
                                    EXP
                  GOTO
                                    EXP1024ONE
                                                      ; return 10**x = 1
                  BTFSC
                                    AARGB0,MSB
                                                      ; determine sign
                  GOTO
                                    TNEXP1024
TPEXP1024
                 MOVFP
                                    AEXP, WREG
                                                      ; positive domain check
                  SUBLW
                                    MAXLOG1024EXP
                 BTFSS
                                    _C
                 GOTO
                                    DOMERR24
                  BTFSS
                                    _{\rm Z}
                  GOTO
                                    EXP1024ARGOK
                                    AARGB0, WREG
                 MOVFP
                                   MAXLOG1024B0
                  SUBLW
                  BTFSS
                                    _C
                  GOTO
                                    DOMERR24
                 BTFSS
                                    _{\rm Z}
                                    EXP1024ARGOK
                 GOTO
                 MOVFP
                                    AARGB1, WREG
                                   MAXLOG1024B1
                  SUBLW
                  BTFSS
                                    _C
                                    DOMERR24
                  GOTO
                  GOTO
                                    EXP1024ARGOK
TNEXP1024
                 MOVFP
                                   AEXP, WREG
                                                      ; negative domain check
                                   MINLOG1024EXP
                  SUBLW
                 BTFSS
                                    _C
                                    DOMERR24
                 GOTO
                  BTFSS
                                    _{z}
                  GOTO
                                    EXP1024ARGOK
                  MOVFP
                                    AARGB0, WREG
                  SUBLW
                                   MINLOG1024B0
                 BTFSS
                                    C
                  GOTO
                                    DOMERR24
                 BTFSS
                                    _{\rm Z}
                  GOTO
                                    EXP1024ARGOK
                                    AARGB1, WREG
                 MOVFP
                  SUBLW
                                    MINLOG1024B1
                  BTFSS
                                    _C
                  GOTO
                                    DOMERR24
EXP1024ARGOK
                 MOVFP
                                    FPFLAGS, WREG
                 MOVWF
                                    DARGB3
                                                      ; save rounding flag
                 BCF
                                    FPFLAGS, RND
                                                      ; disable rounding
                                    RREXP1024
                  CALL
                                                      ; range reduction
                 {\tt MOVLW}
                                    0x7E
                  CPFSEQ
                                    AEXP
                                    EXP1024L
                  GOTO
EXP1024H
                  BTFSS
                                    AARGB0,MSB-1
                                    EXP1024HL
                  GOTO
```

```
POL24
                                EXP24HH, 3, 0
                                                ; minimax approximation on [.75,1]
                                EXP1024OK
                GOTO
                                                ; minimax approximation on [.5,.75]
EXP1024HL
                POL24
                                EXP24HL, 3, 0
                GOTO
                                EXP1024OK
EXP1024L
                                0x7D
                MOVLW
                CPFSEQ
                                AEXP
                GOTO
                                EXP1024LL
                POL24
                                EXP24LH, 3, 0
                                                ; minimax approximation on [.25,.5]
                GOTO
                                EXP1024OK
EXP1024LL
                POL24
                                EXP24LL,3,0
                                                ; minimax approximation on [0,.25]
EXP1024OK
                MOVFP
                                EARGB3, WREG
                ADDWF
                                AEXP,F
                BTFSS
                                DARGB3, RND
                RETLW
                                0x00
                                FPFLAGS, RND
                                                ; restore rounding flag
                BSF
                CALL
                                RND3224
                                0x00
                RETLW
EXP1024ONE
                MOVLW
                                EXPBIAS
                                                ; return 10**x = 1.0
                MOVWF
                                AEXP
                                AARGB0,F
                CLRF
                                AARGB1,F
                CLRF
                CLRF
                                AARGB2,F
                                0x00
                RETLW
Range reduction routine for the exponential function
                x/\log 10(2) = z + n
RREXP1024
                MOVPF
                                AARGB0, DARGB0
                BSF
                                AARGB0,MSB
                MOVPF
                                AARGB0,BARGB0
                MOVPF
                                AARGB1, BARGB1
                                                ; 1/log10(2) = 3.32192809489
                MOVLW
                                0xD4
                MOVPF
                                WREG, AARGB0
                MOVLW
                                0x9A
                MOVPF
                                WREG, AARGB1
                MOVLW
                                0x78
                MOVPF
                                WREG, AARGB2
                                FXM2416U
                                                ; x * (1/log10(2))
                CALL
                INCF
                                AEXP,F
                INCF
                                AEXP,F
                BTFSC
                                AARGB0,MSB
                GOTO
                                RREXP24YOK
                RLCF
                                AARGB3,F
```

```
AARGB2,F
                  RLCF
                                    AARGB1,F
                  RLCF
                  RLCF
                                    AARGB0,F
                  DECF
                                    AEXP,F
RREXP1024YOK
                  BTFSS
                                    DARGB0, MSB
                                                      ; restore sign
                  BCF
                                    AARGB0,MSB
                  MOVFP
                                    AEXP, WREG
                                                      ; save x/log10(2) in DARG
                  MOVPF
                                    WREG, DEXP
                  MOVPF
                                    AARGB0, DARGB0
                                    AARGB1, DARGB1
                  MOVPF
                  MOVPF
                                    AARGB2, DARGB2
                                    FLOOR24
                  CALL
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, BEXP
                                                      ; save float(n) in BARG
                  BTFSC
                                    _{\rm Z}
                                    RREXP1024ZOK
                                                      idone if n = 0
                  GOTO
                  MOVPF
                                    AARGB0, BARGB0
                  MOVPF
                                    AARGB1,BARGB1
                  CLRF
                                    BARGB2,F
                                                      ; n = [x * (1/log10(2))]
                  CALL
                                    INT2416
                  MOVPF
                                    AARGB1, EARGB3
                                                      ; save n in EARG
                                    DEXP, WREG
                  MOVFP
                                    WREG, AEXP
                  MOVPF
                  MOVFP
                                    DARGB0, AARGB0
                  MOVFP
                                    DARGB1, AARGB1
                  MOVFP
                                    DARGB2, AARGB2
                  CALL
                                    FPS32
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, DEXP
                                                      ; save z in DARG
                  MOVPF
                                    AARGB0, DARGB0
                                    AARGB1, DARGB1
                  MOVPF
                  MOVPF
                                    AARGB2, DARGB2
                  RETLW
                                    0x00
RREXP1024ZOK
                  MOVFP
                                    DEXP, WREG
                  MOVWF
                                    AEXP
                  MOVFP
                                    DARGB0, AARGB0
                  MOVFP
                                    DARGB1, AARGB1
                  MOVFP
                                    DARGB2, AARGB2
                  CLRF
                                    EARGB3,F
                  RETLW
                                    0x00
         Evaluate log(x)
                  24 bit floating point number in AEXP, AARGBO, AARGB1
                  CALL
                          LOG24
         Use:
         Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
```

```
Result: AARG <-- LOG( AARG )
        Testing on [MINNUM, MAXNUM] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 12
                         1442
                                  1316.5 clks
                 min
                         max
                                  mean
                                           rms
        Error: -0x02
                         0x00
                                  -0.81
                                           0.92
        This approximation of the natural log function is based upon the
        expansion
                 log(x) = log(2) * log2(x) = log(2) * (n + log2(f))
        where .5 \ll f \ll 1 and n is an integer. The additional transformation
                         2*f-1, f < 1/sqrt(2), n=n-1
                          f-1, otherwise
        produces a naturally segmented representation of log2(1+z) on the
        intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
        approximations.
LOG24
                 CLRF
                                  AARGB2,W
                                                   ; clear next significant byte
                 BTFSS
                                  AARGB0,MSB
                                                   ; test for negative argument
                 CPFSGT
                                  AEXP
                                                   ; test for zero argument
                 GOTO
                                  DOMERR24
                 MOVFP
                                  FPFLAGS, WREG
                                                   ; save rounding flag
                 MOVWF
                                  DARGB3
                                  FPFLAGS, RND
                                                   ; disable rounding
                 BCF
                 MOVED
                                  AEXP, WREG
                                  WREG, EARGB3
                 MOVPF
                 MOVLW
                                  EXPBIAS-1
                 SUBWF
                                  EARGB3,F
                 MOVWF
                                  AEXP
                 MOVLW
                                  0xF3
                                                   ; .70710678118655 = 7E3504F3
                 SUBWF
                                  AARGB2,W
                 MOVLW
                                  0x04
                 SUBWFB
                                  AARGB1,W
                 MOVLW
                                  0x35
                 SUBWFB
                                  AARGB0,W
                 BTFSS
                                  _C
                 GOTO
                                  LOG24L
        minimax rational approximation on [0,.sqrt(2)-1]
LOG24H
                 MOVLW
                                  0x7F
                 MOVPF
                                  WREG, BEXP
                 CLRF
                                  BARGB0,F
                                  BARGB1,F
                 CLRF
                 CLRF
                                  BARGB2,F
                 CALL
                                  FPS32
                 MOVFP
                                  AEXP, WREG
```

	MOVPF	WREG, DEXP
	MOVPF	AARGB0, DARGB0
	MOVPF	AARGB1,DARGB1
	MOVPF	AARGB2,DARGB2
	POLL124	LOG24HQ,2,0
	MOVFP	AEXP, WREG
	MOVPF	WREG, CEXP
	MOVPF	AARGB0,CARGB0
	MOVPF	AARGB1,CARGB1
	MOVPF	AARGB2,CARGB2
	MOVFP	DEXP, WREG
	MOVPF	WREG, AEXP
	MOVFP	DARGBO, AARGBO
	MOVFP	DARGB1,AARGB1
	MOVFP	DARGB2,AARGB2
	DOI 24	1.0024UD 1.0
	POL24	LOG24HP,1,0
	MOVFP	CEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	CARGB0, WREG
	MOVPF	WREG, BARGB0
	MOVFP	CARGB1, WREG
	MOVPF	WREG, BARGB1
	MOVFP	CARGB2, WREG
	MOVPF	WREG,BARGB2
	CALL	FPD32
	GOTO	LOG240K
;	minimax rational	approximation on [1/sqrt(2)-1,0]
; LOG24L	minimax rational	approximation on [1/sqrt(2)-1,0]
	minimax rational	approximation on [1/sqrt(2)-1,0]  AEXP,F
	INCF	AEXP,F
	INCF MOVLW	AEXP,F 0x7F
	INCF MOVLW MOVPF	AEXP,F 0x7F WREG,BEXP BARGB0,F
	INCF MOVLW MOVPF CLRF	AEXP,F 0x7F WREG,BEXP
	INCF MOVLW MOVPF CLRF CLRF CLRF	AEXP,F 0x7F WREG,BEXP BARGB0,F BARGB1,F BARGB2,F
	INCF MOVLW MOVPF CLRF CLRF	AEXP,F 0x7F WREG,BEXP BARGB0,F BARGB1,F
	INCF MOVLW MOVPF CLRF CLRF CLRF	AEXP,F 0x7F WREG,BEXP BARGB0,F BARGB1,F BARGB2,F
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F FPS32 EARGB3, F
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CLRF MOVFP	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F FPS32 EARGB3, F AEXP, WREG
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F FPS32 EARGB3, F AEXP, WREG WREG, DEXP
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CMF CALL DECF MOVFP MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32 EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32 EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CMF CALL DECF MOVFP MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32 EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32 EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF MOVPF MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF MOVPF MOVPF MOVPF MOVPF MOVPF MOVPF MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP AARGB0, CARGB0
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB1, CARGB1
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB1 AARGB2, CARGB2  DEXP, WREG
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CLRF CALL  DECF  MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB1 AARGB2, CARGB2  DEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  DEXP, WREG WREG, AEXP
	INCF MOVLW MOVPF CLRF CLRF CLRF CLRF CLRF CALL DECF MOVFP MOVPF	AEXP, F 0x7F WREG, BEXP BARGB0, F BARGB1, F BARGB2, F  FPS32  EARGB3, F  AEXP, WREG WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  LOG24LQ, 2, 0  AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB1 AARGB2, CARGB2  DEXP, WREG

	MOVFP	DARGB2,AARGB2
	POL24	LOG24LP,1,0
	MOVFP	CEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	CARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	CARGB1, WREG
	MOVPF	WREG, BARGB1
	MOVFP	CARGB2, WREG
	MOVPF	WREG, BARGB2
	110 V 1 1	MADO / DI MODE
	CALL	FPD32
LOG240K		
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2,WREG
	MOVPF	WREG,BARGB2
	CALL	FPM32
	MOVFP	AEXP, WREG
	MOVPF	WREG, DEXP
	MOVPF	AARGB0, DARGB0
	MOVPF	AARGB1,DARGB1
	MOVPF	AARGB2, DARGB2
	CLRF	AARGBO,F
	MOVFP	EARGB3,AARGB1
	BTFSC	AARGB1,MSB
	SETF	AARGBO,F
	CALL	FL01624
	CLRF	AARGB2,F
	CLIKI	AMODZ, I
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2, WREG
	MOVPF	WREG,BARGB2
	CALL	FPA32
; fixed po	oint multiplicati	on by log(2)
	MOVPF	AARGBO, EARGB3
	BSF	AARGB0,MSB
	MOVLW	0xB1
	MOVPF	WREG, BARGBO
	MOVLW	0x72
	MOVPF	WREG, BARGB1
	MOVLW	0x18
	MOVPF	WREG,BARGB2
	CALL	EVM2424II
	CALL	FXM2424U
	BTFSC	AARGB0,MSB

```
GOTO
                                   LOG24DONE
                 RLCF
                                   AARGB3,F
                 RLCF
                                   AARGB2,F
                 RLCF
                                   AARGB1,F
                 RLCF
                                   AARGB0,F
                 DECE
                                   AEXP,F
LOG24DONE
                 BTFSS
                                   EARGB3, MSB
                 BCF
                                   AARGB0, MSB
                 BTFSS
                                   DARGB3, RND
                                   0x00
                 RETLW
                 BSF
                                   FPFLAGS, RND
                                                     ; restore rounding flag
                                   RND3224
                 CALL
                 RETLW
                                   0x00
        minimax rational coefficients for log2(1+z)/z on [1/sqrt(2)-1,0]
LOG24HP0
                 EQU
                                   0x81
                                                    ; LOG24HP0 = .73551298732E+1
LOG24HP00
                 EQU
                                   0x6B
                                   0x5D
LOG24HP01
                 EOU
LOG24HP02
                 EQU
                                   0x39
LOG24HP1
                 EQU
                                   0x81
                                                     ; LOG24HP1 = .40900513905E+1
                                   0x02
LOG24HP10
                 EQU
                                   0xE1
LOG24HP11
                 EQU
LOG24HP12
                 EQU
                                   0xB3
LOG24HQ0
                 EQU
                                   0x81
                                                     ; LOG24HQ0 = .50982159260E+1
LOG24HQ00
                 EQU
                                   0x23
LOG24HQ01
                                   0x24
                 EQU
LOG24HQ02
                 EQU
                                   0x96
                                   0x81
                                                     ; LOG24HQ1 = .53849258895E+1
LOG24HQ1
                 EQU
LOG24HQ10
                 EQU
                                   0x2C
LOG24HQ11
                 EQU
                                   0x51
LOG24HQ12
                                   0x50
                 EQU
                                                     i LOG24HQ2 = 1.0
LOG24HQ2
                 EQU
                                   0x7F
LOG24HQ20
                 EQU
                                   0x00
LOG24HO21
                 EQU
                                   0x00
LOG24HQ22
                                   0x00
                 EQU
        minimax rational coefficients for log2(1+z)/z on [0,.sqrt(2)-1]
LOG24LP0
                 EQU
                                   0x82
                                                     ; LOG24LP0 = .103115556038E+2
LOG24LP00
                                   0x24
                 EQU
                                   0xFC
LOG24LP01
                 EQU
LOG24LP02
                 EQU
                                   0x22
LOG24LP1
                 EQU
                                   0x81
                                                     ; LOG24LP1 = .457749066375E+1
LOG24LP10
                                   0x12
                 EQU
LOG24LP11
                                   0x7A
                 EQU
LOG24LP12
                                   0xCE
                 EQU
LOG24LQ0
                 EQU
                                   0x81
                                                     ; LOG24LQ0 = .714746549793E+1
LOG24LQ00
                 EQU
                                   0x64
LOG24LQ01
                 EQU
                                   0xB8
LOG24LQ02
                                   0x0A
                 EQU
LOG24LQ1
                                                     ; LOG24LQ1 = .674551124538E+1
                 EQU
                                   0x81
```

```
LOG24LO10
               EOU
                               0x57
LOG24LO11
               EOU
                               0xDB
LOG24LQ12
               EQU
                               0x3A
LOG24LQ2
                EQU
                               0x7F
                                               ; LOG24LQ2 = 1.0
T/0G24T/020
               EQU
                               0x00
LOG24LQ21
                               0x00
               EQU
LOG24LQ22
               EQU
                               0x00
Evaluate log10(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Use:
               CALL
                       LOG1024
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- LOG( AARG )
        Testing on [MINNUM, MAXNUM] from 100000 trials:
               min
                       max
                               mean
        Timing: 12
                       1457
                               1317.7 clks
               min
                       max
                               mean
                                       rms
        Error: -0x01
                       0x00
                               -0.32
                                       0.57
                                               nsb
        This approximation of the natural log function is based upon the
        expansion
               log10(x) = log10(2) * log2(x) = log10(2) * (n + log2(f))
        where .5 \ll f \ll 1 and n is an integer. The additional transformation
                       2 * f - 1, f < 1/sqrt(2), n = n - 1
                       f - 1, otherwise
        produces a naturally segmented representation of log2(1+z) on the
        intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
        approximations.
LOG1024
                CLRF
                               AARGB2.W
                                               ; clear next significant byte
                BTFSS
                               AARGB0,MSB
                                               ; test for negative argument
                CPFSGT
                               AEXP
                                               ; test for zero argument
                GOTO
                               DOMERR24
               MOVFP
                               FPFLAGS, WREG
                                               ; save rounding flag
               MOVWF
                               DARGB3
                BCF
                               FPFLAGS, RND
                                               ; disable rounding
               MOVFP
                               AEXP, WREG
                MOVPF
                               WREG, EARGB3
                MOVLW
                               EXPBIAS-1
                               EARGB3,F
                SUBWF
                MOVWF
                               AEXP
                MOVLW
                               0xF3
                                               ; .70710678118655 = 7E3504F3
                SUBWF
                               AARGB2,W
                MOVLW
                               0x04
```

```
SUBWFB
                                    AARGB1,W
                  MOVLW
                                    0x35
                  SUBWFB
                                    AARGB0,W
                                    _C
                  BTFSS
                  GOTO
                                    LOG1024L
         minimax rational approximation on [0,.sqrt(2)-1]
LOG1024H
                  MOVLW
                                    0x7F
                  MOVPF
                                    WREG, BEXP
                  CLRF
                                    BARGB0,F
                  CLRF
                                    BARGB1,F
                  CLRF
                                    BARGB2,F
                  CALL
                                    FPS32
                                    AEXP, WREG
                  MOVFP
                  MOVPF
                                    WREG, DEXP
                                    AARGB0,DARGB0
                  MOVPF
                  MOVPF
                                    AARGB1,DARGB1
                  MOVPF
                                    AARGB2, DARGB2
                  POLL124
                                    LOG24HQ,2,0
                  MOVFP
                                    AEXP, WREG
                                    WREG, CEXP
                  MOVPF
                  MOVPF
                                    AARGB0, CARGB0
                                    AARGB1, CARGB1
                  MOVPF
                  MOVPF
                                    AARGB2, CARGB2
                  MOVFP
                                    DEXP, WREG
                                    WREG, AEXP
                  MOVPF
                                    DARGB0, AARGB0
                  MOVFP
                  MOVFP
                                    DARGB1, AARGB1
                                    DARGB2, AARGB2
                  MOVFP
                  POL24
                                    LOG24HP,1,0
                  MOVFP
                                    CEXP, WREG
                                    WREG, BEXP
                  MOVPF
                  MOVFP
                                    CARGBO, WREG
                  MOVPF
                                    WREG, BARGB0
                  MOVFP
                                    CARGB1, WREG
                  MOVPF
                                    WREG, BARGB1
                                    CARGB2, WREG
                  MOVFP
                                    WREG, BARGB2
                  MOVPF
                  CALL
                                    FPD32
                  GOTO
                                    LOG1024OK
         minimax rational approximation on [1/sqrt(2)-1,0]
LOG1024L
                  INCF
                                    AEXP,F
                  MOVLW
                                    0x7F
                  MOVPF
                                    WREG, BEXP
                  CLRF
                                    BARGB0,F
                  CLRF
                                    BARGB1,F
                                    BARGB2,F
                  CLRF
                  CALL
                                    FPS32
                  DECF
                                    EARGB3,F
```

	MOVFP	AEXP, WREG
	MOVPF	WREG, DEXP
	MOVPF	AARGB0,DARGB0
	MOVPF	AARGB1,DARGB1
	MOVPF	AARGB2,DARGB2
	POLL124	LOG24LQ,2,0
	MOVFP	AEXP, WREG
	MOVPF	WREG, CEXP
	MOVPF	AARGB0,CARGB0
	MOVPF	AARGB1,CARGB1
	MOVPF	AARGB2,CARGB2
	MOVFP	DEXP,WREG
	MOVPF	WREG, AEXP
	MOVFP	DARGBO, AARGBO
	MOVFP	DARGB1,AARGB1
	MOVFP	DARGB2, AARGB2
	POL24	LOG24LP,1,0
	MOVFP	CEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	CARGB0, WREG
	MOVPF	WREG, BARGB0
	MOVFP	CARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	CARGB2,WREG
	MOVPF	WREG,BARGB2
	CALL	FPD32
LOG1024OK		
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGB0, WREG
	MOVPF	WREG,BARGB0
	MOVFP	DARGB1,WREG
	MOVPF	WREG,BARGB1
	MOVFP	DARGB2,WREG
	MOVPF	WREG,BARGB2
	CALL	FPM32
	MOVED	AEXP WREC
	MOVEP	AEXP, WREG
	MOVPF	WREG, DEXP
	MOVPF MOVPF	WREG,DEXP AARGB0,DARGB0
	MOVPF	WREG, DEXP
	MOVPF MOVPF MOVPF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2
	MOVPF MOVPF MOVPF CLRF	WREG,DEXP AARGB0,DARGB0 AARGB1,DARGB1 AARGB2,DARGB2 AARGB0,F
	MOVPF MOVPF MOVPF CLRF MOVFP	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2 AARGB0, F EARGB3, AARGB1
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2 AARGB0, F EARGB3, AARGB1 AARGB1, MSB
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2 AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2 AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL CLRF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL CLRF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F  DEXP, WREG
	MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL CLRF MOVFP MOVFP	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F  DEXP, WREG WREG, BEXP
	MOVPF MOVPF MOVPF MOVPF  CLRF MOVFP BTFSC SETF CALL CLRF  MOVFP MOVFP MOVPF MOVFP	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F  DEXP, WREG WREG, BEXP DARGB0, WREG
	MOVPF MOVPF MOVPF MOVPF  CLRF MOVFP BTFSC SETF CALL CLRF  MOVFP MOVFP MOVPF MOVPF MOVPF	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F  DEXP, WREG WREG, BEXP DARGB0, WREG WREG, BARGB0
	MOVPF MOVPF MOVPF MOVPF CLRF MOVFP BTFSC SETF CALL CLRF MOVFP MOVFP MOVPF MOVPF MOVPF MOVFP	WREG, DEXP AARGB0, DARGB0 AARGB1, DARGB1 AARGB2, DARGB2  AARGB0, F EARGB3, AARGB1 AARGB1, MSB AARGB0, F FL01624 AARGB2, F  DEXP, WREG WREG, BEXP DARGB0, WREG WREG, BARGB0 DARGB1, WREG

```
MOVPF
                            WREG, BARGB2
              CALL
                             FPA32
       fixed point multiplication by log10(2)
              MOVPF
                            AARGB0, EARGB3
              BSF
                            AARGB0,MSB
              MOVT W
                             0x9A
              MOVPF
                             WREG, BARGB0
                             0x20
              MOVLW
              MOVPF
                             WREG, BARGB1
              MOVLW
                             0x9B
              MOVPF
                            WREG, BARGB2
              CALL
                            FXM2424U
              DECF
                            AEXP,F
              BTFSC
                            AARGB0,MSB
              GOTO
                            LOG1024DONE
              RLCF
                             AARGB3,F
              RLCF
                            AARGB2,F
                            AARGB1,F
              RLCF
              RLCF
                            AARGB0,F
              DECF
                            AEXP,F
LOGIO24DONE
                            EARGB3,MSB
              BTFSS
              BCF
                            AARGB0, MSB
              BTFSS
                             DARGB3, RND
              RETLW
                             0x00
              BSF
                            FPFLAGS, RND
                                          ; restore rounding flag
              CALL
                            RND3224
              RETLW
                             0x00
Evaluate cos(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              CALL
                     COS24
       Use:
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Result: AARG <-- COS( AARG )
       Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
              min
                            mean
                     max
       Timing: 912
                     1618
                            1458.6 clks
              min
                     max
                            mean
                                    rms
       Error: -0x56
                     0x13
                            -7.05
                                    0.00
                                           nsb
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x \mod pi/4, with an additional variable j
;
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
```

```
sin(z) = z * p(z**2), cos(z) = q(z**2)
        where p and q are minimax polynomial approximations.
COS24
                MOVFP
                                 FPFLAGS, WREG
                                                  ; save rounding flag
                MOVWF
                                 DARGB3
                                  FPFLAGS, RND
                                                  ; disable rounding
                 BCF
                 CLRF
                                  CARGB3,F
                                                   ; initialize sign in CARGB3
                 BCF
                                  AARGB0,MSB
                                                   ; use |x|
                                  RRSINCOS24
                 CALL
RRCOS240K
                 RRCF
                                  EARGB3,W
                 XORWF
                                  EARGB3,W
                BTFSC
                                  WREG, LSB
                 GOTO
                                  COSZSIN24
                 CALL
                                  ZCOS24
                                  COSSIGN24
                 GOTO
COSZSIN24
                 CALL
                                  ZSIN24
COSSIGN24
                 BTFSC
                                  EARGB3, LSB+1
                 BTG
                                  CARGB3, MSB
                 BTFSC
                                  CARGB3,MSB
                 BTG
                                  AARGB0,MSB
                BTFSS
                                  DARGB3, RND
                RETLW
                                  0x00
                 BSF
                                  FPFLAGS, RND
                                                 ; restore rounding flag
                 CALL
                                  RND3224
                 RETLW
                                  0x00
        Evaluate sin(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
                         SIN24
                CALL
        Use:
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- SIN( AARG )
        Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
                min
                         max
                                 mean
        Timing: 942
                         1637
                                 1465.7 clks
                min
                         max
                                  mean
                                          rms
        Error: -0x56
                         0x13
                                  -7.13
                                          20.90
                                                  nsb
```

```
The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
;
;
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition z = x \mod pi/4, with an additional variable j
;
        indicating the correct octant, leading to the appropriate call
;
        to either the sine or cosine approximations
                sin(z) = z * p(z**2), cos(z) = q(z**2)
        where p and q are minimax polynomial approximations.
SIN24
                MOVFP
                                FPFLAGS, WREG
                                                ; save rounding flag
                MOVWF
                                DARGB3
                BCF
                                FPFLAGS, RND
                                                ; disable rounding
                CLRF
                                CARGB3,F
                                                ; initialize sign in CARGB3
                BTFSC
                                AARGB0,MSB
                                                ; toggle sign if x < 0
                                CARGB3, MSB
                BSF
                BCF
                                AARGB0,MSB
                                                ; use |x|
                CALL
                                RRSINCOS24
RRSIN240K
                RRCF
                                EARGB3,W
                XORWF
                                EARGB3,W
                BTFSC
                                WREG, LSB
                                SINZCOS24
                GOTO
                                ZSIN24
                CALL
                GOTO
                                SINSIGN24
SINZCOS24
                                ZCOS24
                CALL
SINSIGN24
                BTFSC
                                CARGB3, MSB
                                AARGB0,MSB
                \operatorname{BTG}
                BTFSS
                                DARGB3, RND
                RETLW
                                0x00
                BSF
                                FPFLAGS, RND
                                                ; restore rounding flag
                CALL
                                RND3224
                RETLW
                                0x00
Evaluate sin(x) and cos(x)
        Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Use:
                CALL
                        SINCOS24
        Output: 24 bit floating point numbers in AEXP, AARGBO, AARGB1 and
                BEXP, BARGB0, BARGB1
;
        Result: AARG <-- COS( AARG )
               BARG <-- SIN( AARG )
```

```
Testing on [-LOSSTHR, LOSSTHR] from 100000 trials:
                 min
                          max
                                   mean
        Timing: 1516
                          2248
                                   2128.2 clks
                 min
                          max
                                  mean
                                           rms
                -0x56
                                   -7.12
        Error:
                          0x13
                                           20.89
                                                    nsb
                                                             sine
                 -0x56
                          0x13
                                   -7.13
                                           20.90
                                                             cosine
        The actual argument x on [-LOSSTHR, LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition z = x \mod pi/4, with an additional variable j
        indicating the correct octant, leading to the appropriate call
        to either the sine or cosine approximations
                 sin(z) = z * p(z**2), cos(z) = q(z**2)
        where p and q are minimax polynomial approximations. In this case,
        only one range reduction is necessary.
SINCOS24
                                   FPFLAGS, WREG
                                                    ; save rounding flag
                 MOVFP
                 MOVWF
                                   DARGB3
                 BCF
                                   FPFLAGS, RND
                                                    ; disable rounding
                 MOVFP
                                   AEXP, WREG
                                                    ; save x in EARG
                 MOVWF
                                   EEXP
                 MOVPF
                                   AARGB0, EARGB0
                 MOVPF
                                   AARGB1, EARGB1
                 CLRF
                                   EARGB2,F
                 BCF
                                   AARGB0,MSB
                                                    ; use |x|
                 CLRF
                                   CARGB3,F
                                                    ; initialize sign in CARGB3
                                   RRSINCOS24
                                                    ; range reduction
                 CALL
                 MOVFP
                                   CARGB3, WREG
                                                    ; save sign from range reduction
                 MOVWF
                                   ZARGB3
                 BTFSC
                                   EARGB0, MSB
                                                    ; toggle sign if x < 0
                                   CARGB3, MSB
                 BTG
                 CALL
                                   RRSIN240K
                 BTFSC
                                   DARGB3, RND
                 CALL
                                   RND3224
                 MOVFP
                                   AEXP, WREG
                                                    ; save sin(x) in EARG
                 MOVWF
                                   EEXP
                 MOVPF
                                   AARGB0, EARGB0
                 MOVPF
                                   AARGB1, EARGB1
                 MOVPF
                                   AARGB2, EARGB2
                 MOVFP
                                   DEXP, WREG
                                                    ; restore z*z in AARG
                 MOVWF
                                   AEXP
                 MOVFP
                                   DARGB0, AARGB0
                 MOVFP
                                   DARGB1, AARGB1
                 MOVFP
                                   DARGB2, AARGB2
                 MOVFP
                                   ZARGB3, WREG
                                                    ; restore sign from range reduction
                 MOVWF
                                   CARGB3
```

```
CALL
                                  RRCOS240K
                                  EEXP, WREG
                 MOVFP
                                                   ; restore sin(x) in BARG
                 MOVPF
                                  WREG, BEXP
                 MOVFP
                                  EARGBO, WREG
                 MOVPF
                                  WREG, BARGBO
                 MOVFP
                                  EARGB1, WREG
                 MOVPF
                                  WREG, BARGB1
                 MOVFP
                                  EARGB2, WREG
                                  WREG, BARGB2
                 MOVPF
                 BTFSS
                                  DARGB3, RND
                 RETLW
                                  0x00
                                  FPFLAGS, RND
                 BSF
                                                   ; restore rounding flag
                 CALL
                                  RND3224
                 RETLW
                                  0x00
        Range reduction routine for trigonometric functions
;
        The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition
                 z = x \mod pi/4,
        produced by first evaluating y and j through the relations
                 y = floor(x/(pi/4)), j = y - 8*[y/8].
        where j equals the correct octant. For j odd, adding one to j
        and y eliminates the odd octants. Additional logic on j and the
;
        sign of the result leads to appropriate use of the sine or cosine
        routine in each case.
        The calculation of z is then obtained through a pseudo extended
        precision method
                 z = x \mod pi/4 = x - y*(pi/4) = ((x - p1*y)-p2*y)-p3*y
        where pi/4 = p1 + p2 + p3, with p1 close to pi/4 and p2 close to
        pi/4 - p1. The numbers p1 and p2 are chosen to have an exact
        machine representation with slightly more than the lower half of
        the mantissa bits zero, typically leading to no error in computing
        the terms in parenthesis. This calculation breaks down leading to
        a loss of precision for |x| > LOSSTHR = sqrt(2**24)*pi/4, or for |x|
        close to an integer multiple of pi/4. This loss threshold has been
        chosen based on the efficacy of this calculation, with a domain error
        reported if this threshold is exceeded.
RRSINCOS24
                 MOVFP
                                  AEXP, WREG
                                                   ; loss threshold check
                 SUBLW
                                  LOSSTHR24EXP
                 BTFSS
                                  _C
                                  DOMERR24
                 GOTO
                 BTFSS
                                  _{\rm Z}
                                  RRSINCOS24ARGOK
                 GOTO
                                  AARGB0, WREG
                 MOVFP
                 SUBLW
                                  LOSSTHR24B0
                 BTFSS
                                  C
                                  DOMERR24
                 GOTO
                 BTFSS
                 GOTO
                                  RRSINCOS24ARGOK
```

```
AARGB1, WREG
                  MOVFP
                  SUBLW
                                    LOSSTHR24B1
                  BTFSS
                                    _C
                                    DOMERR24
                  GOTO
RRSINCOS24ARGOK
                  MOVFP
                                    AEXP, WREG
                                                      ; save |x| in CARG
                  MOVPF
                                    WREG, CEXP
                                    AARGB0, CARGB0
                  MOVPF
                  MOVPF
                                    AARGB1, CARGB1
                  CLRF
                                    CARGB2,F
         fixed point multiplication by 4/pi
                  BSF
                                    AARGB0, MSB
                  MOVPF
                                    AARGB0, BARGB0
                  MOVPF
                                    AARGB1, BARGB1
                  MOVLW
                                                      ; 4/pi = 1.27323954474
                                    0xA2
                                    WREG, AARGB0
                  MOVPF
                  MOVLW
                                    0xF9
                  MOVPF
                                    WREG, AARGB1
                                    0x83
                  MOVLW
                  MOVPF
                                    WREG, AARGB2
                  CALL
                                    FXM2416U
                  INCF
                                    AEXP,F
                  BTFSC
                                    AARGB0,MSB
                  GOTO
                                    RRSINCOS24YOK
                                    AARGB3,F
                  RLCF
                                    AARGB2,F
                  RLCF
                  RLCF
                                    AARGB1,F
                  RLCF
                                    AARGB0,F
                  DECF
                                    AEXP,F
RRSINCOS24YOK
                  BCF
                                    AARGB0,MSB
                                                      ; y = [|x| * (4/pi)]
                  CALL
                                    INT3224
                  BTFSS
                                    AARGB2,LSB
                  GOTO
                                    SAVEY24
                  INCF
                                    AARGB2,F
                  CLRF
                                    WREG, F
                  ADDWFC
                                    AARGB1,F
                  ADDWFC
                                    AARGB0,F
SAVEY24
                  MOVPF
                                    AARGB0, DARGB0
                                                      ; save y in DARG
                                    AARGB1, DARGB1
                  MOVPF
                                    AARGB2,DARGB2
                  MOVPF
                  MOVLW
                                    0x07
                                                      ; j = y \mod 8
                  ANDWF
                                    AARGB2,F
                  MOVLW
                                    0x03
                                    AARGB2
                  CPFSGT
                  GOTO
                                    JOK24
                  BTG
                                    CARGB3,MSB
                  MOVLW
                                    0x04
                  SUBWF
                                    AARGB2,F
JOK24
                                                      ; save j in EARGB3
                  MOVPF
                                    AARGB2, EARGB3
```

```
MOVFP
                                   DARGB0, AARGB0
                                                     ; restore y to AARG
                 MOVFP
                                   DARGB1, AARGB1
                                   DARGB2, AARGB2
                 MOVFP
                 CALL
                                   FL02432
                 MOVFP
                                   AEXP, WREG
                 MOVPF
                                   WREG, DEXP
                                                     ; save y in DARG
                 BTFSC
                                   _Z
                                   RRSINCOS24ZEQX
                 GOTO
                                   AARGB0, DARGB0
                 MOVPF
                 MOVPF
                                   AARGB1, DARGB1
                 MOVPF
                                   AARGB2, DARGB2
;
        Cody-Waite extended precision calculation of |x| - y * pi/4 using
        fixed point multiplication. Since y >= 1, underflow is not possible
        in any of the products.
                 BSF
                                   AARGB0, MSB
                 MOVLW
                                   0xC9
                                                     ; - p1 = -.78515625
                 MOVPF
                                   WREG, BARGB0
                 CLRF
                                   BARGB1,F
                 CALL
                                   FXM2416U
                                   AARGB0,MSB
                 BTFSC
                 GOTO
                                   RRSINCOS24Z1OK
                 RLCF
                                   AARGB3,F
                 RLCF
                                   AARGB2,F
                 RLCF
                                   AARGB1,F
                 RLCF
                                   AARGB0,F
                                   AEXP,F
                 DECE
RRSINCOS24Z1OK
                 MOVFP
                                   CEXP, WREG
                                                     ; restore x to BARG
                 MOVPF
                                   WREG, BEXP
                 MOVED
                                   CARGBO, WREG
                                   WREG, BARGB0
                 MOVPF
                 MOVFP
                                   CARGB1, WREG
                 MOVPF
                                   WREG, BARGB1
                                   BARGB2,F
                 CLRF
                                   FPA32
                                                     ; z1 = |x| - y * (p1)
                 CALL
                 MOVFP
                                   AEXP, WREG
                                   WREG, CEXP
                                                     ; save z1 in CARG
                 MOVPF
                 MOVPF
                                   AARGB0, CARGB0
                 MOVPF
                                   AARGB1, CARGB1
                 MOVPF
                                   AARGB2, CARGB2
                                   DEXP, WREG
                 MOVFP
                 MOVPF
                                   WREG, AEXP
                 MOVFP
                                   DARGB0, AARGB0
                                                     ; restore y to AARG
                 MOVFP
                                   DARGB1, AARGB1
                                   DARGB2, AARGB2
                 MOVFP
                 BSF
                                   AARGB0,MSB
                                   0xFD
                                                     ; - p2 = -.00024187564849853515624
                 MOVLW
                                   WREG,BARGB0
                 MOVPF
                 MOVLW
                                   0xA0
                 MOVPF
                                   WREG, BARGB1
                                   FXM2416U
                 CALL
```

	MOVLW	0x0D - 1	
	DTECO	AADCDO MCD	
	BTFSC	AARGBO,MSB	
	GOTO	RRSINCOS24Z2OK	
	RLCF RLCF	AARGB3,F	
	RLCF	AARGB2,F AARGB1,F	
	RLCF	AARGB1,F	
	DECF	AEXP,F	
	DBC1	111111 / 1	
RRSINCOS24Z2OK			
	SUBWF	AEXP,F	
	MOVFP	CEXP, WREG	; restore z1 to BARG
	MOVPF	WREG, BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG,BARGB1	
	MOVFP	CARGB2, WREG	
	MOVPF	WREG,BARGB2	
	CALL	FPA32	; z2 = z1 - y * (p2)
	MOVFP	AEXP, WREG	
	MOVPF	WREG, CEXP	; save z2 in CARG
	MOVPF	AARGB0,CARGB0	
	MOVPF	AARGB1,CARGB1	
	MOVPF	AARGB2,CARGB2	
	MOVED	DEAD MDEG	
	MOVFP MOVPF	DEXP,WREG WREG,AEXP	
	MOVFP		; restore y to AARG
	MOVFP	DARGB1, AARGB1	/ Testore y to AARO
	MOVFP	DARGB2, AARGB2	
		Dimobb (immobb	
	BSF	AARGB0,MSB	
	MOVLW	0xA2	; - p3 = -3.77489497744597636E-8
	MOVPF	WREG,BARGB0	
	MOVLW	0x21	
	MOVPF	WREG,BARGB1	
	MOVLW	0x69	
	MOVPF	WREG,BARGB2	
	<b>03.7.7</b>		
	CALL	FXM2424U	
	MOVLW	0x19 - 1	
	MOVIM	UAI) - I	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS24Z3OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS24Z3OK			
	SUBWF	AEXP,F	
	MOMED	CEVD MDEC	
	MOVFP	CEXP, WREG	; restore z2 to BARG
	MOVPF	WREG, BEXP	
	MOVFP	CARGBO, WREG	
	MOVED	WREG, BARGBO	
	MOVFP	CARGB1,WREG	

```
WREG, BARGB1
                  MOVPF
                                    CARGB2, WREG
                  MOVFP
                  MOVPF
                                    WREG, BARGB2
                  CALL
                                    FPA32
                                                      z = z2 - y * (p3)
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, CEXP
                                                      ; save z in CARG
                  MOVPF
                                    AARGB0, CARGB0
                                    AARGB1,CARGB1
                  MOVPF
                  MOVPF
                                    AARGB2, CARGB2
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, BEXP
                                    AARGB0, BARGB0
                  MOVPF
                  MOVPF
                                    AARGB1,BARGB1
                  MOVPF
                                    AARGB2, BARGB2
                                                      ; z * z
                  CALL
                                    FPM32
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, DEXP
                                                      ; save z * z in DARG
                  MOVPF
                                    AARGB0, DARGB0
                                    AARGB1, DARGB1
                  MOVPF
                  MOVPF
                                    AARGB2,DARGB2
                  RETLW
                                    0x00
RRSINCOS24ZEOX
                                    CEXP, WREG
                  MOVFP
                  MOVPF
                                    WREG, AEXP
                  MOVFP
                                    CARGB0, AARGB0
                                    CARGB1, AARGB1
                  MOVFP
                                    CARGB2, AARGB2
                  MOVED
                  MOVFP
                                    AEXP, WREG
                  MOVPF
                                    WREG, BEXP
                                    AARGB0,BARGB0
                  MOVPF
                  MOVPF
                                    AARGB1,BARGB1
                  MOVPF
                                    AARGB2, BARGB2
                                                      ; z * z
                  CALL
                                    FPM32
                  MOVFP
                                    AEXP, WREG
                                    WREG, DEXP
                                                      ; save z * z in DARG
                  MOVPF
                  MOVPF
                                    AARGB0, DARGB0
                  MOVPF
                                    AARGB1, DARGB1
                                    AARGB2,DARGB2
                  MOVPF
                  RETLW
                                    0x00
         minimax polynomial approximation p(x**2) on [0,pi/4]
ZCOS24
                  POL24
                                    COS24,3,0
                  RETLW
                                    0x00
         minimax polynomial approximation x*p(x**2) on [0,pi/4]
ZSIN24
                  POL24
                                    SIN24,2,0
                                    CEXP, WREG
                  MOVFP
```

```
MOVPF
                             WREG, BEXP
              MOVFP
                             CARGBO, WREG
              MOVPF
                             WREG, BARGB0
              MOVFP
                             CARGB1, WREG
              MOVPF
                             WREG, BARGB1
              MOVED
                             CARGB2, WREG
              MOVPF
                             WREG, BARGB2
              CALL
                             FPM32
              RETLW
                             0x00
       minimax polynomial coefficients for \sin(z)/z = p(z^{**}2) on [0,pi/4]
SIN240
              EQU
                             0x7E
                                            ; LP0 = .73551298732E+1******
SIN2400
              EQU
                             0x7F
SIN2401
                             0xFF
              EQU
SIN2402
              EQU
                             0xAC
SIN241
              EQU
                             0x7C
                                            ; LP1 = .40900513905E+1
SIN2410
              EQU
                             0xAA
                             0x99
SIN2411
              EQU
SIN2412
                             0x9D
              EQU
SIN242
              EQU
                             0x78
                                            ; LQ0 = .50982159260E+1
                             0x05
SIN2420
              EQU
SIN2421
              EQU
                             0x10
SIN2422
              EQU
                             0x48
       minimax polynomial coefficients for cos(z) = q(z^{**}2) on [0,pi/4]
       with COS240 constrained to be 1.
                                            ; LP0 = .73551298732E+1*******
COS240
              EQU
                             0x7F
COS2400
              EQU
                             0x00
COS2401
                             0x00
              EQU
COS2402
              EQU
                             0x00
COS 241
              EQU
                             0x7D
                                           ; LP1 = .40900513905E+1
COS2410
              EQU
                             0xFF
COS2411
                             0xFF
              EQU
COS2412
              EQU
                             0xD0
                                            ; LQ0 = .50982159260E+1
COS 242
                             0x7A
              EQU
COS2420
              EQU
                             0x2A
COS2421
              EQU
                             0x9E
COS2422
              EQU
                             0x76
COS243
              EQU
                             0x75
                                           ; LQ1 = .53849258895E+1
COS2430
              EQU
                             0xB2
COS2431
              EQU
                             0x12
COS2432
              EQU
                             0xBF
Evaluate sqrt(x)
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
       Use:
              CALL
                      SQRT24
```

```
Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- SQRT( AARG )
        Testing on [0,MAXNUM] from 100000 trials:
                min
                         max
                                 mean
        Timing: 6
                                 292.7
                                          clks
                min
                         max
                                 mean
                                          rms
                         0 \times 05
        Error:
                -0x10
                                  -3.56
                                          5.20
                                                   nsb
        Range reduction for the square root function is naturally produced by
        the floating point representation,
;
                x = f * 2**e, where 1 <= f < 2,
;
        leading to the expression
                | sqrt(f) * 2**(e/2),e even sqrt(x) = |
                            sqrt(f) * sqrt(2) * 2**(e/2),e odd
;
        The approximation of sqrt(f) utilizes a table lookup of 16 bit zeroth
        degree minimax estimates of the square root as a seed to a single
        Newton-Raphson iteration,
                y = (y0 + f/y0)/2,
;
        where the precision of the result is guaranteed by the precision of the
        seed and the quadratic conversion of the method.
SORT24
                BTFSC
                                 AARGB0,MSB
                                                   ; test for negative argument
                GOTO
                                 DOMERR24
                CLRF
                                 AARGB2.W
                                                  ; return if argument zero
                 CPFSGT
                                 AEXP
                 RETLW
                                  0x00
                MOVED
                                 AEXP, WREG
                MOVPF
                                 WREG, CEXP
                                                   ; save x in CARG
                                 AARGB0, CARGB0
                MOVPF
                MOVPF
                                 AARGB1, CARGB1
                                                  ; save RND flag in DARGB3
                MOVFP
                                 FPFLAGS, WREG
                MOVPF
                                 WREG, DARGB3
                 BCF
                                  FPFLAGS, RND
                                                   ; disable rounding
                MOVLW
                                  EXPBIAS
                                                   ; initialize exponent
                MOVPF
                                  WREG, AEXP
        generation of y0 using 16 bit zeroth degree minimax approximations to the
        square root of AARG, with the top 8 explicit bits of AARG as a pointer.
                MOVLW
                                 HIGH (RATBL256M); access table for y0
                MOVWF
                                 TBLPTRH
                RLCF
                                 AARGB1.W
                RLCF
                                 AARGB0,W
                                 LOW (RATBL256M)
                ADDLW
                MOVWF
                                 TBLPTRL
                BTFSC
                                  _C
```

	INCF TABLRD TLRD TLRD	TBLPTRH,F 0,1,AARGB0 1,AARGB0 0,AARGB1	
	BTFSC GOTO	CEXP,LSB RRSOK24	; is CEXP even or odd?
; fixed	point multiplicat	ion by sqrt(2)	
	BSF	AARGB0,MSB	; make MSB explicit
	MOVLW	0xB5	; sqrt(2) = 1.41421356237
	MOVPF	WREG, BARGB0	-
	MOVLW	0x05	
	MOVPF	WREG,BARGB1	
	CALL	FXM1616U	
	INCF	AEXP,F	
	BTFSC	AARGB0,MSB	
	GOTO	RRSOK24	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSOK24			
RR50R24	BCF	AARGB0,MSB	; make MSB implicit
	MOVLW	EXPBIAS	; divide exponent by two
	ADDWF	CEXP, W	/ divide exponent by two
	RRCF	WREG, F	
	Idici	MICEO / I	
	MOVPF	WREG, AEXP	
	MOVPF	WREG, BEXP	
	MOVPF	WREG, DEXP	
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB0,BARGB0	
	MOVPF	AARGB1,BARGB1	
	MOVFP	CEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	CARGB0, AARGB0	
	MOVFP	CARGB1,AARGB1	
	CALL	FPD24	; Newton-Raphson iteration
	MOVFP	DEXP, WREG	
	MOVPF	WREG, BEXP	
	MOVFP	DARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	DARGB1,WREG	
	MOVPF	WREG,BARGB1	
	CLRF	BARGB2,F	
	RTF90	DARGB3,RND	
	BTFSC BSF	FPFLAGS, RND	; restore rounding flag
	CALL	FPA32	, rescore rounding riag
	CUILI	11772	

```
DECF
                                     AEXP, F
                  RETLW
                                     0 \times 00
         Zeroth degree minimax approximations to sqrt(f), with pointer from
         the 8 most significant explicit bits of f, the mantissa of x.
RATBL256M
                  DATA
                            0x001F
                  DATA
                            0x005F
                  DATA
                            0x009F
                  DATA
                            0 \times 00 DE
                  DATA
                            0x011E
                  DATA
                            0x015D
                  DATA
                            0x019D
                  DATA
                            0x01DC
                            0x021B
                  DATA
                            0x025A
                  DATA
                            0x0298
                  DATA
                  DATA
                            0x02D7
                  DATA
                            0x0316
                            0 \times 0354
                  DATA
                            0x0392
                  DATA
                            0x03D1
                  DATA
                  DATA
                            0x040F
                            0x044D
                  DATA
                            0x048B
                  DATA
                  DATA
                            0x04C8
                  DATA
                            0x0506
                  DATA
                            0x0544
                  DATA
                            0x0581
                            0x05BE
                  DATA
                  DATA
                            0 \times 05 FB
                  DATA
                            0x0639
                  DATA
                            0 \times 0675
                  DATA
                            0x06B2
                  DATA
                            0x06EF
                  DATA
                            0x072C
                  DATA
                            0x0768
                  DATA
                            0x07A5
                  DATA
                            0 \times 0.7 E1
                            0x081D
                  DATA
                            0x0859
                  DATA
                  DATA
                            0x0896
                  DATA
                            0x08D1
                            0x090D
                  DATA
                  DATA
                            0x0949
                  DATA
                            0x0985
                  DATA
                            0x09C0
                            0x09FC
                  DATA
                  DATA
                            0x0A37
                  DATA
                            0x0A72
                  DATA
                            0x0AAD
                  DATA
                            0x0AE8
                            0x0B23
                  DATA
                  DATA
                            0x0B5E
                  DATA
                            0x0B99
                  DATA
                            0x0BD3
                  DATA
                            0x0C0E
                  DATA
                            0x0C48
                            0x0C83
                  DATA
                  DATA
                            0x0CBD
```

DATA

DATA

0x0CF7 0x0D31

## **AN660**

DATA	0x0D6B
DATA	$0 \times 0 DA5$
DATA	0x0DDF
DATA	0x0E18
DATA	0x0E52
DATA	0x0E8C
DATA	0x0EC5
DATA	0x0EFE
DATA	0x0F38
DATA	0x0F71
DATA	$0 \times 0 \text{FAA}$
DATA	0x0FE3
DATA	0x101C
DATA	0x1055
DATA	0x108D
DATA	0x10C6
	0x10C0
DATA	
DATA	0x1137
DATA	0x116F
DATA	0x11A7
DATA	0x11E0
DATA	0x1218
DATA	0x1250
DATA	0x1288
DATA	0x12C0
DATA	0x12F7
DATA	0x132F
DATA	0x1367
DATA	0x139E
DATA	0x13D6
DATA	0x140D
DATA	$0 \times 1444$
DATA	0x147C
DATA	0x14B3
DATA	0x14EA
DATA	0x1521
DATA	0x1558
DATA	0x158E
DATA	0x15C5
DATA	0x15FC
DATA	0x1632
	0x1652
DATA	
DATA	0x169F
DATA	0x16D6
DATA	0x170C
DATA	0x1742
DATA	0x1778
DATA	
עיייע	0x17AE
DAIA	0x17AE 0x17E4
DATA DATA	0x17E4
DATA	0x17E4 0x181A
DATA DATA	0x17E4 0x181A 0x1850
DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886
DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB
DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1
DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927
DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C
DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x18850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991
DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x18850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC 0x1A31 0x1A66
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC 0x1A31 0x1A66 0x1A9B
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC 0x1A31 0x1A66 0x1A9B 0x1AD0
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC 0x1A31 0x1A66 0x1A9B 0x1AD0 0x1B05
DATA DATA DATA DATA DATA DATA DATA DATA	0x17E4 0x181A 0x1850 0x1886 0x18BB 0x18F1 0x1927 0x195C 0x1991 0x19C7 0x19FC 0x1A31 0x1A66 0x1A9B 0x1AD0

DATA	0x1BA3
DATA	0x1BD8
DATA	0x1C0C
DATA	0x1C41
DATA	0x1C75
DATA	0x1CAA
DATA	0x1CDE
DATA	0x1D12
DATA	0x1D46
DATA	0x1D7A
DATA	0x1DAE
DATA	0x1DE2
DATA	0x1E16
DATA	0x1E4A
DATA	0x1E7D
DATA	0x1EB1
DATA	0x1EE5 0x1F18
DATA DATA	0x1F16 0x1F4C
DATA	0x1F4C 0x1F7F
DATA	0x1F7F 0x1FB2
DATA	0x1FE6
DATA	0x2019
DATA	0x204C
DATA	0x207F
DATA	0x20B2
DATA	0x20E5
DATA	0x2118
DATA	0x214B
DATA	0x217E
DATA	0x21B0
DATA	0x21E3
DATA	0x2215
DATA	0x2248
DATA	0x227A
DATA	0x22AD
DATA	0x22DF
DATA	0x2311
DATA	0x2344
DATA	0x2376
DATA	0x23A8
DATA	0x23DA
DATA	0x240C
DATA	0x243E
DATA DATA	0x2470 0x24A1
DATA	0x24A1 0x24D3
DATA	0x24D3
DATA	0x2536
DATA	0x2568
DATA	0x2599
DATA	0x25CB
DATA	0x25FC
DATA	0x262E
DATA	0x265F
DATA	0x2690
DATA	0x26C1
DATA	0x26F2
DATA	0x2723
DATA	0x2754
DATA	0x2785
DATA	0x27B6
DATA	0x27E7
DATA	0x2818
DATA	0x2848
DATA	0x2879

## **AN660**

DATA	0x28AA
DATA	0x28DA
DATA	0x290B
DATA	0x293B
DATA	0x296B
DATA	0x299C
DATA	0x29CC
DATA	0x29FC
DATA	0x2A2C
DATA	0x2A5D
DATA	0x2A8D
DATA	0x2ABD
DATA	0x2AED
DATA	0x2B1C
DATA	0x2B4C
DATA	0x2B7C
DATA	0x2BAC
DATA	0x2BDC
DATA	0x2C0B
DATA	0x2C3B
DATA	0x2C6A
DATA	0x2C9A
DATA	0x2CC9
DATA	0x2CF9
DATA	0x2D28
DATA	0x2D57
DATA	0x2D87
DATA	0x2DB6
DATA	0x2DE5
DATA	0x2E14
DATA	0x2E43
DATA	0x2E72
DATA	0x2EA1
DATA	0x2ED0
DATA	0x2EFF
DATA	0x2F2D
DATA	0x2F5C
DATA	0x2F8B
DATA	0x2FB9
DATA	0x2FE8
DATA	0x3017
DATA	0x3045
DATA	0x3074
DATA	0x30A2
DATA	0x30D0
DATA	0x30FF
DATA	0x312D
DATA	0x315B
DATA DATA	0x3189
DATA	0x31B7 0x31E5
DATA	0x31E3
DATA	0x3213
DATA	0x3241 0x326F
DATA	0x320F
DATA	0x325B
DATA	0x32F9
DATA	0x32F9 $0x3327$
DATA	0x3327
DATA	0x3331
DATA	0x33B0
DATA	0x33DD
DATA	0x340B
DATA	0x3438
DATA	0x3466
DATA	0x3493

```
DATA
                          0x34C0
                 DATA
                          0x34EE
        Evaluate floor(x)
                 24 bit floating point number in AEXP, AARGBO, AARGB1
        Use:
                 CALL
                          FLOOR24
        Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
        Result: AARG <-- FLOOR( AARG )
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                 min
                          max
                                  mean
                                   30.11
        Timing: 18
                          39
                                           clks
                 min
                                   mean
                                           rms
        Error:
                 0x00
                          0x00
                                   0.0
                                           0.0
                                                    nsb
FLOOR24
                 CLRF
                                                    ; clear next significant byte
                                   AARGB2,W
                 CPFSGT
                                   AEXP
                                                    ; test for zero argument
                                   0x00
                 RETLW
                 MOVFP
                                   AARGB0, AARGB3
                                                    ; save mantissa
                 MOVFP
                                   AARGB1,AARGB4
                                  EXPBIAS
                                                    ; compute unbiased exponent
                 MOVLW
                                  AEXP,W
                 SUBWE
                 BTFSC
                                   WREG, MSB
                 GOTO
                                  FLOOR24ZERO
                 SUBLW
                                   0x10-1
                 MOVWF
                                   TEMPB0
                                                    ; save number of zero bits in TEMPBO
                 BTFSC
                                   WREG, LSB+3
                                                    ; divide by eight
                 GOTO
                                   FLOOR24MASKH
FLOOR24MASKL
                                   TBLPTRH, F
                 CLRF
                 MOVFP
                                   TEMPB0, WREG
                                                    ; get remainder for mask pointer
                                   0x07
                 ANDLW
                 ADDLW
                                   LOW (FLOOR24MASKTABLE)
                 MOVWF
                                   TBLPTRL
                                  HIGH (FLOOR24MASKTABLE); access table for F0
                 MOVLW
                 ADDWFC
                                  TBLPTRH, F
                 TABLRD
                                   0,1,WREG
                 TLRD
                                   0,WREG
                                  AARGB1,F
                 ANDWF
                                   AARGB0,MSB
                                                    ; if negative, round down
                 BTFSS
                                   0x00
                 RETLW
                                   AARGB7
                 MOVWF
                 MOVFP
                                   AARGB4, WREG
                                   AARGB1
                 CPFSEQ
                                   FLOOR24RNDL
                 GOTO
                 RETLW
                                   0x00
```

## **AN660**

FLOOR24RNDL			
	COMF	AARGB7,W	
	INCF	WREG,F	
	ADDWF	AARGB1,F	
	CLRF	WREG, F	
	ADDWFC	AARGB0,F	
	BTFSS	_C	; has rounding caused carryout?
	RETLW	0x00	
	RRCF	AARGB0,F	
	RRCF	AARGB1,F	
	INCFSZ	AEXP,F	; check for overflow
	RETLW	$0 \times 0 0$	
	GOTO	SETFOV24	
FLOOR24MASKH			
	CLRF	TBLPTRH,F	
	MOVFP	TEMPB0,WREG	; get remainder for mask pointer
	ANDLW	0x07	
		/	
	ADDLW	LOW (FLOOR24MASK	(TABLE)
	MOVWF	TBLPTRL	OMMADI D. A.
	MOVLW		SKTABLE); access table for F0
	ADDWFC	TBLPTRH, F	
	TABLRD	0,1,WREG	
	TLRD	0,WREG	
	ANDWF	AARGB0,F	
	CLRF	AARGB1,F	
	BTFSS	AARGB0,MSB	; if negative, round down
	RETLW	0x00	
	MOVWF	AARGB7	
	MOVFP	AARGB4,WREG	
	CPFSEQ	AARGB1	
	GOTO	FLOOR24RNDH	
	MOVFP	AARGB3,WREG	
	CPFSEQ	AARGB0	
	GOTO	FLOOR24RNDH	
	RETLW	0x00	
FLOOR24RNDH			
	COMF	AARGB7,W	
	INCF	WREG,F	
	ADDWF	AARGB0,F	
	BTFSS	_C	; has rounding caused carryout?
	RETLW	0x00	
	RRCF	AARGB0,F	
	RRCF	AARGB1,F	
	INCFSZ	AEXP,F	
	RETLW	$0 \times 0 0$	
	GOTO	SETFOV24	; check for overflow
FLOOR24ZERO			
LUCKATUEKU	BTFSC	AARGB0,MSB	
	GOTO	FLOOR24MINUSONE	
	CLRF	AEXP, F	
	CLRF	AARGB0,F	
	CLRF	AARGB1,F	
	RETLW	0x00	
FLOOR24MINUSONE		0.55	
	MOVLW	0x7F	
	MOVWF	AEXP	
	MOVIM	0x80	
	MOVWF	AARGB0	

```
CLRF
                                AARGB1,F
                RETLW
                                0x00
        table for least significant byte requiring masking, using pointer from
        the remainder of the number of zero bits divided by eight.
FLOOR24MASKTABLE
                DATA
                                OxFF
                DATA
                                0xFE
                DATA
                                0xFC
                DATA
                                0xF8
                DATA
                                0xF0
                                0xE0
                DATA
                DATA
                                0xC0
                DATA
                                0x80
                DATA
                                0x00
Floating Point Relation A < B
        Input:
               24 bit floating point number in AEXP, AARGBO, AARGB1
;
                24 bit floating point number in BEXP, BARGBO, BARGB1
        Use:
               CALL
                       TALTB24
;
        Output: logical result in WREG
;
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
        Result: if A < B TRUE, WREG = 0x01
                if A < B FALSE, WREG = 0 \times 00
                min
                        max
                                mean
        Timing: 8
                        27
                               11.6
                                       clks
TALTB24
                MOVFP
                                AARGB0, WREG
                                                ; test if signs opposite
                XORWF
                                BARGB0,W
                                WREG, MSB
                BTFSC
                                TALTB240
                GOTO
                BTFSC
                                AARGB0,MSB
                GOTO
                                TALTB24N
TALTB24P
                                AEXP, WREG
                MOVFP
                                               ; compare positive arguments
                SUBWF
                                BEXP,W
                BTFSS
                                _C
                RETLW
                                0x00
                BTFSS
                                _{\rm Z}
                RETLW
                                0x01
                MOVFP
                                AARGB0, WREG
                SUBWF
                                BARGB0,W
                BTFSS
                                _C
                RETLW
                                0x00
                BTFSS
                                _{\rm Z}
                RETLW
                                0x01
                MOVFP
                                AARGB1, WREG
                                BARGB1,W
                SUBWF
                BTFSS
                                _C
                RETLW
                                0x00
                BTFSS
                                _{\rm Z}
```

```
RETLW
                                   0x01
                 RETLW
                                   0x00
TALTB24N
                                   BEXP, WREG
                 MOVFP
                                                     ; compare negative arguments
                 SUBWF
                                   AEXP,W
                 BTFSS
                                   C
                                   0x00
                 RETLW
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
                 MOVFP
                                   BARGB0, WREG
                 SUBWF
                                   AARGB0,W
                 BTFSS
                                   _C
                 RETLW
                                   0x00
                 BTFSS
                                    _Z
                 RETLW
                                   0x01
                 MOVFP
                                   BARGB1, WREG
                                   AARGB1,W
                 SUBWF
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 BTFSS
                                   _Z
                 RETLW
                                   0x01
                                   0x00
                 RETLW
TALTB240
                 BTFSS
                                   BARGB0, MSB
                 RETLW
                                   0x01
                                   0x00
                 RETLW
        Floating Point Relation A <= B
        Input:
                 24 bit floating point number in AEXP, AARGBO, AARGB1
                 24 bit floating point number in BEXP, BARGBO, BARGB1
        Use:
                 CALL
                          TALEB24
        Output: logical result in WREG
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
        Result: if A <= B TRUE, WREG = 0x01
                 if A <= B FALSE, WREG = 0x00
                 min
                          max
                                   mean
        Timing: 8
                          25
                                   11.5
                                            clks
TALEB24
                 MOVFP
                                   AARGB0, WREG
                                                     ; test if signs opposite
                 XORWF
                                   BARGB0,W
                                   WREG, MSB
                 BTFSC
                                   TALEB240
                 GOTO
                 BTFSC
                                   AARGB0,MSB
                 GOTO
                                   TALEB24N
TALEB24P
                 MOVFP
                                   AEXP, WREG
                                                     ; compare positive arguments
                                   BEXP,W
                 SUBWF
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 BTFSS
                                    _{\rm Z}
                                   0x01
                 RETLW
                 MOVFP
                                   AARGB0, WREG
                                   BARGB0,W
                 SUBWF
```

```
_C
                BTFSS
                                0x00
                RETLW
                BTFSS
                                _z
                                0x01
                RETLW
                MOVFP
                                AARGB1, WREG
                                BARGB1,W
                SUBWF
                BTFSS
                                _C
                                0x00
                RETLW
                RETLW
                                0x01
TALEB24N
                MOVFP
                                BEXP, WREG
                                                ; compare negative arguments
                SUBWF
                                AEXP,W
                                _C
                BTFSS
                                0x00
                RETIW
                BTFSS
                                _Z
                RETLW
                                0x01
                                BARGB0, WREG
                MOVFP
                SUBWF
                                AARGB0,W
                BTFSS
                                C
                RETLW
                                0x00
                BTFSS
                                _Z
                                0x01
                RETLW
                MOVFP
                                BARGB1, WREG
                SUBWF
                                AARGB1,W
                BTFSS
                                _C
                RETLW
                                0x00
                                0x01
                RETLW
TALEB240
                BTFSS
                                BARGB0, MSB
                                0x01
                RETLW
                RETLW
                                0x00
Floating Point Relation A > B
                24 bit floating point number in AEXP, AARGBO, AARGB1
                24 bit floating point number in BEXP, BARGBO, BARGB1
;
        Use:
                CALL
                        TAGTB24
        Output: logical result in WREG
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
;
;
        Result: if A > B TRUE, WREG = 0 \times 01
                if A > B FALSE, WREG = 0x00
                min
                        max
                                mean
        Timing: 8
                        27
                                11.5
                                        clks
TAGTB24
                MOVFP
                                BARGB0, WREG
                                                ; test if signs opposite
                                AARGB0,W
                XORWF
                BTFSC
                                WREG, MSB
                GOTO
                                TAGTB240
                BTFSC
                                BARGB0, MSB
                GOTO
                                TAGTB24N
TAGTB24P
                MOVFP
                                BEXP, WREG
                                                ; compare positive arguments
                SUBWF
                                AEXP,W
                BTFSS
                                _C
```

```
RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                                    BARGB0, WREG
                  MOVFP
                  SUBWE
                                    AARGB0,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  MOVFP
                                    BARGB1, WREG
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  RETLW
                                    0x00
TAGTB24N
                  MOVFP
                                    AEXP, WREG
                                                       ; compare negative arguments
                                    BEXP,W
                  SUBWF
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  MOVFP
                                    AARGB0, WREG
                                    BARGB0,W
                  SUBWF
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _Z
                  RETLW
                                    0x01
                  MOVFP
                                    AARGB1, WREG
                                    BARGB1,W
                  SUBWF
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0x01
                  RETLW
                                    0x00
TAGTB240
                                    AARGB0,MSB
                  BTFSS
                                    0x01
                  RETLW
                  RETLW
                                    0x00
         Floating Point Relation A >= B
                  24 bit floating point number in AEXP, AARGBO, AARGB1
                  24 bit floating point number in BEXP, BARGBO, BARGB1
                  CALL
         Use:
                           TAGEB24
         Output: logical result in WREG
         Testing on [-MAXNUM, MAXNUM] from 100000 trials:
         Result: if A >= B TRUE, WREG = 0x01
                  if A >= B FALSE, WREG = 0 \times 00
                  min
                           max
                                    mean
         Timing: 8
                                    11.5
                                             clks
TAGEB24
                  MOVFP
                                    BARGB0, WREG
                                                       ; test if signs opposite
```

```
XORWF
                                    AARGB0,W
                  BTFSC
                                    WREG, MSB
                  GOTO
                                    TAGEB240
                                    BARGB0, MSB
                  BTFSC
                  GOTO
                                    TAGEB24N
TAGEB24P
                  MOVFP
                                    BEXP, WREG
                                                      ; compare positive arguments
                  SUBWF
                                    AEXP,W
                                    _C
                  BTFSS
                  RETLW
                                    0x00
                                    _z
                  BTFSS
                                    0x01
                  RETLW
                                    BARGB0, WREG
                  MOVFP
                  SUBWF
                                    AARGB0,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0 \times 01
                  MOVFP
                                    BARGB1, WREG
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  RETLW
                                    0 \times 01
TAGEB24N
                                    AEXP, WREG
                                                      ; compare negative arguments
                  MOVFP
                                    BEXP,W
                  SUBWE
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _Z
                                    0x01
                  RETLW
                  MOVFP
                                    AARGB0, WREG
                  SUBWF
                                    BARGB0,W
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _Z
                  RETLW
                                    0x01
                                    AARGB1, WREG
                  MOVFP
                                    BARGB1,W
                  SUBWF
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                                    0x01
                  RETLW
TAGEB240
                                    AARGB0,MSB
                  BTFSS
                  RETLW
                                    0x01
                  RETLW
                                    0x00
         Floating Point Relation A == B
                  24 bit floating point number in AEXP, AARGBO, AARGB1
;
                  24 bit floating point number in BEXP, BARGBO, BARGB1
         Use:
                  CALL
                           TAEQB24
         Output: logical result in WREG
```

```
Testing on [-MAXNUM, MAXNUM] from 100000 trials:
       Result: if A == B TRUE, WREG = 0x01
              if A == B FALSE, WREG = 0 \times 00
              min
                     max
                           mean
      Timing: 4
                           6.0
                                   clks
TAEQB24
              MOVFP
                           AEXP, WREG
              CPFSEO
                           BEXP
              RETLW
                           0x00
              MOVFP
                           AARGB0, WREG
              CPFSEQ
                           BARGB0
              RETLW
                            0x00
                           AARGB1,WREG
              MOVFP
              CPFSEQ
                           BARGB1
              RETLW
                            0x00
              RETLW
                            0x01
Floating Point Relation A =! B
       Input: 24 bit floating point number in AEXP, AARGBO, AARGB1
              24 bit floating point number in BEXP, BARGBO, BARGB1
      Use:
             CALL
                    TANEB24
      Output: logical result in WREG
      Testing on [-MAXNUM, MAXNUM] from 100000 trials:
       Result: if A = ! B TRUE, WREG = 0x01
              if A =! B FALSE, WREG = 0 \times 00
              min
                     max
                          mean
      Timing: 4
                    11
                           6.0
                                   clks
TANEB24
              MOVFP
                           AEXP, WREG
              CPFSEQ
                           BEXP
              RETLW
                           0 \times 01
              MOVFP
                           AARGB0, WREG
                           BARGB0
              CPFSEQ
              RETLW
                           0x01
              MOVFP
                           AARGB1,WREG
              CPFSEQ
                            BARGB1
              RETLW
                            0x01
              RETLW
                            0x00
Nearest neighbor rounding
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Use:
             CALL
                    RND3224
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
      Result: AARG <-- RND( AARG )
```

```
Testing on [MINNUM, MAXNUM] from 10000 trials:
              min
                      max
                             mean
       Timing: 3
                                    clks
                      21
              min
                      max
                             mean
       Error: 0
                                    nsb
RND3224
              BTFSS
                             AARGB2,MSB
                                           ; is NSB < 0x80?
              RETLW
                             0x00
                             _C
                                            ; set carry for rounding
              BSF
              MOVLW
                             0x80
              CPFSGT
                             AARGB2
                                            ; select even if NSB = 0x80
              RRCF
                             AARGB1,W
              MOVPF
                             AARGB0,SIGN
                                           ; save sign
                             AARGB0,MSB
                                            ; make MSB explicit
              BSF
              CLRF
                             WREG,F
                                            ; round
              ADDWFC
                             AARGB1,F
              ADDWFC
                             AARGB0,F
              BTFSS
                             _C
                                            ; has rounding caused carryout?
                             RND32240K
              GOTO
                             ACCB0, F
              RRCF
                                            ; if so, right shift
                             ACCB1, F
              RRCF
              INFSNZ
                             EXP, F
                                            ; test for floating point overflow
              GOTO
                             SETFOV24
RND32240K
              BTFSS
                             SIGN, MSB
                             AARGB0,MSB
              BCF
                                            ; clear sign bit if positive
              RETLW
                             0x00
```



**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 F: PIC17CXXX 32-BIT ELEMENTARY FUNCTION LIBRARY

```
RCS Header $Id: ef32.a17 1.61 1997/03/11 15:48:45 F.J.Testa Exp $
       $Revision: 1.61 $
       PIC17 32-BIT ELEMENTARY FUNCTION LIBRARY
       All routines return WREG = 0x00 for successful completion, and WREG = 0xFF
       for an error condition specified in FPFLAGS.
       Test statistics are typically from 100000 trials, with timing in cycles
;
       and error in the next significant byte. In almost all cases, the floating
       point routines satisfy a unit in the last position (1*ulp) accuracy
;
       requirement, resulting in |nsb error| <= 0xFF. The integer and logical
;
       routines are exact.
;
       Routine Function
                             Timing in cycles
                                                    Error in nsb
                             min
                                    max mean
                                                    min
                                                           max
                                                                   mean
                                                                          rms
       SQRT32 32 bit sqrt(x) 10
                                     568
                                            494.0
                                                    -0x41
                                                           0x41
                                                                   0.04
                                                                          36.87
       EXP32
             32 bit exp(x)
                            14
                                     2024 1834.7 -0xA2
                                                           0x9A
                                                                   2.20
                                                                          29.18
       EXP1032 32 bit exp10(x) 14
                                     2084
                                          1845.3 - 0 \times 69
                                                           0xD9
                                                                   21.72
                                                                         39.44
       LOG32
              32 bit log(x) 12
                                     2147
                                          1985.0 -0x01 0x02
                                                                   0.55
                                                                          0.77
                                                           0 \times 02
       LOG1032 32 bit log10(x) 2001
                                     2308
                                          2135.9 - 0 \times 01
                                                                   -0.11
                                                                         0.60
       SIN32
             32 bit sin(x)
                           1338
                                     2408
                                            2182.5 -0x182 0x18D
                                                                   -0.91
                                                                         62.74
       COS32 32 bit cos(x) 1256
                                     2405
                                          2182.6 -0x19A 0x148
                                                                   -1.20
                                                                          62.83
       SINCOS3232 bit cos(x), sin(x) 23283432
                                          3217.8 -0x19A 0x148
                                                                   -1.20
                                                                         62.83
                                                    -0x182 0x18D
                                                                   -0.91
                                                                         62.74
       POW24 24 bit pow(x,y)=x**y 28524255
                                          3915.7 -0x6B
                                                           0 \times 77
                                                                   -0.48 16.49
       POW32
             32 bit pow(x,y)=x**y4280 5574
                                          5168.4 -0x270 0x209
                                                                   8.94
                                                                          92.21
       FLOOR32 32 bit floor(x) 30
                                    45
                                            35.2
                                                    0 \times 00
                                                           0 \times 00
                                                                   0.0
                                                                         0.0
       RAND32 32 bit rand(x) 117
                                    117
                                            117
       TALTB32 32 bit A < B 8
                                    33
                                            11.6
       TALEB32 32 bit A <= B
                                     31
                                            11.6
       TAGTB32 32 bit A > B
                           8
                                     33
                                            11.6
       TAGEB32 32 bit A >= B
                                     31
                                            11.6
       TAEQB32 32 bit A == B
                                     14
                                            5.9
       TANEB32 32 bit A != B
                                     14
                                            5.9
```

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
                                                     xxxxxxx
                     RADIX
                     POINT
polynomial evaluation macros
POLL132 macro
                       COF, N, ROUND
        32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
       with leading coefficient of one, and where AARG is assumed have been be saved
       in DARG. The result is in AARG.
       ROUND = One rounding is enabled; can be previously enabled
       ROUND = 1rounding is enabled
       ROUND = 2rounding is enabled then disabled before last add
       ROUND = 3rounding is assumed disabled then enabled before last add
       ROUND = 4rounding is assumed enabled and then disabled before last
               add if DARGB3,RND is clear
       ROUND = 5rounding is assumed disabled and then enabled before last
               add if DARGB3,RND is set
       local
             i.i
       variablei = N, j = 0
       variablei = i - 1
        if
               ROUND == 1 || ROUND == 2
               BSF
                               FPFLAGS, RND
        endif
               MOVLW
                               COF#v(i)
               MOVWF
                               BEXP
       variable j = 0
       while
               j <= 2
               MOVLW
                               COF#v(i)#v(j)
               MOVWF
                               BARGB#v(j)
       variable j = j + 1
```

```
endw
        CALL
                          FPA32
variablei = i - 1
while
      i >= 0
                          DEXP, WREG
        MOVFP
        MOVPF
                          WREG, BEXP
        MOVFP
                          DARGB0, WREG
        MOVPF
                          WREG, BARGB0
                          DARGB1, WREG
        MOVFP
                          WREG, BARGB1
        MOVPF
                          DARGB2, WREG
        MOVFP
        MOVPF
                          WREG, BARGB2
        CALL
                          FPM32
        MOVLW
                          COF#v(i)
        MOVWF
                          BEXP
variable j = 0
while
        j <= 2
        MOVLW
                          COF#v(i)#v(j)
        MOVWF
                         BARGB#v(j)
variablej = j + 1
endw
if
        i == 0
        if
                 ROUND == 2
        BCF
                         FPFLAGS, RND
        endif
                 ROUND == 3
        if
        BSF
                        FPFLAGS, RND
        endif
        if
                 ROUND == 4
        BTFSS
                         DARGB3, RND
        BCF
                          FPFLAGS, RND
        endif
        if
                 ROUND == 5
                         DARGB3,RND
        BTFSC
        BSF
                          FPFLAGS, RND
        endif
endif
        CALL
                         FPA32
variablei = i - 1
```

```
endw
        endm
POL32
        macro
                         COF, N, ROUND
        32 bit evaluation of polynomial of degree N, PN(AARG), with coefficients COF,
        and where AARG is assumed have been be saved in DARG. The result is in AARG.
        ROUND = Ono rounding is enabled; can be previously enabled
        ROUND = 1rounding is enabled
        ROUND = 2rounding is enabled then disabled before last add
        ROUND = 3rounding is assumed disabled then enabled before last add
        ROUND = 4rounding is assumed enabled and then disabled before last
                         add if DARGB3,RND is clear
        ROUND = 5rounding is assumed disabled and then enabled before last
                         add if DARGB3,RND is set
        local
                i,j
        variablei = N, j = 0
                ROUND == 1 || ROUND == 2
        if
                 BSF
                                 FPFLAGS, RND
        endif
                MOVLW
                                 COF#v(i)
                MOVWF
                                  BEXP
                j <= 2
        while
                MOVLW
                                 COF#v(i)#v(j)
                MOVWF
                                 BARGB#v(j)
        variable j = j + 1
        endw
                 CALL
                                 FPM32
        variablei = i - 1
                MOVLW
                                  COF#v(i)
                MOVWF
                                  BEXP
        variable j = 0
               j <= 2
        while
                MOVLW
                                  COF#v(i)#v(j)
                MOVWF
                                  BARGB#v(j)
        variable j = j + 1
        endw
                 CALL
                                  FPA32
        variable i = i - 1
        while
               i >= 0
                MOVFP
                                  DEXP, WREG
```

```
MOVPF
                           WREG, BEXP
         MOVFP
                           DARGB0, WREG
         MOVPF
                           WREG, BARGB0
                           DARGB1,WREG
         {\tt MOVFP}
                           WREG, BARGB1
         MOVPF
                           DARGB2, WREG
         MOVFP
         MOVPF
                           WREG, BARGB2
         CALL
                           FPM32
         MOVLW
                           COF#v(i)
         MOVWF
                            BEXP
variable j = 0
while
         j <= 2
         MOVLW
                           COF#v(i)#v(j)
        MOVWF
                           BARGB#v(j)
variable j = j + 1
{\tt endw}
if
         i == 0
         if
                  ROUND == 2
         BCF
                          FPFLAGS, RND
         {\tt endif}
         if
                  ROUND == 3
         BSF
                          FPFLAGS, RND
         {\tt endif}
                  ROUND == 4
         if
                           DARGB3, RND
         BTFSS
         BCF
                           FPFLAGS, RND
         endif
         if
                  ROUND == 5
         BTFSC
                           DARGB3,RND
         BSF
                           FPFLAGS, RND
         {\tt endif}
endif
         CALL
                          FPA32
variablei = i - 1
endw
{\tt endm}
```

```
*************************
        Evaluate exp(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                CALL
                        EXP32
        Use:
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- EXP( AARG )
        Testing on [MINLOG, MAXLOG] from 100000 trials:
                min
                        max
                                mean
        Timing: 14
                        2024
                                1834.7 clks
                min
                        max
                                 mean
                                         rms
        Error: -0xA2
                        0x9A
                                 2.20
                                         29.18
        This approximation of the exponential function is based upon the
        expansion
                \exp(x) = e^{*x} = e^{*x}(z + n*\log(2)) = e^{*x}z * 2**n,
        where -\log(2)/2 \ll z \ll \log(2)/2 and n is an integer, evaluated during
        range reduction. Segmented fifth degree minimax polynomial approximations
        are used to estimate e^*z on the intervals [-\log(2)/2,0] and [0,\log(2)/2].
EXP32
                                                 ; test for |x| < 2**(-32)/2
                                 0x5E
                MOVLW
                CPESGT
                                EXP
                GOTO
                                EXP32ONE
                                                 ; return e^*x = 1
                BTFSC
                                AARGB0,MSB
                                                 ; determine sign
                GOTO
                                TNEXP32
TPEXP32
                                AEXP, WREG
                                                 ; positive domain check
                MOVFP
                SUBLW
                                 MAXLOG32EXP
                BTFSS
                                 DOMERR32
                COTO
                BTFSS
                                 Z
                                 EXP32ARGOK
                GOTO
                MOVFP
                                 AARGB0, WREG
                                 MAXLOG32B0
                SUBLW
                BTFSS
                                 _C
                GOTO
                                 DOMERR32
                BTFSS
                                 _{\rm Z}
                                 EXP32ARGOK
                GOTO
                                 AARGB1, WREG
                MOVFP
                SUBLW
                                 MAXLOG32B1
                BTFSS
                                 _C
                                 DOMERR32
                GOTO
                BTFSS
                                 _{\rm Z}
                                 EXP32ARGOK
                GOTO
                                 AARGB2, WREG
                MOVFP
                SUBLW
                                 MAXLOG32B2
                BTFSS
                                 C
                                 DOMERR32
                GOTO
                GOTO
                                 EXP32ARGOK
```

TNEXP32			
	MOVFP	AEXP, WREG	; negative domain check
	SUBLW	MINLOG32EXP	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	MOVFP	AARGB0,WREG	
	SUBLW	MINLOG32B0	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	MOVFP	AARGB1,WREG	
	SUBLW	MINLOG32B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP32ARGOK	
	MOVFP	AARGB2,WREG	
	SUBLW	MINLOG32B2	
	BTFSS	_C	
	GOTO	DOMERR32	
EXP32ARGOK			
	MOVFP		; save RND flag
	MOVWF	DARGB3	
	DOE	EDELAGG DND	· analala wawadina
	BSF	FPFLAGS, RND	; enable rounding
	CALL	RREXP32	
	BTFSC	DARGB0,MSB	
	GOTO	EXP32L	
EXP32H			
	POL32	EXP32H,5,4	; minimax approximation on [0,log(2)/2]
	GOTO	EXP32OK	
EXP32L			
	POL32	EXP32L,5,4	<pre>; minimax approximation on [-log(2)/2,0]</pre>
EXP32OK			
	MOVFP	EARGB3,WREG	
	ADDWF	AEXP,F	
	RETLW	0x00	
EXD 3 COME	MONTEN	EVDDIAG	1 0
EXP32ONE	MOVINE	EXPBIAS	; return $e^*x = 1.0$
	MOVWF	AEXP	
	CLRF	AARGB0,F	
	CLRF	AARGB1,F	
	CLRF	AARGB2,F	
	CLRF RETLW	AARGB3,F 0x00	
	TO T TIM	0.000	
DOMERR32	BSF	FPFLAGS,DOM	; domain error
	RETLW	0xFF	
	**		

```
*******************************
        Range reduction routine for the exponential function
        The evaluation of z and n through the decomposition
                x = z + n*log(2)
        is performed by first evaluating n through the relation
                n = floor(x*log2(e) + .5)
        The calculation of z is then obtained through a pseudo extended
        precision method
                z = x - n*log(2) = (x - n*c1) + n*c2
        where c1 is close to log(2) and has an exact machine representation,
        typically leading to no error in computing the term in parenthesis.
RREXP32
                MOVFP
                                 AEXP, WREG
                MOVPF
                                 WREG, CEXP
                                                  ; save x in CARG
                                 AARGB0, CARGB0
                MOVPF
                MOVPF
                                 AARGB1, CARGB1
                                 AARGB2, CARGB2
                MOVPF
                                 AARGB0,MSB
                BSF
                MOVPF
                                 AARGB0, BARGB0
                MOVPF
                                 AARGB1,BARGB1
                MOVPF
                                 AARGB2,BARGB2
                MOVLW
                                 0xB8
                                                 i = 1/\ln(2) = 1.44269504089
                MOVPF
                                 WREG, AARGB0
                MOVLW
                                 0xAA
                MOVPF
                                 WREG, AARGB1
                MOVLW
                                 0x3B
                MOVPF
                                 WREG, AARGB2
                                 0x29
                MOVLW
                MOVPF
                                 WREG, AARGB3
                                 FXM3224U
                                                 ; x * (1/ln2)
                CALL
                INCF
                                 AEXP,F
                BTFSC
                                 AARGB0, MSB
                                 RREXP32YOK
                GOTO
                RLCF
                                 AARGB3,F
                RLCF
                                 AARGB2,F
                RLCF
                                 AARGB1,F
                RLCF
                                 AARGB0,F
                DECF
                                 AEXP,F
RREXP32YOK
                BTFSS
                                 CARGB0, MSB
                BCF
                                 AARGB0, MSB
                CALL
                                 RND4032
                MOVLW
                                 0x7E
                                                 ; k = [x / ln2 + .5]
                MOVWF
                                 BEXP
                CLRF
                                 BARGB0,F
                CLRF
                                 BARGB1,F
                CLRF
                                 BARGB2,F
                                 FPA32
                CALL
```

	CALL	FLOOR32	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, EEXP	; save float k in EARG
	BTFSC	_Z	
	GOTO	RREXP32FEQX	
	MOVPF	AARGB0,EARGB0	
	MOVPF	AARGB1,EARGB1	
	MOVPF	AARGB2,EARGB2	
	BSF	AARGB0,MSB	
	MOVLW	0xB1	; c1 = .693359375
	MOVWF	BARGB0	
	MOVLW	0x80	
	MOVWF	BARGB1	
	CALL	FXM2416U	
	BTFSC	AARGB0,MSB	
	GOTO	RREXP32F1OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
	DECI	111111 / 1	
RREXP32F1OK	BTFSC	EARGB0,MSB	; make AARG negative
	BCF	AARGB0,MSB	
	MOVFP	CEXP, WREG	
	MOVPF	WREG, BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG, BARGBO	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG, BARGB1	
	MOVFP	CARGB2, WREG	
	MOVPF	WREG,BARGB2	
	CALL	FPA32	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, DEXP	; save fl in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	MOVLW	0xDE	; c2 = .00021219444005
	MOVWF	BARGB0	, 6200021219444005
	MOVLW	0x80	
	MOVWF	BARGB1	
		0x83	
	MOVLW MOVWF	BARGB2	
	PIOVWI	BARGBZ	
	MOVFP	EEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVLW	0x0D-1	
	SUBWF	AEXP,F	
	MOVFP	EARGB0,AARGB0	
	MOVFP	EARGB1,AARGB1	
	MOVFP	EARGB2,AARGB2	
	BSF	AARGB0,MSB	
	231	1111000   1100	
	CALL	FXM2424U	

	BTFSC	AARGB0,MSB	
	GOTO	RREXP32F2OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RREXP32F2OK	BTFSS	EARGB0,MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, BEXP	
	MOVFP	DARGB0,WREG	
	MOVPF	WREG, BARGB0	
	MOVFP	DARGB1,WREG	
	MOVPF	WREG, BARGB1	
	MOVFP	DARGB2,WREG	
	MOVPF	WREG,BARGB2	
	a	=== 20	
	CALL	FPA32	
	MOVFP	AEXP, WREG	
	MOVPF		; save f in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	MOVFP	EEXP, WREG	
	MOVWF	AEXP	
	MOVFP	EARGB0, AARGB0	
	MOVFP	EARGB1,AARGB1	
	BCF	FPFLAGS, RND	
	CALL	INT2416	; k = [x / ln2 + .5]
	BSF	FPFLAGS, RND	
		,	
	MOVPF	AARGB1,EARGB3	; save integer k in EARGB3
	MOVFP	DEXP, WREG	
	MOVWF	AEXP	; restore f in AARG
	MOVFP	DARGB0, AARGB0	/ Testore I III AARG
	MOVED	DARGBI, AARGBI	
	MOVFP	DARGB2,AARGB2	
	RETLW	0x00	
	KEIHW	0.000	
RREXP32FEQX			
THE SEE BOX	MOVFP	CEXP, WREG	
	MOVWF	DEXP	t many f
	MOVWF	AEXP	; save $f = x in DARG$ , AARG
	MOVFP	CARGB0, WREG	
	MOVWF	DARGB0	
	MOVWF	AARGB0	
	MOVFP	CARGB1,WREG	
	MOVWF	DARGB1	
	MOVWF	AARGB1	
	MOVFP	CARGB2, WREG	
	MOVWF	DARGB2	
	MOVWF	AARGB2	
	CLRF	EARGB3,F	

	RETL	0x00	
;			
;	fifth degree	minimax polynomia	.l coefficients for e**(x) on [0,(ln2)/2]
EXP32H0	EQU	0x7F	; EXP32H0 = 1.0
EXP32H00		0x00	/ EAF 32110 - 1.0
EXP32H00		0x00	
EXP32H01		0x00	
EAPJZHUZ	EQU EQU	0000	
EXP32H1	EQU	0x7F	; EXP32H1 = 1.00000025499
EXP32H10	EQU	0x00	
EXP32H11	EQU	0x00	
EXP32H12	EQU	0x02	
EXP32H2	EQU	0x7D	; EXP32H2 = .499991163105
EXP32H20		0x7F	
EXP32H21		0xFE	
EXP32H22		0xD7	
EXP32H3	EQU	0x7C	; EXP32H3 = .166777360103
EXP32H30		0x2A	
EXP32H31		0xC7	
EXP32H32	EQU	0xAF	
EXP32H4	EQU	0x7A	; EXP32H4 = .410473706887E-1
EXP32H40		0x28	
EXP32H41		0x21	
EXP32H42		0x4A	
EXP32H5	EQU	0x78	; EXP32H5 = .989943653774E-2
EXP32H50	EQU	0x22	
EXP32H51	EQU	0x31	
EXP32H52	EQU	0x3F	
;	fifth degree	minimax polynomia	.1 coefficients for $e^{**}(x)$ on $[-(ln2)/2,0]$
EXP32L0	EQU	0x7F	; EXP32L0 = 1.0
EXP32L00		0x00	/ EAF32D0 - 1.0
EXP32L01		0x00	
EXP32L02		0x00	
1111 J21102	120	02100	
EXP32L1	EQU	0x7E	; EXP32L1 = .999999766814
EXP32L10		0x7F	
EXP32L11	EQU	0xFF	
EXP32L12	EQU	0xFC	
EXP32L2	EQU	0x7D	; EXP32L2 = .499992371926
EXP32L20		0x7F	
EXP32L21	~	0xFF	
EXP32L22	~	0x00	
	~-		
EXP32L3	EQU	0x7C	; EXP32L3 = .166574299807
EXP32L30	~	0x2A	
EXP32L31	EQU	0x92	
EXP32L32	EQU	0x75	
EXP32L4	EQU	0x7A	; EXP32L4 = .411548782678E-1
EXP32L40		0x28	, Harrier 1110/020/05 1
EXP32L41		0x92	
EXP32L42		0x05	
		0203	

```
EXP32L5
                 EOU
                                  0x77
                                                   ; EXP32L5 = .699995870637E-2
EXP32L50
                 EOU
                                  0x65
EXP32L51
                 EQU
                                  0x5F
EXP32L52
                 EQU
                                  0xE9
        Evaluate expl0(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                         EXP1032
        Use:
                 CALL
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- EXP10( AARG )
        Testing on [MINLOG10, MAXLOG10] from 100000 trials:
                 min
                         max
                                  mean
        Timing: 14
                          2084
                                  1845.3 clks
                 min
                          max
                                  mean
                                           rms
                                  21.72
        Error: -0x69
                         0xD9
                                           39.44
                                                   nsb
        This approximation of the exponential function is based upon the
        expansion
                 expl0(x) = 10**x = 10**(z + n*log10(2)) = 10**z * 2**n,
        where -\log 10(2)/2 \le z \le \log 10(2)/2 and n is an integer, evaluated during
        range reduction. Segmented fifth degree minimax polynomial approximations
        are used to estimate 10**z on the intervals [-log10(2)/2,0] and [0,log10(2)/2].
EXP1032
                                                   ; test for |x| < 2**(-32)/2
                 MOVLW
                                  0x5E
                 CPESGT
                                  AEXP
                 GOTO
                                  EXP1032ONE
                                                    ; return 10**x = 1
                                  AARGB0,MSB
                 BTFSC
                                                    ; determine sign
                                  TNEXP1032
                 GOTO
TPEXP1032
                                  AEXP, WREG
                 MOVFP
                                                   ; positive domain check
                 SUBLW
                                  MAXLOG1032EXP
                 BTFSS
                                  _C
                                  DOMERR32
                 COTO
                 BTFSS
                                  _{\rm Z}
                 GOTO
                                  EXP1032ARGOK
                                  AARGB0, WREG
                 MOVFP
                                  MAXLOG1032B0
                 SUBLW
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
                 BTFSS
                                  _Z
                                  EXP1032ARGOK
                 GOTO
                                  AARGB1, WREG
                 MOVFP
                 SUBLW
                                  MAXLOG1032B1
                 BTFSS
                                  _C
                 GOTO
                                  DOMERR32
                 BTFSS
                                   Ζ
                                  EXP1032ARGOK
                 GOTO
                                  AARGB2, WREG
                 MOVFP
```

	SUBLW BTFSS	MAXLOG1032B2 _C	
	GOTO	DOMERR32	
	GOTO	EXP1032ARGOK	
TNEXP1032			
	MOVFP	AEXP, WREG	; negative domain check
	SUBLW	MINLOG1032EXP	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
	MOVFP	AARGB0,WREG	
	SUBLW	MINLOG1032B0	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
	MOVFP	AARGB1,WREG	
	SUBLW	MINLOG1032B1	
	BTFSS	_C	
	GOTO	DOMERR32	
	BTFSS	_Z	
	GOTO	EXP1032ARGOK	
	MOVFP	AARGB2,WREG	
	SUBLW	MINLOG1032B2	
	BTFSS	_C	
	GOTO	DOMERR32	
EXP1032ARGOK			
	MOVFP	FPFLAGS, WREG	; save RND flag
	MOVWF	DARGB3	
	BSF	FPFLAGS,RND	; enable rounding
	CALL	RREXP1032	
	BTFSC	DARGB0,MSB	
	GOTO	EXP1032L	
EXP1032H			
	POL32	EXP1032H,5,4	; minimax approximation on $[0,log10(2)/2]$
	GOTO	EXP1032OK	
EXP1032L	DOI 22	EVD10201 F 4	· minimor amounting on [ ]10/2//2 0]
	POL32	EXP1032L,5,4	; minimax approximation on [-log10(2)/2,0]
EXP1032OK			
	MOVFP	EARGB3,WREG	
	ADDWF	AEXP,F	
	RETLW	0x00	
EXP1032ONE	MOVLW	EXPBIAS	; return 10**x = 1.0
	MOVWF	AEXP	
	CLRF	AARGB0,F	
	CLRF	AARGB1,F	
	CLRF	AARGB2,F	
	CLRF	AARGB3,F	
	RETLW	0x00	
; * * * * * * * * * * * *	*******	******	*************

Range reduction routine for the exponential function

```
The evaluation of z and n through the decomposition
                 x = z + n*log10(2)
        is performed by first evaluating n through the relation
                 n = floor(x*log2(10) + .5)
        The calculation of z is then obtained through a pseudo extended
        precision method
                 z = x - n*log10(2) = (x - n*c1) - n*c2
        where c1 is close to log10(2) and has an exact machine representation,
        typically leading to no error in computing the term in parenthesis.
RREXP1032
                 MOVFP
                                  AEXP, WREG
                 MOVPF
                                   WREG, CEXP
                                                    ; save x in CARG
                 MOVPF
                                   AARGB0, CARGB0
                 MOVPF
                                   AARGB1, CARGB1
                 MOVPF
                                   AARGB2, CARGB2
                 BSF
                                   AARGB0,MSB
                 MOVPF
                                   AARGB0, BARGB0
                                   AARGB1,BARGB1
                 MOVPF
                 MOVPF
                                   AARGB2,BARGB2
                 MOVLW
                                   0xD4
                                                    ; 1/log10(2) = 3.32192809489
                 MOVPF
                                   WREG, AARGB0
                 MOVLW
                                   0x9A
                                   WREG, AARGB1
                 MOVPF
                 MOVLW
                                   0x78
                 MOVPF
                                   WREG, AARGB2
                 MOVLW
                                   0x47
                 MOVPF
                                   WREG, AARGB3
                 CALL
                                   FXM3224U
                                                    ; x * (1/log10(2))
                 INCF
                                   AEXP,F
                 INCF
                                   AEXP,F
                 BTFSC
                                   AARGB0, MSB
                 GOTO
                                   RREXP1032YOK
                 RLCF
                                   AARGB3,F
                 RLCF
                                   AARGB2,F
                 RLCF
                                   AARGB1,F
                 RLCF
                                   AARGB0,F
                 DECF
                                   AEXP,F
RREXP1032YOK
                 BTFSS
                                   CARGB0, MSB
                                   AARGB0,MSB
                 BCF
                 CALL
                                   RND4032
                                                    ; k = [x / log10(2) + .5]
                 MOVLW
                                   0x7E
                 MOVWF
                                   BEXP
                 CLRF
                                   BARGB0,F
                                   BARGB1,F
                 CLRF
                 CLRF
                                   BARGB2,F
                 CALL
                                   FPA32
                                   FLOOR32
                 CALL
```

	MOVFP	AEXP,WREG	
			floor lo de DADO
	MOVPF	WREG, EEXP	; save float k in EARG
	BTFSC	_Z	
	GOTO	RREXP1032FEQX	
	MOVPF	AARGB0,EARGB0	
	MOVPF	AARGB1,EARGB1	
	MOVPF	AARGB2,EARGB2	
	BSF	AARGB0,MSB	
	DOF	AARGBU, MSB	
	MOVLW	0x9A	; c1 = .301025390625
	MOVWF	BARGB0	
	MOVLW	0x20	
	MOVWF	BARGB1	
	DECF	AEXP,F	
	CALL	FXM2416U	
	CALL	120121100	
	BTFSC	AARGB0,MSB	
	GOTO	RREXP1032F10K	
		AARGB3,F	
	RLCF		
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RREXP1032F1OK	BTFSC	EARGB0,MSB	; make AARG negative
	BCF	AARGB0,MSB	
		•	
	MOLIED	CEVE LIBEC	
	MOVFP	CEXP, WREG	
	MOVPF	WREG,BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG, BARGBO	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG,BARGB1	
	MOVFP	CARGB2,WREG	
	MOVPF	WREG, BARGB2	
	110 111	Witze, Britesi	
	a	======================================	
	CALL	FPA32	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, DEXP	; save fl in DARG
			/ bave II III bake
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	MOVLW	0x9A	; c2 = 4.6050389811952113E-6
			7 62 - 4.00303696119321136-0
	MOVWF	BARGB0	
	MOVLW	0x84	
	MOVWF	BARGB1	
		0xFC	
	MOVLW		
	MOVWF	BARGB2	
	MOVFP	EEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVLW	0x12-1	
	SUBWF	AEXP,F	
	MOVFP	EARGB0, AARGB0	
	MOVFP	EARGB1,AARGB1	
	MOVFP	EARGB2, AARGB2	
	BSF	AARGB0,MSB	
	CALL	FXM2424U	
	Ç11111	2 11112 12 10	

	BTFSC	AARGB0,MSB	
	GOTO	RREXP1032F2OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RREXP1032F2OK	BTFSC	EARGB0,MSB	
1012111 10021 2011	BCF	AARGB0,MSB	
	201	1111020 / 1102	
	CALL	RND4032	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, BEXP	
	MOVFP	DARGBO, WREG	
	MOVPF	WREG, BARGB0	
	MOVFP	DARGB1, WREG	
	MOVPF	WREG, BARGB1	
	MOVFP	DARGB2,WREG	
	MOVPF	WREG, BARGB2	
	CALL	FPA32	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, DEXP	; save f in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	MOVFP	EEXP,WREG	
	MOVWF	AEXP	
	MOVFP	EARGB0, AARGB0	
	MOVFP	EARGB1,AARGB1	
	BCF	FPFLAGS,RND	
	CALL	INT2416	k = [x / log10(2) + .5]
	BSF	FPFLAGS, RND	/ K = [ K / 10910(2) 1 .5 ]
	DOI	I I I LAGO, IGID	
	MOVPF	AARGB1,EARGB3	; save integer k in EARGB3
	MOVFP	DEXP, WREG	
	MOVWF	AEXP	; restore f in AARG
	MOVFP	DARGB0, AARGB0	, repeate r in imme
	MOVFP	DARGB1, AARGB1	
	MOVFP	DARGB2, AARGB2	
		, -	
	RETLW	0x00	
RREXP1032FEQX			
10021 1011	MOVFP	CEXP, WREG	
	MOVWF	DEXP	
	MOVWF	AEXP	; save $f = x \text{ in DARG, AARG}$
	MOVFP	CARGB0, WREG	. 13.0 1 11 Dinto, Imito
	MOVWF	DARGB0	
	MOVWF	AARGB0	
	MOVFP	CARGB1,WREG	
	MOVWF	DARGB1, WREG	
	MOVWF	AARGB1	
	MOVFP	CARGB2, WREG	
	MOVWF	DARGB2	
	MOVWF	AARGB2	
	PIO V WIP	1 MICOLE	

	CLRF	EARGB3,F
	RETLW	0x00
;		
; fifth	degree minima	polynomial coefficients for $10**(x)$ on $[0,(log10(2))/2]$
EXP1032H0	EQU	0x7F ; $EXP1032H0 = 1.0$
EXP1032H00	EQU	0x $0$ 0
EXP1032H01	EQU	0x00
EXP1032H02	EQU	0x $0$ 0
EXP1032H1	EQU	0x80 ; EXP1032H1 = 2.302585504840E0
EXP1032H10	EQU	0x13
EXP1032H11	EQU	0x5D
EXP1032H12	EQU	0x90
EXP1032H2	EQU	0x80 ; EXP1032H2 = 2.650909138708E0
EXP1032H20	EQU	0x29
EXP1032H20 EXP1032H21	EQU	0x29 0xA8
EXP1032H21 EXP1032H22	EQU	0x7F
	~ -	
EXP1032H3	EQU	0x80 ; EXP1032H3 = 2.035920309947E0
EXP1032H30	EQU	0x02
EXP1032H31	EQU	0x4C
EXP1032H32	EQU	0x85
EXP1032H4	EQU	0x7F ; EXP1032H4 = 1.154596329197E0
EXP1032H40	EQU	0x13
EXP1032H41	EQU	0xC9
EXP1032H42	EQU	0xD0
EXP1032H5	EQU	0x7E ; EXP1032H5 = 6.388992868121E-1
EXP1032H50	EQU	0x23
EXP1032H51	EQU	0x8E
EXP1032H52	EQU	0xE7
; fifth	degree minima:	polynomial coefficients for $10**(x)$ on $[-(log10(2))/2,0]$
EXP1032L0	EQU	0x7F ; EXP1032L0 = 1.0
EXP1032L00	EQU	0x00
EXP1032L01	EQU	0x00
EXP1032L02	EQU	0x00
EXP1032L1	EQU	0x80 ; EXP1032L1 = 2.302584716116E0
EXP1032L10	EQU	0x13
EXP1032L10 EXP1032L11	EQU	0x5D
EXP1032L11 EXP1032L12	EQU	0x8C
	~ -	
EXP1032L2	EQU	0x80 ; EXP1032L2 = 2.650914554552E0
EXP1032L20	EQU	0x29
EXP1032L21	EQU	0xA8
EXP1032L22	EQU	0x96
EXP1032L3	EQU	0x80 ; EXP1032L3 = 2.033640565225E0
EXP1032L30	EQU	0x02
EXP1032L31	EQU	0x27
EXP1032L32	EQU	0x2B
EXP1032L4	EQU	0x7F ; EXP1032L4 = 1.157459289066E0
EXP1032L40	EQU	0x14
EXP1032L41	EQU	0x27
EXP1032L42	EQU	0xA0

```
EXP1032L5
                EOU
                                 0x7D
                                                 ; EXP1032L5 = 4.544952589676E-1
EXP1032L50
                EOU
                                 0x68
EXP1032L51
                EQU
                                 0xB3
EXP1032L52
                EQU
                                 0x9A
; *********************
        Evaluate log(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                CALL
                        LOG32
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- LOG( AARG )
        Testing on [MINNUM, MAXNUM] from 100000 trials:
                min
                        max
                                 mean
        Timing: 12
                        2147
                                 1985.0 clks
                min
                        max
                                 mean
                                         rms
        Error: -0x01
                        0 \times 02
                                 0.55
                                         0.77
                                                 nsb
        This approximation of the natural log function is based upon the
        expansion
                log(x) = log(f) + log(2**n) = log(f) + n*log(2)
        where .5 <= f < 1 and n is an integer. The additional transformation
                         2*f-1, f < 1/sqrt(2), n = n - 1
                        f-1, otherwise
        produces a naturally segmented representation of log(1+z) on the
        intervals [1/sqrt(2)-1,0] and [0,sqrt(2)-1], utilizing minimax rational
        approximations. The final evaluation of
                log(1+z) + n*log(2) = (log(1+z) - n*c2) + n*c1
        is performed in pseudo extended precision where c1 is close to log(2)
        and has an exact machine representation.
LOG32
                CLRF
                                 AARGB3,W
                BTFSS
                                 AARGB0,MSB
                                                 ; test for negative argument
                                                 ; test for zero argument
                CPFSGT
                                 AEXP
                GOTO
                                 DOMERR32
                MOVFP
                                 FPFLAGS, WREG
                                                 ; save rounding flag
                MOVWF
                                 DARGB3
                                 FPFLAGS, RND
                                                 ; enable rounding
                BSF
                                 AEXP, WREG
                MOVFP
                MOVPF
                                 WREG, EARGB3
                MOVLW
                                 EXPBIAS-1
                SUBWE
                                 EARGB3.F
                                 AEXP
                MOVWF
                MOVLW
                                 0xF3
                                                 ; .70710678118655 = 7E3504F3
                SUBWF
                                 AARGB2,W
```

	MOVLW	0x04	
	SUBWFB	AARGB1,W	
	MOVLW	0x35	
	SUBWFB	AARGB0,W	
	SUBWED	AARGBU,W	
	DEFICA	a	
	BTFSS	_C	
	GOTO	LOG32FLOW	
LOG32FHIGH	MOVLW	0x7F	
	MOVPF	WREG,BEXP	
	CLRF	BARGB0,F	
	CLRF	BARGB1,F	
	CLRF	BARGB2,F	
	CALL	FPS32	
	GOTO	LOGZ32OK	
LOG32FLOW	INCF	AEXP,F	
2000212011	MOVLW	0x7F	
	MOVPF	WREG, BEXP	
	CLRF	BARGB0,F	
	CLRF	BARGB1,F	
	CLRF	BARGB2,F	
	CALL	FPS32	
	DECF	EARGB3,F	
LOGZ320K			
	MOVFP	AEXP, WREG	; save z
	MOVPF	WREG, DEXP	
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
			- ( )
	POLL132	LOG32Q,2,0	; Q(z)
	MOVFP	AEXP, WREG	
	MOVPF	WREG,CEXP	
	MOVPF	AARGB0,CARGB0	
	MOVPF	AARGB1,CARGB1	
	MOVPF	AARGB2,CARGB2	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	DARGB0, AARGB0	
	MOVFP	DARGB1, AARGB1	
	MOVFP	DARGB2, AARGB2	
	DOT 20	T0020D 1 0	. 7( )
	POL32	LOG32P,1,0	; P(z)
	MOVFP	CEXP, WREG	
	MOVPF	WREG,BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1, WREG	
	MOVPF	WREG, BARGB1	
	MOVFP	CARGB2,WREG	
	MOVPF	WREG, BARGB2	
	<del></del>		
	CALL	FPD32	; P(z)/Q(z)
	Ç. 11111	11000	. 1 (2// 2(2/
	MOVFP	AFYD WRFC	; save in CARG
		AEXP, WREG	, bave iii child
	MOVPF	WREG, CEXP	
	MOVPF	AARGB0,CARGB0	

MOVPF	AARGB1,CARGB1	
MOVPF	AARGB2,CARGB2	
MOVFP	DEXP, WREG	
MOVPF	WREG, BEXP	
MOVFP	DARGBO, WREG	
MOVPF	WREG, BARGBO	
MOVFP	DARGB1,WREG	
MOVPF	WREG, BARGB1	
MOVEP	DARGB2, WREG	
MOVPF	WREG,BARGB2	
MOMED	DEVD MDEG	
MOVED	DEXP, WREG	
MOVPF	WREG, AEXP	
MOVFP	DARGB0, AARGB0	
MOVFP	DARGB1, AARGB1	
MOVFP	DARGB2, AARGB2	
CALL	EDW2.0	
CALL	FPM32	; z*z
MOVFP	AEXP, WREG	; save in EARG
MOVPF	WREG, EEXP	
MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1,EARGB1	
MOVPF	AARGB2,EARGB2	
MOVFP	CEXP, WREG	; z*z*P(z)/Q(z)
MOVPF	WREG,BEXP	
MOVFP	CARGB0, WREG	
MOVPF	WREG,BARGB0	
MOVFP	CARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	CARGB2, WREG	
MOVPF	WREG,BARGB2	
CALL	FPM32	
MOVFP	DEXP, WREG	; z*(z*z*P(z)/Q(z))
MOVPF	WREG, BEXP	
MOVFP	DARGB0, WREG	
MOVPF	WREG,BARGB0	
MOVFP	DARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	DARGB2,WREG	
MOVPF	WREG, BARGB2	
CALL	FPM32	
MOVFP	EEXP, WREG	; $5*z*z + z*(z*z*P(z)/Q(z))$
MOVPF	WREG, BEXP	
MOVFP	EARGB0, WREG	
MOVPF	WREG,BARGB0	
MOVFP	EARGB1,WREG	
MOVPF	WREG, BARGB1	
MOVFP	EARGB2, WREG	
MOVPF	WREG, BARGB2	
TSTFSZ	BEXP	
DECF	BEXP, F	
DECF	BEAF, F	
CALL	FPS32	
CULL	LEOJA	
MOVFP	DEXP, WREG	; $z5*z*z + z*(z*z*P(z)/Q(z))$
MOVPF	WREG, BEXP	
MOVFP	DARGBO, WREG	
MOVPF	WREG, BARGBO	
MOVFP	DARGB1,WREG	
PIOVET	PIMODI, WILLG	

	MOVPF	WREG, BARGB1	
	MOVFP	DARGB2,WREG	
	MOVPF	WREG, BARGB2	
	MOVPF	WKEG, BAKGBZ	
	TSTFSZ	EARGB3	
	GOTO	ADJLOG32	
	DEED 6.6	515 <i>6</i> 52 515	
	BTFSS	DARGB3,RND	
	BCF	FPFLAGS, RND	
	CALL	FPA32	
	RETLW	0x00	
ADJLOG32			
AD0 10032	CALL	FPA32	
	MOVFP	AEXP, WREG	; save in EARG
	MOVPF	WREG, EEXP	
	MOVPF	AARGB0, EARGB0	
	MOVPF	AARGB1,EARGB1	
	MOVPF	AARGB2,EARGB2	
	CLRF	AARGB0,F	
	MOVFP	EARGB3,AARGB1	
	BTFSC	AARGB1,MSB	
	SETF	AARGB0,F	
	CALL	FL01624	
	CLRF	AARGB2,F	
	MOVFP	AEXP,WREG	; save k in DARG
	MOVPF	WREG, DEXP	
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	DOE	AADGDO MGD	
	BSF	AARGB0,MSB	
	MOVLW	0x0D-1	; .000212194440055
	SUBWF	AEXP,F	
	MOVLW	0xDE	
	MOVWF	BARGB0	
	MOVLW	0x80	
	MOVWF	BARGB1	
	MOVINE	0x83	
	MOVWF	BARGB2	
	CALL	FXM2424U	
	DEED 6.6	3300000	
	BTFSC	AARGB0,MSB	
	GOTO	LOG32F1OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
		·	
LOG32F1OK			
	BTFSC	DARGB0,MSB	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	MOVFP	EEXD MBEG	; log(1+z) + k*log(2)
		EEXP, WREG	, 10g(112)   K=10g(2)
	MOVPF	WREG, BEXP	
	MOVFP	EARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	EARGB1,WREG	

MOVPF

```
WREG, BARGB1
                                 EARGB2, WREG
                MOVFP
                MOVPF
                                 WREG, BARGB2
                CALL
                                 FPA32
                MOVFP
                                 AEXP, WREG
                                                 ; save in EARG
                MOVPF
                                 WREG, EEXP
                MOVPF
                                 AARGB0, EARGB0
                                 AARGB1,EARGB1
                MOVPF
                MOVPF
                                 AARGB2, EARGB2
                MOVLW
                                 0xB1
                                                 ; .693359375
                MOVWF
                                 BARGB0
                                 0x80
                MOVLW
                MOVWF
                                 BARGB1
                MOVFP
                                 DEXP, WREG
                                 WREG, AEXP
                MOVPF
                                 DARGB0, AARGB0
                MOVFP
                                 DARGB1, AARGB1
                MOVFP
                MOVFP
                                 DARGB2, AARGB2
                BSF
                                 AARGB0,MSB
                CALL
                                 FXM2416U
                                 AARGB0,MSB
                BTFSC
                GOTO
                                 LOG32FOK
                RLCF
                                 AARGB3,F
                RLCF
                                 AARGB2,F
                RLCF
                                 AARGB1,F
                                 AARGB0,F
                RLCF
                DECF
                                 AEXP,F
LOG32FOK
                                 DARGB0, MSB
                BTFSS
                BCF
                                 AARGB0,MSB
                                 RND4032
                CALL
                                                 ; log(1+z) + k*log(2)
                MOVFP
                                 EEXP, WREG
                MOVPF
                                 WREG, BEXP
                MOVFP
                                 EARGB0, WREG
                                 WREG, BARGB0
                MOVPF
                MOVFP
                                 EARGB1, WREG
                MOVPF
                                 WREG, BARGB1
                                 EARGB2, WREG
                MOVFP
                MOVPF
                                 WREG, BARGB2
                BTFSS
                                 DARGB3,RND
                BCF
                                 FPFLAGS, RND
                CALL
                                 FPA32
                RETLW
                                 0x00
;------
        minimax rational approximationz-.5*z*z+z*(z*z*P(z)/Q(z))
LOG32P0
                EQU
                                 0x7E
                                                 ; LOG32P0 = .83311400452
                                 0x55
LOG32P00
                EQU
LOG32P01
                EQU
                                 0x46
LOG32P02
                                 0xF6
                EQU
LOG32P1
                EQU
                                 0x7D
                                                 ; LOG32P1 = .48646956294
LOG32P10
                EQU
                                 0x79
```

```
LOG32P11
               EOU
                               0x12
LOG32P12
               EQU
                               0x8A
LOG32Q0
               EQU
                               0x80
                                               ; LOG32Q0 = .24993759223E1
LOG32Q00
               EQU
                               0x1F
LOG32Q01
               EQU
                               0xF5
LOG32Q02
                               0xC6
               EQU
                               0x80
                                               ; LOG32Q1 = .33339502905E+1
LOG32Q1
               EQU
LOG32Q10
                               0x55
               EQU
LOG32Q11
               EQU
                               0x5F
LOG32Q12
                               0x72
               EQU
LOG32Q2
               EQU
                               0x7F
                                               ; LOG32Q2 = 1.0
LOG32Q20
                               0x00
               EQU
LOG32Q21
               EQU
                               0x00
LOG32Q22
               EQU
                               0x00
Evaluate log10(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
               CALL
                       LOG1032
        Use:
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- LOG10( AARG )
        Testing on [MINNUM, MAXNUM] from 100000 trials:
               min
                       max
                               mean
        Timing: 2001
                       2308
                               2135.9 clks
               min
                       max
                               mean
                                       rms
        Error: -0x01
                       0x02
                               -0.11
                                       0.60
                                               nsb
LOG1032
               MOVFP
                               FPFLAGS, WREG
               MOVWF
                               ZARGB0
                               FPFLAGS, RND
               BCF
                               LOG32
               CALL
               MOVPF
                               AARGB0, DARGB0
                               AARGB0,MSB
               BSF
               MOVLW
                               0xDE
                                               ; log10(e) = .43429448190325
               MOVPF
                               WREG, BARGB0
                               0x5B
               MOVLW
                               WREG, BARGB1
               MOVPF
               MOVLW
                               0xD8
               MOVPF
                               WREG, BARGB2
               MOVLW
                               0xA9
                               WREG, BARGB3
               MOVPF
                                               ; log(x) * log10(e)
                CALL
                               FXM3232U
                               AEXP,F
               DECF
               BTFSC
                               AARGB0, MSB
                GOTO
                               LOG1032OK
                RLCF
                               AARGB4,F
                               AARGB3,F
               RLCF
```

```
RLCF
                             AARGB2.F
              RLCF
                             AARGB1,F
              RLCF
                             AARGB0,F
              DECF
                             AEXP,F
LOG10320K
              BTFSS
                             DARGB0,MSB
                             AARGB0, MSB
              BCF
              BTFSS
                             ZARGB0, RND
                             0x00
              RETLW
              BSF
                             FPFLAGS, RND
              CALL
                             RND4032
              RETLW
                             0x00
Evaluate cos(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Use:
              CALL
                     COS32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- COS( AARG )
       Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
                             mean
                     max
       Timing: 1256
                     2405
                             2182.6 clks
              min
                     max
                             mean
                                    rms
       Error: -0x19A 0x148
                             -1.20
                                    62.83
                                            nsb
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x \mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
       \sin(z) = z * (z**2) * p(z**2), \cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
       where p and q are minimax polynomial approximations.
COS32
              MOVFP
                             FPFLAGS, WREG
                                           ; save rounding flag
              MOVWF
                             DARGB3
                                           ; enable rounding
              BSF
                             FPFLAGS, RND
              CLRF
                             CARGB3,F
                                            ; initialize sign in CARGB3
              BCF
                             AARGB0,MSB
                                            ; use |x|
                             RRSINCOS32
                                            ; range reduction
              CALL
RRCOS320K
                             EARGB3,W
              RRCF
              XORWF
                             EARGB3.W
              BTFSC
                             WREG, LSB
                             COSZSIN32
              GOTO
                             ZCOS32
              CALL
```

```
GOTO
                                  COSSIGN32
COSZSIN32
                 CALL
                                  ZSIN32
                                 EARGB3,LSB+1
COSSIGN32
                BTFSC
                                 CARGB3, MSB
                BTG
                BTFSC
                                  CARGB3, MSB
                                 AARGB0,MSB
                BTG
                 BTFSS
                                  DARGB3, RND
                RETLW
                                  0x00
                BSF
                                 FPFLAGS, RND
                                                  ; restore rounding flag
                CALL
                                 RND4032
                RETLW
                                  0x00
        Evaluate sin(x)
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                CALL
                         SIN32
        Use:
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- SIN( AARG )
        Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
                min
                         max
                                 mean
        Timing: 1338
                                 2182.5 clks
                         2408
                         max
                                 mean
                                          rms
        Error: -0x182 0x18D
                                  -0.91
                                          62.74
                                                   nsb
        The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
        alternative trigonometric argument z on [-pi/4,pi/4], through
        the definition z = x mod pi/4, with an additional variable j
;
        indicating the correct octant, leading to the appropriate call
        to either the sine or cosine approximations
        \sin(z) = z * (z**2) * p(z**2), \cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
        where p and q are minimax polynomial approximations.
SIN32
                MOVFP
                                 FPFLAGS, WREG
                                                   ; save rounding flag
                MOVWF
                                  DARGB3
                BSF
                                 FPFLAGS, RND
                                                  ; enable rounding
                 CLRF
                                  CARGB3,F
                                                   ; initialize sign in CARGB3
                BTFSC
                                  AARGB0, MSB
                                  CARGB3, MSB
                 BSF
                                                   ; use |x|
                BCF
                                 AARGB0,MSB
                                  RRSINCOS32
                 CALL
                                                   ; range reduction
RRSIN32OK
                                  EARGB3,W
                 RRCF
```

```
XORWE
                               EARGR3.W
               BTFSC
                               WREG, LSB
               GOTO
                               SINZCOS32
               CALL
                               ZSIN32
                               SINSIGN32
               GOTO
SINZCOS32
               CALL
                               ZCOS32
SINSIGN32
               BTFSC
                               CARGB3, MSB
                               AARGB0,MSB
               BTG
               BTFSS
                               DARGB3, RND
               RETLW
                               0x00
               BSF
                               FPFLAGS, RND
                                             ; restore rounding flag
               CALL
                               RND4032
                               0x00
               RETLW
Evaluate sin(x) and cos(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
               CALL
                       SINCOS32
       Use:
       Output: 32 bit floating point cos(x) in AEXP, AARGBO, AARGB1, AARGB2 and
                                     sin(x) BEXP, BARGBO, BARGB1, BARGB2
       Result: AARG <-- COS( AARG )
               BARG <-- SIN( AARG )
       Testing on [-LOSSTHR,LOSSTHR] from 100000 trials:
               min
                       max
                               mean
       Timing: 2328
                       3432
                             3217.8 clks
               min
                       max
                               mean
                                       rms
       Error:
               -0x19A
                       0x148
                               -1.20
                                       62.83
                                               nsb
                                                       cos(x)
               -0x182
                       0x18D
                               -0.91
                                       62.74
                                                       sin(x)
       The actual argument x on [-LOSSTHR,LOSSTHR] is mapped to the
       alternative trigonometric argument z on [-pi/4,pi/4], through
       the definition z = x mod pi/4, with an additional variable j
       indicating the correct octant, leading to the appropriate call
       to either the sine or cosine approximations
       \sin(z) = z * (z**2) * p(z**2), \cos(z) = 1 - .5 * z**2 + (z**4) * q(z**2)
       where p and q are minimax polynomial approximations. In this case,
       only one range reduction is necessary.
SINCOS32
                               FPFLAGS, WREG
               MOVFP
                                              ; save rounding flag
               MOVWF
                               DARGB3
               BSF
                               FPFLAGS, RND
                                               ; enable rounding
               MOVFP
                               AEXP, WREG
                                               ; save x in EARG
               MOVWF
                               EEXP
                               AARGB0, EARGB0
               MOVPF
               MOVPF
                               AARGB1, EARGB1
               MOVPF
                               AARGB2, EARGB2
```

```
BCF
                  AARGB0, MSB
                                    ; use |x|
                                    ; initialize sign in CARGB3
CLRF
                  CARGB3,F
                  RRSINCOS32
                                    ; range reduction
CALL
MOVFP
                  CARGB3, WREG
                                    ; save sign from range reduction
MOVWF
                  ZARGB2
BTFSC
                  EARGB0,MSB
                                    ; toggle sign if x < 0
                  CARGB3, MSB
BTG
CALL
                  RRSIN320K
MOVFP
                  AEXP, WREG
                                    ; save sin(x) in EARG
MOVWF
                  EEXP
MOVPF
                  AARGB0, EARGB0
                 AARGB1, EARGB1
MOVPF
MOVPF
                  AARGB2, EARGB2
MOVPF
                  AARGB3, ZARGB3
BSF
                  FPFLAGS, RND
                                    ; enable rounding
MOVFP
                  DEXP, WREG
                                    ; restore z*z in AARG
MOVWF
                  AEXP
MOVFP
                  DARGB0, AARGB0
MOVFP
                  DARGB1, AARGB1
MOVFP
                  DARGB2, AARGB2
MOVFP
                  ZARGB2, WREG
                                    ; restore sign from range reduction
MOVWF
                  CARGB3
                 RRCOS320K
CALL
MOVFP
                  EEXP, WREG
                                    ; restore sin(x) in BARG
MOVPF
                  WREG, BEXP
MOVFP
                  EARGB0, WREG
                  WREG, BARGB0
MOVPF
MOVFP
                  EARGB1, WREG
MOVPF
                  WREG, BARGB1
MOVFP
                  EARGB2, WREG
                  WREG, BARGB2
MOVPF
                  ZARGB3, WREG
MOVFP
                  BARGB3
MOVWF
RETLW
                  0x00
```

```
sign of the result leads to appropriate use of the sine or cosine
        routine in each case.
        The calculation of z is then obtained through a pseudo extended
        precision method
        z = x \mod pi/4 = x - y*(pi/4) = (((x - p1*y)-p2*y)-p3*y)-p4*y
        where pi/4 = p1 + p2 + p3 + p4, with p1 close to pi/4, p2 close to
        pi/4 - p1, and p3 close to pi/4 - p1 - p2. The numbers p1, p2 and p3
        are chosen to have an exact machine representation with slightly more
        than the lower half of the mantissa bits zero, typically leading to no
         error in computing the terms in parenthesis. This calculation breaks
        down leading to a loss of precision for |x| > LOSSTHR = sqrt(2**24)*pi/4,
        or for |x| close to an integer multiple of pi/4. This loss threshold has
        been chosen based on the efficacy of this calculation, with a domain error
        reported if this threshold is exceeded.
RRSTNCOS32
                 MOVED
                                   AEXP, WREG
                                                     ; loss threshold check
                 SUBLW
                                   LOSSTHR32EXP
                 BTFSS
                                   _C
                 GOTO
                                   DOMERR32
                 BTFSS
                                   _{\rm Z}
                                   RRSINCOS32ARGOK
                 GOTO
                 MOVFP
                                   AARGB0, WREG
                 SUBLW
                                   LOSSTHR32B0
                 BTFSS
                                   _C
                 GOTO
                                   DOMERR32
                 BTFSS
                                   _{\rm Z}
                 GOTO
                                   RRSINCOS32ARGOK
                                   AARGB1, WREG
                 MOVED
                 SUBLW
                                   LOSSTHR32B1
                 BTFSS
                                   _C
                 GOTO
                                   DOMERR32
                 BTFSS
                                   _{\rm Z}
                 GOTO
                                   RRSINCOS32ARGOK
                 MOVFP
                                   AARGB2, WREG
                 SUBLW
                                   LOSSTHR32B2
                 BTFSS
                                   _C
                                   DOMERR32
                 GOTO
RRSINCOS32ARGOK
                 MOVFP
                                   AEXP, WREG
                                   WREG, CEXP
                                                     ; save |x| in CARG
                 MOVPF
                 MOVPF
                                   AARGB0, CARGB0
                 MOVPF
                                   AARGB1, CARGB1
                 MOVPF
                                   AARGB2, CARGB2
         fixed point multiplication by 4/pi
                 BSF
                                   AARGB0,MSB
                 MOVPF
                                   AARGB0, BARGB0
                                   AARGB1, BARGB1
                 MOVPF
                 MOVPF
                                   AARGB2, BARGB2
                 MOVLW
                                                     ; 4/pi = 1.27323954474
                                   0xA2
                 MOVPF
                                   WREG, AARGB0
                 MOVLW
                                   0xF9
                                   WREG, AARGB1
                 MOVPF
                 MOVLW
                                   0x83
                 MOVPF
                                   WREG, AARGB2
                 MOVLW
                                   0x6E
```

```
MOVPF
                                   WREG, AARGB3
                 CALL
                                   FXM3224U
                 INCF
                                   AEXP,F
                 BTFSC
                                   AARGB0,MSB
                 GOTO
                                   RRSINCOS32YOK
                 RLCF
                                   AARGB3,F
                                   AARGB2,F
                 RLCF
                 RLCF
                                   AARGB1,F
                 RLCF
                                   AARGB0,F
                 DECF
                                   AEXP,F
RRSINCOS32YOK
                 BCF
                                   AARGB0,MSB
                 BCF
                                   FPFLAGS, RND
                                                     ; y = [|x| * (4/pi)]
                                   INT3224
                 CALL
                                   FPFLAGS, RND
                 BSF
                 BTFSS
                                   AARGB2,LSB
                 GOTO
                                   SAVEY32
                                   AARGB2,F
                 INCF
                                   WREG, F
                 CLRF
                 ADDWFC
                                   AARGB1,F
                 ADDWFC
                                   AARGB0,F
SAVEY32
                 MOVPF
                                   AARGB0,DARGB0
                                                     ; save y in DARG
                                   AARGB1, DARGB1
                 MOVPF
                 MOVPF
                                   AARGB2, DARGB2
                                   0x07
                                                     ; j = y \mod 8
                 MOVLW
                                   AARGB2,F
                 ANDWF
                 MOVLW
                                   0x03
                 CPFSGT
                                   AARGB2
                 GOTO
                                   JOK32
                 BTG
                                   CARGB3, MSB
                 MOVLW
                                   0x04
                 SUBWF
                                   AARGB2,F
JOK32
                                   AARGB2, EARGB3
                                                     ; save j in EARGB3
                 MOVPF
                 MOVFP
                                   DARGB0, AARGB0
                                                     ; restore y to AARG
                 MOVFP
                                   DARGB1, AARGB1
                 MOVFP
                                   DARGB2, AARGB2
                 CALL
                                   FLO2432
                 MOVFP
                                   AEXP, WREG
                                   WREG, DEXP
                                                     ; save y in DARG
                 MOVPF
                 BTFSC
                                   _Z
                                   RRSINCOS32ZEQX
                 GOTO
                 MOVPF
                                   AARGB0,DARGB0
                 MOVPF
                                   AARGB1,DARGB1
                 MOVPF
                                   AARGB2, DARGB2
        Cody-Waite extended precision calculation of |x| - y * pi/4 using
;
        fixed point multiplication. Since y >= 1, underflow is not possible
        in any of the products.
                 BSF
                                   AARGB0, MSB
                 MOVLW
                                   0xC9
                                                     i - p1 = -.78515625
                 MOVPF
                                   WREG, BARGB0
```

	CLRF	BARGB1,F	
	CALL	FXM2416U	
	BTFSC	AARGB0,MSB	
		RRSINCOS32Z1OK	
	GOTO		
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32Z1OK			_
	MOVFP	CEXP, WREG	; restore x to BARG
	MOVPF	WREG,BEXP	
	MOVFP	CARGB0,WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG,BARGB1	
	MOVFP	CARGB2, WREG	
	MOVPF	WREG,BARGB2	
	CALL	FPA32	; $z1 =  x  - y * (p1)$
	MOVFP	AEXP, WREG	
	MOVPF	WREG, CEXP	; save z1 in CARG
	MOVPF	AARGB0,CARGB0	, pave 21 in cinc
	MOVPF	AARGB1,CARGB1	
	MOVPF	AARGB2,CARGB2	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	DARGB0, AARGB0	; restore y to AARG
	MOVFP	DARGB1, AARGB1	
	MOVFP	DARGB2, AARGB2	
	BSF	AARGB0,MSB	
	MOVLW	0xFD	; - p2 =00024187564849853515624
	MOVPF	WREG, BARGB0	
	MOVLW	0xA0	
	MOVPF	WREG,BARGB1	
	CALL	FXM2416U	
	MOVLW	0x0D - 1	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS32Z2OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32Z2OK			
RRBINCOBSZZZOR	SUBWF	AEXP,F	
	MOVFP	CEXP, WREG	; restore z1 to BARG
	MOVPF	WREG, BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1, WREG	
	MOVPF	WREG,BARGB1	
	MOVPF MOVFP	WREG,BARGB1 CARGB2,WREG	

	CALL	FPA32	; z2 = z1 - y * (p2)
	MOVFP	AEXP, WREG	
	MOVPF	WREG, CEXP	; save z2 in CARG
	MOVPF	AARGB0, CARGB0	
	MOVPF	AARGB1,CARGB1	
	MOVPF	AARGB2,CARGB2	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	DARGB0, AARGB0	; restore y to AARG
	MOVFP	DARGB1,AARGB1	
	MOVFP	DARGB2,AARGB2	
	BSF	AARGB0,MSB	
	MOM IN	0xA2	; - p3 = -3.7747668102383613583E-8
	MOVLW MOVPF	WREG,BARGB0	7 - p33.7747000102303013303E-0
	MOVLW	0x20	
	MOVPF	WREG,BARGB1	
	a	TYP10 41 677	
	CALL	FXM2416U	
	MOVLW	0x19 - 1	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS32Z3OK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32Z3OK			
RRSINCOSSZZSOR	SUBWF	AEXP,F	
	DODWI	ABZI , I	
	MOVFP	CEXP, WREG	; restore z2 to BARG
	MOVPF	WREG, BEXP	
	MOVFP	CARGB0, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG,BARGB1	
	MOVFP	CARGB2,WREG	
	MOVPF	WREG,BARGB2	
	CALL	FPA32	; z3 = z2 - y * (p3)
	MOVFP	AEXP, WREG	
	MOVPF	WREG, CEXP	; save z3 in CARG
	MOVPF	AARGB0, CARGB0	
	MOVPF	AARGB1,CARGB1	
	MOVPF	AARGB2,CARGB2	
	MOVFP	DEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	DARGB0, WREG	
	MOVWF	BARGB0	; restore y to BARG
	MOVFP	DARGB1,WREG	
	MOVWF	BARGB1	
	MOVFP	DARGB2,WREG	
	MOVWF	BARGB2	
	BSF	BARGB0,MSB	
	MOVLW	0xB4	; - p4 = -3.77489497744597636E-8

	MOVPF	WREG,AARGB0	
	MOVLW	0x61	
	MOVPF	WREG, AARGB1	
	MOVLW	0x1A	
	MOVPF	WREG, AARGB2	
	MOVLW	0x63	
	MOVPF	WREG,AARGB3	
	CALL	FXM3224U	
	MOVLW	0x28 - 1	
	DTECC	AADCDO MCD	
	BTFSC	AARGBO,MSB	
	GOTO	RRSINCOS32Z4OK	
	RLCF RLCF	AARGB4,F AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32Z4OK			
	SUBWF	AEXP,F	
	CALL	RND4032	
	MOVFP	CEXP, WREG	; restore z3 to BARG
	MOVPF	WREG, BEXP	
	MOVFP	CARGBO, WREG	
	MOVPF	WREG,BARGB0	
	MOVFP	CARGB1,WREG	
	MOVPF	WREG, BARGB1	
	MOVFP	CARGB2,WREG	
	MOVPF	WREG,BARGB2	
	BCF	FPFLAGS, RND	; disable rounding
	CALL	FPA32	• + / - / \
			; z = z3 - y * (p4)
RRSINCOS320K			, z = zs - y " (p4)
RRSINCOS320K			, z = 23 - y " (p4)
RRSINCOS320K	MOVFP	AEXP,WREG	
RRSINCOS320K	MOVFP MOVPF	AEXP, WREG WREG, CEXP	; save z in CARG
RRSINCOS320K	MOVFP MOVPF	AEXP, WREG WREG, CEXP AARGB0, CARGB0	
RRSINCOS320K	MOVFP MOVPF MOVPF	AEXP, WREG WREG, CEXP AARGBO, CARGBO AARGB1, CARGB1	
RRSINCOS320K	MOVFP MOVPF	AEXP, WREG WREG, CEXP AARGB0, CARGB0	
RRSINCOS32OK	MOVFP MOVPF MOVPF	AEXP, WREG WREG, CEXP AARGBO, CARGBO AARGB1, CARGB1	
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2	; save z in CARG
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO	AEXP, WREG WREG, CEXP AARGBO, CARGBO AARGB1, CARGB1 AARGB2, CARGB2 AARGB3, MSB RRSINCOS32ZOK	<pre>; save z in CARG ; is NSB &lt; 0x80?</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF	AEXP, WREG WREG, CEXP AARGBO, CARGBO AARGB1, CARGB1 AARGB2, CARGB2 AARGB3, MSB RRSINCOS32ZOK	; save z in CARG
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2 AARGB3, MSB RRSINCOS32ZOK	<pre>; save z in CARG ; is NSB &lt; 0x80?</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3	<pre>; save z in CARG ; is NSB &lt; 0x80? ; set carry for rounding</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2 AARGB3, MSB RRSINCOS32ZOK	<pre>; save z in CARG ; is NSB &lt; 0x80?</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB3 AARGB2, W	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3	<pre>; save z in CARG ; is NSB &lt; 0x80? ; set carry for rounding</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB3 AARGB2, W  AARGB0, SIGN	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80 ; save sign</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB3 AARGB2, W  AARGB0, SIGN	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80 ; save sign</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF MOVPF BSF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit</pre>
RRSINCOS320K	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF MOVPF BSF CLRF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF MOVPF BSF CLRF ADDWFC	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit</pre>
RRSINCOS 3 2 OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO BSF MOVLW CPFSGT RRCF MOVPF BSF CLRF ADDWFC ADDWFC ADDWFC	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit  ; round</pre>
RRSINCOS 3 2 OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO  BSF MOVLW CPFSGT RRCF  MOVPF BSF  CLRF ADDWFC ADDWFC ADDWFC BTFSS	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit</pre>
RRSINCOS 3 2 OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO  BSF MOVLW CPFSGT RRCF  MOVPF BSF  CLRF ADDWFC ADDWFC ADDWFC BTFSS GOTO	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F _C RRSINCOS32RZOK	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit  ; round  ; has rounding caused carryout?</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO  BSF MOVLW CPFSGT RRCF  MOVPF BSF  CLRF ADDWFC ADDWFC ADDWFC BTFSS GOTO RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F  _C RRSINCOS32RZOK CARGB0, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit  ; round</pre>
RRSINCOS32OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO  BSF MOVLW CPFSGT RRCF  MOVPF BSF  CLRF ADDWFC ADDWFC ADDWFC BTFSS GOTO RRCF RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F  _C RRSINCOS32RZOK CARGB0, F CARGB1, F CARGB1, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit  ; round  ; has rounding caused carryout?</pre>
RRSINCOS 3 2 OK	MOVFP MOVPF MOVPF MOVPF MOVPF BTFSS GOTO  BSF MOVLW CPFSGT RRCF  MOVPF BSF  CLRF ADDWFC ADDWFC ADDWFC BTFSS GOTO RRCF	AEXP, WREG WREG, CEXP AARGB0, CARGB0 AARGB1, CARGB1 AARGB2, CARGB2  AARGB3, MSB RRSINCOS32ZOK  _C 0x80 AARGB3 AARGB2, W  AARGB0, SIGN CARGB0, MSB  WREG, F CARGB2, F CARGB1, F CARGB0, F  _C RRSINCOS32RZOK CARGB0, F	<pre>; save z in CARG  ; is NSB &lt; 0x80?  ; set carry for rounding  ; select even if NSB = 0x80  ; save sign ; make MSB explicit  ; round  ; has rounding caused carryout?</pre>

	INFSNZ	CEXP, F	; test for floating point overflow
	GOTO	SETFOV32	
RRSINCOS32RZOK	DEED G.G.	G = G12 1/GB	
	BTFSS	SIGN, MSB	· alass sion bit if manihing
	BCF	CARGB0,MSB	; clear sign bit if positive
RRSINCOS32ZOK			
10101110000000001	BSF	AARGB0,MSB	; make MSB explicit
	MOVPF	AARGB0,BARGB0	-
	MOVPF	AARGB1,BARGB1	
	MOVPF	AARGB2,BARGB2	
	MOVPF	AARGB3,BARGB3	
	CALL	FXM3232U	; z * z
	BCF	_C	; multiply exponent by 2
	RLCF	AEXP,F	, marciply exponent by 2
	MOVLW	EXPBIAS-1	
	SUBWFB	AEXP,F	
	INCF	AEXP,F	
	BTFSC	AARGB0,MSB	
	GOTO	RRSINCOS32ZZOK	
	RLCF	AARGB3,F	
	RLCF	AARGB2,F	
	RLCF	AARGB1,F	
	RLCF	AARGB0,F	
	DECF	AEXP,F	
RRSINCOS32ZZOK	5.00	110000 1100	
	BCF	AARGB0,MSB	
	CALL	RND4032	
	BSF	FPFLAGS, RND	; enable rounding
	MOMED	AEXP,WREG	
	MOVFP MOVPF	WREG, DEXP	; save z * z in DARG
	MOVPF	AARGB0,DARGB0	, bave 2 2 III bline
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	RETLW	0x00	
RRSINCOS32ZEQX			
~	MOVFP	CEXP, WREG	
	MOVPF	WREG, AEXP	
	MOVFP	CARGB0, AARGB0	
	MOVFP	CARGB1,AARGB1	
	MOVFP	CARGB2, AARGB2	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, BEXP	
	MOVPF	AARGB0,BARGB0	
	MOVPF	AARGB1,BARGB1	
	MOVPF	AARGB2,BARGB2	
	CALL	FPM32	; z * z
	<u></u>	11102	
	MOVFP	AEXP, WREG	
	MOVPF	WREG, DEXP	; save z * z in DARG
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	RETLW	0x00	

;**************************************		
ZCOS32	POL32	COS32D,2,1
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2, WREG
	MOVPF	WREG, BARGB2
	CALL	FPM32
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2, WREG
	MOVPF	WREG, BARGB2
	CALL	FPM32
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2, WREG
	MOVPF	WREG, BARGB2
	DECF	BEXP, F
	CALL	FPS32
	MOVLW	EXPBIAS
	MOVWF	BEXP
	CLRF	BARGBO,F
	CLRF	BARGB1,F
	CLRF	BARGB2,F
	BCF	FPFLAGS,RND
	CALL	FPA32
	RETLW	0x00
ZSIN32		
	POL32	SIN32D,3,1
	MOVFP	DEXP, WREG
	MOVPF	WREG, BEXP
	MOVFP	DARGBO, WREG
	MOVPF	WREG, BARGBO
	MOVFP	DARGB1,WREG
	MOVPF	WREG, BARGB1
	MOVFP	DARGB2, WREG
	MOVPF	WREG, BARGB2
	CALL	FPM32
	MOVFP	CEXP, WREG
	MOVPF	WREG, BEXP

```
CARGBO, WREG
                 MOVFP
                 MOVPF
                                   WREG, BARGB0
                 MOVFP
                                   CARGB1, WREG
                 MOVPF
                                   WREG, BARGB1
                 MOVFP
                                   CARGB2, WREG
                 MOVPF
                                   WREG, BARGB2
                 CALL
                                   FPM32
                 MOVED
                                   CEXP, WREG
                 MOVPF
                                   WREG, BEXP
                 MOVFP
                                   CARGBO, WREG
                 MOVPF
                                   WREG, BARGB0
                 MOVFP
                                   CARGB1, WREG
                                   WREG, BARGB1
                 MOVPF
                 MOVFP
                                   CARGB2, WREG
                                   WREG, BARGB2
                 MOVPF
                 BCF
                                   FPFLAGS, RND
                 CALL
                                   FPA32
                 RETLW
                                   0x00
        minimax polynomial coefficients for \sin(z) = z + z * (z**2) * p(z**2) on [-pi/4, pi/4]
SIN32D0
                 EOU
                                   0x7C
                                                     ; SIN32D0 = -1.666666664079712E-1
SIN32D00
                 EQU
                                   0xAA
SIN32D01
                 EQU
                                   0xAA
SIN32D02
                 EQU
                                   0xAB
SIN32D1
                 EQU
                                   0x78
                                                     ; SIN32D1 = 8.333329304850749E-3
SIN32D10
                 EOU
                                   0x08
SIN32D11
                                   0x88
                 EQU
SIN32D12
                 EQU
                                   0x84
SIN32D2
                 EQU
                                   0x72
                                                     ; SIN32D2 = -1.983931227180460E-4
SIN32D20
                                   0xD0
                 EQU
SIN32D21
                 EQU
                                   0x07
SIN32D22
                 EQU
                                   0xC0
SIN32D3
                 EQU
                                   0x6C
                                                     ; SIN32D3 = 2.718121647219611E-6
SIN32D30
                 EQU
                                   0x36
SIN32D31
                 EQU
                                   0x68
SIN32D32
                 EQU
                                   0xF9
        minimax polynomial coefficients for cos(z) = 1 - .5*z**2 + z**4*q(z**2)
        on [-pi/4,pi/4]
                                   0x7A
                                                     ; COS32D0 = 4.166664568297614E-2
COS32D0
                 EQU
COS32D00
                 EQU
                                   0x2A
COS32D01
                 EQU
                                   0xAA
COS32D02
                 EQU
                                   0xA5
COS32D1
                 EQU
                                   0x75
                                                     ; COS32D1 = -1.388731625438419E-3
COS32D10
                 EQU
                                   0xB6
COS32D11
                 EQU
                                   0x06
COS32D12
                 EQU
                                   0x1A
COS32D2
                 EQU
                                   0x6F
                                                     ; COS32D2 = 2.443315706066392E-5
COS32D20
                 EQU
                                   0x4C
COS32D21
                 EQU
                                   0xF5
```

```
COS32D22
               EOU
                               OxCE
Evaluate sqrt(x)
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                       SQRT32
               CALL
       Use:
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
       Result: AARG <-- SQRT( AARG )
       Testing on [0,MAXNUM] from 100000 trials:
               min
                       max
                               mean
       Timing: 10
                       568
                               494.0
                                       clks
               min
                       max
                               mean
                                       rms
        Error:
               -0x41
                       0x41
                               0.04
                                       36.87
       Range reduction for the square root function is naturally produced by
       the floating point representation,
               x = f * 2**e, where 1 <= f < 2,
       leading to the expression
                          | sqrt(f) * 2**(e/2),e even
                          sqrt(f) * sqrt(2) * 2**(e/2),e odd
       The approximation of sqrt(f) utilizes a table lookup of 16 bit
       estimates of the square root with linear interpolation between
       adjacent entries as a seed to a single Newton-Raphson iteration,
               y = (y0 + f/y0)/2,
       where the precision of the result is guaranteed by the precision of the
       seed and the quadratic conversion of the method.
SQRT32
               BTFSC
                               AARGB0,MSB
                                              ; test for negative argument
                               DOMERR32
               GOTO
               CLRF
                               AARGB3,W
                                               ; return if argument zero
               CPFSGT
                               AEXP
                               0x00
               RETLW
               MOVFP
                               AEXP, WREG
               MOVPF
                               WREG, CEXP
                                               ; save x in CARG
               MOVPF
                               AARGB0, CARGB0
                               AARGB1, CARGB1
               MOVPF
               MOVPF
                               AARGB2, CARGB2
               MOVFP
                               FPFLAGS, WREG
                                               ; save RND flag in DARGB3
               MOVPF
                               WREG, DARGB3
                               FPFLAGS, RND
               BSF
                                               ; enable rounding
               MOVLW
                               EXPBIAS
                                               ; initialize exponent
               MOVPF
                               WREG, AEXP
```

```
to the square root of AARG, with the top 8 explicit bits of AARG as a pointer
and the remaining 15 explicit bits as the argument to linear interpolation.
                          HIGH (RATBL256I); access table for y0
        MOVLW
        MOVWF
                          TBLPTRH
        RLCF
                          AARGB1,W
        RLCF
                          AARGB0,W
        ADDLW
                          LOW (RATBL256I)
        MOVWF
                          TBLPTRL
        BTFSC
                          _C
        INCF
                          TBLPTRH, F
                          0,1,TEMPB0
        TABLED
        TLRD
                          1,TEMPB0
        TABLRD
                          0,0,TEMPB1
        TLRD
                          0,AARGB5
        MOVFP
                          TEMPB1, WREG
                                            ; calculate difference
        SUBWF
                          AARGB5,W
        MOVWF
                          AARGB5
        BCF
                          _C
                                            ; interpolate
        RLCF
                          AARGB2,W
                          AARGB5
        MULWF
        MOVPF
                          PRODH, TBLPTRH
        RLCF
                          AARGB1,W
        MULWF
                          AARGB5
        MOVPF
                          PRODL, WREG
        ADDWF
                          TBLPTRH, F
        BTFSC
                          _C
                          PRODH, F
        INCF
        CLRF
                          TEMPB2,F
        MOVFP
                          TBLPTRH, WREG
        ADDWF
                          TEMPB2,F
        MOVPF
                          PRODH, WREG
        ADDWFC
                          TEMPB1,F
        CLRF
                          WREG, F
        ADDWFC
                          TEMPB0,F
                                            ; y0
        MOVFP
                          TEMPB0, AARGB0
        MOVFP
                          TEMPB1, AARGB1
        MOVFP
                          TEMPB2, AARGB2
        BTFSC
                          CEXP, LSB
                                            ; is CEXP even or odd?
        GOTO
                          RRSQRT320K
fixed point multiplication by sqrt(2)
                          AARGB0,MSB
        BSF
                                            ; make MSB explicit
                          0xB5
                                            ; sqrt(2) = 1.41421356237
        MOVLW
        MOVPF
                          WREG, BARGB0
        MOVLW
                          0x04
                          WREG, BARGB1
        MOVPF
        MOVLW
                          0xF3
        MOVPF
                          WREG, BARGB2
                          FXM2424U
        CALL
        INCF
                          AEXP,F
        BTFSC
                          AARGB0,MSB
                          RRSQRT320K
        GOTO
```

generation of y0 by interpolating between consecutive 16 bit approximations

```
RLCF
                                    AARGB3,F
                  RLCF
                                    AARGB2,F
                  RLCF
                                    AARGB1,F
                  RLCF
                                    AARGB0,F
                  DECF
                                    AEXP,F
RRSQRT320K
                  BCF
                                    AARGB0,MSB
                                                      ; make MSB implicit
                                    RND4032
                  CALL
                                    EXPBIAS
                                                      ; divide exponent by two
                  MOVLW
                  ADDWF
                                    CEXP,W
                  RRCF
                                    WREG, F
                  MOVPF
                                    WREG, AEXP
                  MOVPF
                                    WREG, BEXP
                  MOVPF
                                    WREG, DEXP
                                    AARGB0,DARGB0
                  MOVPF
                  MOVPF
                                    AARGB1, DARGB1
                  MOVPF
                                    AARGB2, DARGB2
                                    AARGB0, BARGB0
                  MOVPF
                  MOVPF
                                    AARGB1,BARGB1
                  MOVPF
                                    AARGB2,BARGB2
                                    CEXP, WREG
                  MOVFP
                  MOVPF
                                    WREG, AEXP
                                    CARGBO, AARGBO
                  MOVFP
                  MOVFP
                                    CARGB1, AARGB1
                  MOVFP
                                    CARGB2, AARGB2
                                    FPD32
                  CALL
                                                      ; Newton-Raphson iteration
                  MOVFP
                                    DEXP, WREG
                  MOVPF
                                    WREG, BEXP
                  MOVFP
                                    DARGB0, WREG
                                    WREG, BARGB0
                  MOVPF
                                    DARGB1, WREG
                  MOVFP
                  MOVPF
                                    WREG, BARGB1
                  MOVFP
                                    DARGB2, WREG
                                    WREG,BARGB2
                  MOVPF
                  BTFSS
                                    DARGB3, RND
                  BCF
                                    FPFLAGS, RND
                                    FPA32
                  CALL
                  DECF
                                    AEXP,F
                                    0x00
                  RETLW
         Rounded to the nearest approximations to sqrt(f), with pointer from
         the 8 most significant explicit bits of f, the mantissa of x. Linear
         interpolation is performed between adjacent entries using the remaining
         explicit bits of f.
RATBL256I
                  DATA
                           0x0000
                  DATA
                           0x0040
                  DATA
                           0x0080
                  DATA
                           0x00BF
                  DATA
                           0x00FF
```

DATA	0x013E
DATA	0x017E
DATA	0x01BD
DATA	0x01FC
DATA	0x023B
DATA	0x027A
DATA	0x02B9
DATA	$0 \times 02 F7$
DATA	$0 \times 0336$
DATA	$0 \times 0374$
DATA	0x03B2
DATA	0x03F0
DATA	0x042F
DATA	0x046C
DATA	0x04AA
DATA	0x04E8
DATA	$0 \times 0526$
DATA	$0 \times 0563$
DATA	$0 \times 05 A0$
DATA	0x05DE
DATA	0x061B
	0x0658
DATA	
DATA	0x0695
DATA	0x06D2
DATA	0x070E
DATA	$0 \times 074 B$
DATA	0x0787
DATA	0x07C4
DATA	0x0800
DATA	0x083C
DATA	$0 \times 0.878$
DATA	0x08B4
DATA	0x08F0
DATA	0x092C
DATA	0x0968
DATA	0x09A3
DATA	0x09DF
DATA	0x0A1A
DATA	0x0A1A
DATA	0x0A90
DATA	0x0ACB
DATA	0x0B06
DATA	0x0B41
DATA	0x0B7C
DATA	0x0BB7
DATA	0x0BF1
DATA	0x0C2C
DATA	0x0C26
DATA	0x0CA1
DATA	0x0CDB
DATA	$0 \times 0 D15$
DATA	$0 \times 0 D4F$
DATA	0x0D89
DATA	0x0DC3
DATA	0x0DFC
DATA	0x0E36
DATA	0x0E30
DATA	0x0EA9
DATA	0x0EE2
DATA	0x0F1C
DATA	0x0F55
DATA	0x0F8E
DATA	0x0FC7
DATA	0x1000
DATA	0x1039
DATA	0x1033
PUIN	UAIU/4

## **AN660**

DATA	0x10AA
DATA	0x10E3
DATA	0x111B
DATA	0x1154
DATA	0x118C
DATA	0x11C4 0x11FC
DATA DATA	0x11FC
DATA	0x1255
DATA	0x12A4
DATA	0x12DC
DATA	0x1314
DATA	0x134C
DATA	0x1383
DATA	0x13BB
DATA	0x13F2
DATA	0x1429
DATA	0x1461 0x1498
DATA DATA	0x1496 0x14CF
DATA	0x14CF
DATA	0x153D
DATA	0x1574
DATA	0x15AB
DATA	0x15E1
DATA	0x1618
DATA	0x164E
DATA	0x1685
DATA	0x16BB
DATA	0x16F2
DATA DATA	0x1728 0x175E
DATA	0x175E
DATA	0x1754 0x17CA
DATA	0x1800
DATA	0x1836
DATA	0x186C
DATA	0x18A1
DATA	0x18D7
DATA	0x190D
DATA	0x1942
DATA	0x1977 0x19AD
DATA DATA	0x19AD 0x19E2
DATA	0x13E2 0x1A17
DATA	0x1A17
DATA	0x1A81
DATA	0x1AB6
DATA	0x1AEB
DATA	0x1B20
DATA	0x1B55
DATA	0x1B8A
DATA	0x1BBE
DATA	0x1BF3
DATA DATA	0x1C27 0x1C5C
DATA	0x1C3C
DATA	0x1CC4
DATA	0x1CF9
DATA	0x1D2D
DATA	0x1D61
DATA	0x1D95
DATA	0x1DC9
DATA	0x1DFD
DATA	0x1E31
DATA	0x1E64

DATA	0x1E98
DATA	0x1ECC
DATA	0x1EFF
DATA	0x1F33
DATA	0x1F66
DATA	0x1F99
DATA	0x1FCD
DATA	0x2000
DATA	0x2033
DATA	0x2066
DATA	0x2099
DATA	0x20CC
DATA	0x20FF
DATA	0x2132
DATA	0x2165
DATA	0x2198
DATA	0x21CA
DATA	0x21FD
DATA	0x222F
DATA	0x2262
DATA	0x2294
DATA	0x22C7
DATA	0x22F9
DATA	0x232B
DATA	0x235D
DATA	0x238F
DATA	0x23C2
DATA	0x23F4
DATA	0x2425
DATA	0x2457
DATA	0x2489
DATA	0x24BB
DATA	0x24ED
DATA	0x251E
DATA	0x2550
DATA	0x2581
DATA	0x25B3
DATA	0x25E4
DATA	0x2616
DATA	0x2647
DATA	0x2678
DATA	0x26A9
DATA	0x26DA
DATA	0x270B
DATA	0x273D
DATA	0x276D
DATA	0x279E
DATA	0x27CF
DATA	0x2800
DATA	0x2831
DATA	0x2861
DATA	0x2892
DATA	0x28C3
DATA	0x28F3
DATA	0x2924
DATA	0x2954
DATA	0x2984
DATA	0x29B5
DATA	0x29E5
DATA	0x2A15
DATA	0x2A45
DATA	0x2A75
DATA	0x2AA5
DATA	0x2AD5
DATA	0x2B05
DATA	0x2B35

0x2B65

ATAG

```
DATA
                     0x2B95
              DATA
                     0x2BC4
              DATA
                     0x2BF4
              DATA
                     0x2C24
                     0x2C53
              DATA
              DATA
                     0x2C83
              DATA
                     0x2CB2
              DATA
                     0x2CE2
              DATA
                     0x2D11
              DATA
                     0x2D40
                     0x2D70
              DATA
              DATA
                     0x2D9F
              DATA
                     0x2DCE
                     0x2DFD
              DATA
              DATA
                     0x2E2C
              DATA
                     0x2E5B
              DATA
                     0x2E8A
                     0x2EB9
              DATA
              DATA
                     0x2EE8
              DATA
                     0x2F17
              DATA
                     0x2F45
              DATA
                     0x2F74
              DATA
                     0x2FA3
                     0x2FD1
              DATA
              DATA
                     0x3000
              DATA
                     0x302F
              DATA
                     0x305D
              DATA
                     0x308B
              DATA
                     0x30BA
              DATA
                     0x30E8
              DATA
                     0x3116
              DATA
                     0x3145
              DATA
                     0x3173
              DATA
                     0x31A1
              DATA
                     0x31CF
              DATA
                     0x31FD
              DATA
                     0x322B
              DATA
                     0x3259
              DATA
                     0x3287
              DATA
                     0x32B5
              DATA
                     0x32E3
              DATA
                     0x3310
              DATA
                     0x333E
                     0x336C
              DATA
              DATA
                     0x3399
              DATA
                     0x33C7
              DATA
                     0x33F5
              DATA
                     0x3422
              DATA
                     0x3450
              DATA
                     0x347D
              DATA
                     0x34AA
              DATA
                     0x34D8
                     0x3505
              DATA
Evaluate pow(x,y) = X**Y
              24 bit floating point number X in AEXP, AARGBO, AARGB1 and
       Input:
              24 bit floating point number Y in BEXP, BARGBO, BARGB1.
       Use:
              CALL
                     POW24
       Output: 24 bit floating point number in AEXP, AARGBO, AARGB1
```

```
Result: AARG <-- POW( AARG )
        Testing on [1/26,26] from 100000 trials:
                 min
                          max
                                  mean
        Timing: 2852
                          4255
                                  3915.7 clks
                 min
                          max
                                  mean
                                           rms
                -0x6B
                          0x77
                                  -0.48
                                           16.49
        Error:
                                                    nsb
        Because of the availability of extended precision routines, the 24 bit
        power function can be estimated directly using the identity
                          x^*y = \exp(y^*\log(x))
        where the 32 bit exponential and natural log functions are called. A test
;
        for overflow from the product y*log(x) is performed explicitly, but the
        actual domain check is done in the exponential function.
POW24
                                  AARGB2,W
                                                    ; clear NSB
                 CLRF
                 BTFSC
                                  AARGB0,MSB
                                                    ; test if AARG < 0
                 GOTO
                                  DOMERR32
                 CPFSGT
                                  BEXP
                                                    ; if BARG=0, return 1.0
                 GOTO
                                  POW24ONE
                 MOVFP
                                  BEXP, WREG
                                                    ; save Y in ZARG
                 MOVWF
                                  ZARGB2
                                  BARGB0, WREG
                 MOVFP
                                  ZARGB0
                 MOVWF
                 MOVFP
                                  BARGB1, WREG
                 MOVWF
                                  ZARGB1
                                  WREG, F
                 CLRF
                                                    ; if AARG=0, return 0.0
                 CPESGT
                                  AEXP
                 GOTO
                                  POW24AZERO
                                  FPFLAGS, WREG
                 MOVFP
                                                    ; save RND flag in ZARGB3
                                  ZARGB3
                 MOVWF
                 BSF
                                  FPFLAGS, RND
                                                    ; enable rounding
                 CALL
                                  LOG32
                                                    ; log(x)
                                  ZARGB2, WREG
                 MOVFP
                 MOVWF
                                  BEXP
                 MOVFP
                                   ZARGB0, WREG
                 MOVWF
                                  BARGB0
                                  ZARGB1, WREG
                 MOVFP
                 MOVWF
                                  BARGB1
                 CLRF
                                  BARGB2,F
                 CALL
                                  FPM32
                                                    ; y*log(x)
                 TSTFSZ
                                  WREG
                                                    ; test for overflow
                                  DOMERR32
                 GOTO
                 BCF
                                  FPFLAGS, RND
                                                    ; disable rounding
                                  EXP32
                                                    ; \exp(y*log(x))
                 CALL
                 BTFSS
                                   ZARGB3, RND
                                   0x00
                 RETLW
```

```
BSF
                              FPFLAGS . RND
               CALL
                              RND4032
               RETLW
                              0x00
POW24ONE
               MOVLW
                              EXPBIAS
               MOVWF
                              AEXP
               CLRF
                              AARGB0,F
               CLRF
                              AARGB1,F
               RETLW
                              0x00
POW24AZERO
               BTFSS
                              BARGB0, MSB
                                              ; if x=0 and y<0, set overflow flag
               RETLW
                              0x00
               GOTO
                              SETFOV24
**************************
Evaluate pow(x,y) = X**Y
               32 bit floating point number X in AEXP, AARGBO, AARGB1, AARGB2 and
               32 bit floating point number Y in BEXP, BARGBO, BARGB1, BARGB2.
       Use:
               CALL
                       POW32
       Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2.
       Result: AARG <-- POW( AARG )
       Testing on [1/26,26] from 70000 trials:
               min
                       max
                              mean
       Timing: 4280
                       5574
                              5168.4 clks
               min
                              mean
                       max
                                      rms
       Error: -0x270
                      0x209
                              8.94
                                      92.21
                                              nsb
       The unavailability of extended precision routines for the 32 bit format
       requires considerably more effort with more sophisticated pseudo extended
       precision methods to control error propagation. Because the relative error
       in the exponential function is proportional to the absolute error of its
       argument, great care must be taken in any algorithm based on an exponential
       identity. Such methods generally rely on extracting as much of the result
       as an integer power of two as possible, followed by computations requiring
       approximations over a relatively small interval. To that end, consider the
       representation of the argument x given by
                      x=f*2**e, where .5 <= f < 1.
       The power function can then be expressed in the form
                       x^**y = 2^*(y^*\log_2(x)),
       with the base 2 log of x represented as
               log2(x) = log2(f*2**e) = e + log2(a) + log2(1+v), v = (f-a)/a,
       where a is chosen so that v is small. We choose a set of values of a defined
       by a(k)=2**(-k/16), k=0,1,\ldots 16, and for a given f, the value of a(k) for
       even k, nearest to f is chosen, resulting in an argument v to the natural
       log function
               log(1+v), 2**-(1/16)-1 < v < 2**(1/16)-1.
```

```
Since the numbers a(k) cannot be represented exactly in full precision, psuedo
;
        extended precision evaluation of v is performed through the expansion
                 v = (f-a(k))/a(k) = (f-A(k)-f*C(k))/A(k), C(k) = B(k)/A(k)
        where a(k) = A(k) + B(k). The number A(k) is equal to a(k) rounded to machine
;
        precision, and then B(k) is the difference computed in higher precision.
        This method assures evaluation of v with a maximum relative error less than
        1 ulp. A minimax approximation of the form
                 log(1+v) = v - .5*v**2 + (v**3)*(p(v)/q(v)),
        with first degree polynomials p and q, followed by conversion to the required
        function log2(1+v), leading to the result
                 log2(x) = e - k/16 + log2(1+v).
;
        The product y*log2(x) is now carefully computed by reducing the number y into
        a sum of two parts with one less than 1/16 and first evaluating small products
;
        of similar magnitude and collecting terms. Each stage of this strategy is
;
        followed by a similar reduction operation where the large part is an integer
;
        plus a number of 16ths. The final form of the product is then expressed as an
;
        integer plus a number of 16ths plus a number on the interval [-.0625,0],
        leading to a final result expressed in the form
                 x^*y = 2^*(y(\log_2(x)) = (2^*i)^*(2^*(-n/16))^*(2^*h),
        where 2**h is evaluated by a minimax approximation of the form
                 (2**h)-1 = h + h*p(h),
;
        with a second degree polynomial p.
POW32
                 CLRF
                                  AARGB3,W
                                                   ; clear NSB
                 BTFSC
                                  AARGB0,MSB
                                                   ; test if AARG < 0
                 GOTO
                                  DOMERR32
                                                   ; if BARG=0, return 1.0
                 CPFSGT
                                  BEXP
                 GOTO
                                  POW32ONE
                 MOVFP
                                  BEXP, WREG
                                                   ; save Y in CARG
                 MOVWF
                                  CEXP
                 MOVFP
                                  BARGBO, WREG
                 MOVWF
                                  CARGB0
                 MOVFP
                                  BARGB1, WREG
                 MOVWF
                                  CARGB1
                                  BARGB2, WREG
                 MOVED
                 MOVWF
                                  CARGB2
                 CLRF
                                  WREG, F
                                                   ; if AARG=0, return 0.0
                 CPESGT
                                  AEXP
                 GOTO
                                  POW32AZERO
                 MOVFP
                                  FPFLAGS, WREG
                                                   ; save RND flag in DARGB3
                 MOVWF
                                  DARGB3
                                  FPFLAGS, RND
                 BSF
                                                   ; enable rounding
        evaluate log2(x)
                                  AEXP, WREG
                 MOVFP
                                  WREG, TMR0L
                 MOVPF
                 MOVLW
                                  EXPBIAS-1
                 SUBWF
                                  TMR0L,F
```

MOVWF AEXP MOVLW 0x01AARGB7 MOVWF 0x09MOVLW MOVWF TEMPB0 POW32GETA CALL CALL TALEB32 TSTFSZ WREG MOVFP TEMPB0, AARGB7 MOVLW 0x04ADDWF AARGB7,W MOVWF TEMPB0 POW32GETA CALL CALL TALEB32 TSTFSZ WREG MOVFP TEMPB0, AARGB7 MOVLW  $0 \times 02$ ADDWF AARGB7,W TEMPB0 MOVWF CALL POW32GETA TALEB32 CALL TSTFSZ WREG MOVFP TEMPB0, AARGB7 MOVLW 0x01TEMPB0 MOVWF CALL POW32GETA CALL TAGEB32 MOVWF TEMPB0 CLRF WREG, F CPFSGT TEMPB0 POW32INCI GOTO MOVLW 0xFF MOVWF AARGB7 POW32INCI AARGB7,F INCF AARGB7, ZARGB0 MOVPF MOVFP AEXP, WREG ; DARG = X MOVWF DEXP AARGB0,DARGB0 MOVPF MOVPF AARGB1,DARGB1 MOVPF AARGB2,DARGB2 MOVPF AARGB7,TEMPB0 CALL POW32GETA CALL FPS32 MOVFP AEXP, WREG ; EARG = X-A1EEXP MOVWF

```
AARGB0, EARGB0
MOVPF
                  AARGB1, EARGB1
MOVPF
MOVPF
                  AARGB2, EARGB2
                  DEXP, WREG
MOVFP
MOVWF
                  AEXP
MOVFP
                  DARGB0, AARGB0
MOVFP
                  DARGB1, AARGB1
MOVFP
                  DARGB2, AARGB2
MOVPF
                  AARGB7, TEMPB0
CALL
                  POW32GETD
CALL
                  FPM32
TSTFSZ
                  AEXP
BTG
                  AARGB0,MSB
MOVFP
                  EEXP, WREG
                  BEXP
MOVWF
MOVFP
                  EARGB0, WREG
                  BARGB0
MOVWF
MOVFP
                  EARGB1, WREG
MOVWF
                  BARGB1
                  EARGB2, WREG
MOVFP
MOVWF
                  BARGB2
CALL
                  FPA32
                                    ; X - A1 - X * (A2/A1)
MOVFP
                  ZARGB0, WREG
MOVWF
                  TEMPB0
CALL
                  POW32GETA
CALL
                  FPD32
MOVFP
                  AEXP, WREG
                                    ; DARG = v = (X - A1 - X * (A2/A1))/A1
                  DEXP
MOVWF
MOVPF
                  AARGB0, DARGB0
MOVPF
                  AARGB1,DARGB1
MOVPF
                  AARGB2, DARGB2
POLL132
                  LOG32BQ,1,0
                                    iQ(z)
                  AEXP, WREG
MOVFP
MOVPF
                  WREG, FEXP
MOVPF
                  AARGB0, FARGB0
MOVPF
                  AARGB1, FARGB1
MOVPF
                  AARGB2, FARGB2
                  DEXP, WREG
MOVFP
MOVPF
                  WREG, AEXP
MOVFP
                  DARGB0, AARGB0
MOVFP
                  DARGB1, AARGB1
                  DARGB2, AARGB2
MOVFP
                  LOG32BP,1,0
POL32
                                    ; P(z)
MOVFP
                  FEXP, WREG
                  WREG, BEXP
MOVPF
MOVFP
                  FARGB0, WREG
                  WREG, BARGB0
MOVPF
MOVFP
                  FARGB1, WREG
MOVPF
                  WREG, BARGB1
MOVFP
                  FARGB2, WREG
MOVPF
                  WREG, BARGB2
CALL
                  FPD32
                                    ; P(z)/Q(z)
```

MOVFP	AEXP, WREG	; save in CARG
		, pave in out
MOVPF	WREG, FEXP	
MOVPF	AARGB0,FARGB0	
MOVPF	AARGB1,FARGB1	
MOVPF	AARGB2,FARGB2	
MOVFP	DEXP, WREG	
MOVPF	WREG,BEXP	
MOVFP	DARGB0, WREG	
MOVPF	WREG,BARGB0	
MOVFP	DARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	DARGB2,WREG	
MOVPF	WREG,BARGB2	
MOVFP	DEXP,WREG	
MOVPF	WREG, AEXP	
MOVFP	DARGB0, AARGB0	
MOVFP	DARGB1,AARGB1	
MOVFP	DARGB2,AARGB2	
CALL	FPM32	; z*z
MOVFP	AEXP, WREG	; save in EARG
MOVPF	WREG, EEXP	
MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1,EARGB1	
MOVPF	AARGB2,EARGB2	
MOVFP	FEXP, WREG	; z*z*P(z)/Q(z)
		, , , , , , , , , , , , , , , , , , , ,
MOVPF	WREG,BEXP	
MOVFP	FARGB0,WREG	
MOVPF	WREG,BARGB0	
MOVFP	FARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	FARGB2,WREG	
MOVPF	WREG,BARGB2	
CALL	FPM32	
MOVFP	DEXP, WREG	; z*(z*z*P(z)/Q(z))
		, 2 (2 2 1 (2), 2 (2),
MOVPF	WREG,BEXP	
MOVFP	DARGB0,WREG	
MOVPF	WREG, BARGB0	
MOVFP	DARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	DARGB2,WREG	
MOVPF	WREG, BARGB2	
CALL	FPM32	
MOVFP	EEXP, WREG	;5*z*z + z*(z*z*P(z)/Q(z))
		/ .5 2 2 1 2 (2 2 1 (2)/ Q(2))
MOVPF	WREG,BEXP	
MOVFP	EARGB0, WREG	
MOVPF	WREG,BARGB0	
MOVFP	EARGB1,WREG	
MOVPF	WREG,BARGB1	
MOVFP	EARGB2,WREG	
MOVPF	WREG, BARGB2	
TSTFSZ	BEXP	
DECF	BEXP,F	
CALL	FPS32	
MOVFP	AEXP, WREG	; save in EARG
		, bave in EARG
MOVWF	EEXP	

MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1,EARGB1	
MOVPF	AARGB2,EARGB2	
HOVII	AAKODZ , EAKODZ	
MOLITE	070	
MOVLW	0x7D ; LOG2(e) - 1	
MOVWF	BEXP	
MOVLW	0x62	
MOVWF	BARGB0	
MOVLW	0xA8	
MOVWF	BARGB1	
MOVLW	0xED	
MOVWF	BARGB2	
MOVWE	DARGDZ	
63.7.7		
CALL	FPM32	
MOVFP	EEXP, WREG	
MOVWF	BEXP	
MOVFP	EARGB0, WREG	
MOVWF	BARGB0	
MOVFP	EARGB1,WREG	
MOVWF	BARGB1	
MOVFP	EARGB2, WREG	
MOVWF	BARGB2	
CALL	FPA32	
MOVFP	AEXP, WREG ; save in EARG	G
MOVWF	EEXP	
MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1,EARGB1	
MOVPF	AARGB2, EARGB2	
	,	
MOVFP	DEXP, WREG	
MOVWF	AEXP	
MOVFP	DARGBO, AARGBO	
MOVFP	DARGB1, AARGB1	
MOVFP	DARGB2, AARGB2	
MOVLW	0x7D ; LOG2(e) - 1	
MOVWF	BEXP	
MOVLW	0x62	
MOVWF	BARGB0	
MOVLW	0xA8	
MOVWF	BARGB1	
MOVLW	0xED	
MOVWF	BARGB2	
MOVWE	DARGDZ	
CATT	EDW20	
CALL	FPM32	
	_	
MOVFP	EEXP, WREG	
MOVWF	BEXP	
MOVFP	EARGB0, WREG	
MOVWF	BARGB0	
MOVFP	EARGB1,WREG	
MOVWF	BARGB1	
MOVFP	EARGB2,WREG	
MOVWF	BARGB2	
I-IO A M.L.	PUNDS	
CATT	ED # 2.2	
CALL	FPA32	
MOVFP	DEXP, WREG	
MOVWF	BEXP	
MOVFP	DARGBO, WREG	
MOVWF	BARGB0	
MOVFP	DARGB1,WREG	
MOVWF	BARGB1	
* **		

MOVFP	DARGB2,WREG	
MOVWF	BARGB2	
CALL	FPA32	
MOVFP	AEXP.WREG	; save z in EARG
MOVWF	EEXP	
MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1, EARGB1	
MOVPF	AARGB2,EARGB2	
MOVFP	ZARGB0,AARGB1	; $w = -i / 16$
CLRF	AARGB0,F	
CALL	FL01624	
CLRF	AARGB2,F	
TSTFSZ	AEXP	
BSF	AARGB0,MSB	
MOVLW	0x04	
TSTFSZ	AEXP	
SUBWF	AEXP,F	
MOVFP	AEXP, WREG	; save w in BARG
MOVWF	BEXP	
MOVPF	AARGB0,BARGB0	
MOVPF	AARGB1,BARGB1	
MOVPF	AARGB2,BARGB2	
MOVET	AAKGBZ, BAKGBZ	
MOMED	TMD OI A A DCD 1	
MOVFP	TMROL, AARGB1	, w = w + e
CLRF	AARGB0,F	
BTFSC	AARGB1,MSB	
COMF	AARGB0,F	
CALL	FL01624	
CLRF	AARGB2,F	
CALL	FPA32	
MOVFP	AEXP,WREG	; save w in FARG
MOVWF	FEXP	
MOVPF	AARGB0,FARGB0	
MOVPF	AARGB1,FARGB1	
MOVPF	AARGB2,FARGB2	
MOVFP	CEXP, WREG	
MOVWF	AEXP	
MOVFP	CARGB0, AARGB0	
MOVFP	CARGB1,AARGB1	
MOVFP	CARGB2,AARGB2	
CALL	REDUCE	; AARG = Yb, DARG = Ya
MOVFP	FEXP, WREG	
MOVWF	BEXP	
MOVFP	FARGB0, WREG	
MOVWF	BARGB0	
MOVFP	FARGB1,WREG	
MOVWF	BARGB1	
MOVFP	FARGB2, WREG	
MOVWF	BARGB2	
CALL	FPM32	
- <del>-</del>	*=	
MOVFP	AEXP, WREG	; save w * Yb in GARG
		, bave w in iii dand
MOVWF	GEXP	
MOVPF	AARGB0,GARGB0	
MOVPF	AARGB1,GARGB1	
MOVPF	AARGB2,GARGB2	

MOVFP	EEXP, WREG	
MOVWF	AEXP	
MOVFP	EARGB0, AARGB0	
MOVFP	EARGB1, AARGB1	
MOVFP	EARGB2,AARGB2	
MOVFP	CEXP, WREG	
MOVWF	BEXP	
MOVFP	CARGB0, WREG	
MOVWF	BARGB0	
MOVFP	CARGB1,WREG	
MOVWF	BARGB1	
MOVFP	CARGB2,WREG	
MOVWF	BARGB2	
MOVWF	DARGDZ	
CATT	EDM20	
CALL	FPM32	
MOVFP	GEXP, WREG	
MOVWF	BEXP	
MOVFP	GARGB0, WREG	
MOVWF	BARGB0	
MOVFP	GARGB1,WREG	
MOVWF	BARGB1	
MOVFP	GARGB2,WREG	
MOVWF	BARGB2	
MOVWF	BARGBZ	
a	=== 2.0	
CALL	FPA32	
MOVFP	DEXP, WREG	; move Ya to CARG
MOVWF	CEXP	
MOVFP	DARGB0, WREG	
MOVWF	CARGB0	
MOVFP	DARGB1,WREG	
MOVWF	CARGB1	
MOVED	CARGB1	
MOVFP	DARGB2,WREG	
MOVFP MOVWF	DARGB2,WREG CARGB2	A AADG TIL DADG TI
MOVFP	DARGB2,WREG	; AARG = Fb, DARG = Fa
MOVFP MOVWF CALL	DARGB2, WREG CARGB2 REDUCE	
MOVFP MOVWF CALL MOVFP	DARGB2, WREG CARGB2 REDUCE AEXP, WREG	; AARG = Fb, DARG = Fa ; save Fb in EARG
MOVFP MOVWF CALL	DARGB2, WREG CARGB2 REDUCE	
MOVFP MOVWF CALL MOVFP	DARGB2, WREG CARGB2 REDUCE AEXP, WREG	
MOVFP MOVWF CALL MOVFP MOVWF	DARGB2, WREG CARGB2 REDUCE AEXP, WREG EEXP	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF	DARGB2,WREG CARGB2 REDUCE AEXP,WREG EEXP AARGB0,EARGB0	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVPF	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVPF MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2  FEXP,WREG BEXP	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2  FEXP,WREG BEXP FARGB0,WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2  FEXP,WREG BEXP FARGB0,WREG BARGB0	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2  FEXP,WREG BEXP FARGB0,WREG BARGB0 FARGB1,WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP	DARGB2,WREG CARGB2  REDUCE  AEXP,WREG EEXP AARGB0,EARGB0 AARGB1,EARGB1 AARGB2,EARGB2  FEXP,WREG BEXP FARGB0,WREG BARGB0 FARGB1,WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB1, WREG BARGB1 FARGB2, WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB1, WREG BARGB1 FARGB2, WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP MOVFP MOVWF	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 FARGB2, WREG BARGB2 CEXP, WREG	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVFP MOVWF MOVFP MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CEXP, WREG BARGB2  CEXP, WREG AEXP CARGB0, AARGB0	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CEXP, WREG BARGB2  CEXP, WREG AEXP CARGB0, AARGB0 CARGB1, AARGB1	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVFP MOVWF MOVFP MOVFP MOVFP MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CEXP, WREG BARGB2  CEXP, WREG AEXP CARGB0, AARGB0	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP MOVWF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB0, AARGB0 CARGB1, AARGB1 CARGB2, AARGB2	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP MOVFP MOVFP MOVFP MOVWF MOVFP MOVWF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CEXP, WREG BARGB2  CEXP, WREG AEXP CARGB0, AARGB0 CARGB1, AARGB1	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB0, AARGB0 CARGB1, AARGB1 CARGB2, AARGB2	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB0, AARGB0 CARGB1, AARGB1 CARGB2, AARGB2  FPM32  DEXP, WREG	
MOVFP MOVWF  CALL  MOVFP MOVPF MOVPF MOVPF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB0, AARGB0 CARGB1, AARGB1 CARGB2, AARGB2	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVPF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB0, AARGB0 CARGB1, AARGB1 CARGB2, AARGB2  FPM32  DEXP, WREG	
MOVFP MOVWF  CALL  MOVFP MOVWF MOVPF MOVFP	DARGB2, WREG CARGB2  REDUCE  AEXP, WREG EEXP AARGB0, EARGB0 AARGB1, EARGB1 AARGB2, EARGB2  FEXP, WREG BEXP FARGB0, WREG BARGB0 FARGB1, WREG BARGB1 FARGB2, WREG BARGB1 CARGB2, AARGB1 CARGB1, AARGB1 CARGB2, AARGB2  FPM32  DEXP, WREG BEXP	

MOVFP	DARGB1,WREG	
MOVWF	BARGB1	
MOVFP	DARGB2,WREG	
MOVWF	BARGB2	
O 3 T T	EDAGO	
CALL	FPA32	
_	-	
CALL	REDUCE	; AARG = Gb, DARG = Ga
MOVFP	EEXP,WREG	
MOVWF	BEXP	
MOVFP	EARGB0, WREG	
MOVWF	BARGB0	
MOVFP	EARGB1,WREG	
MOVWF	BARGB1	
MOVFP		
	EARGB2,WREG	
MOVWF	BARGB2	
G3.T.T	=== 2.0	
CALL	FPA32	
MOVFP	DEXP, WREG	; move Ga to CARG
MOVWF	CEXP	
MOVFP	DARGB0, WREG	
MOVWF	CARGB0	
MOVFP	DARGB1,WREG	
MOVWF	CARGB1	
MOVFP	DARGB2,WREG	
MOVWF	CARGB2	
MOVWI	CARGDZ	
C3.1.1	DEDUCE	. AADG UI- DADG UI-
CALL	REDUCE	; AARG = Hb, DARG = Ha
MOVFP		; save Hb in EARG
MOVWF	EEXP	
MOVPF	AARGB0,EARGB0	
MOVPF	AARGB1,EARGB1	
MOVPF	AARGB2, EARGB2	
MOVFP	CEXP, WREG	
MOVWF	AEXP	
MOVFP	CARGB0, AARGB0	
MOVFP	CARGB1, AARGB1	
MOVFP	CARGB2,AARGB2	
MOVIE	CARGDZ, AARGDZ	
MOMED	DEAD MDEG	
MOVFP	DEXP, WREG	
MOVWF	BEXP	
MOVFP	DARGB0, WREG	
MOVWF	BARGB0	
MOVFP	DARGB1,WREG	
MOVWF	BARGB1	
MOVFP	DARGB2,WREG	
MOVWF	BARGB2	
CALL	FPA32	
MOVLW	0x04	
TSTFSZ	AEXP	
ADDWF	AEXP,F	
ADDWF	ALAF, F	
DCF	בטבו אכנ האה	
BCF	FPFLAGS, RND	
CALL	INT3224	
BSF	FPFLAGS, RND	
MOVFP	AARGB1,WREG	; test for overflow
BTFSC	AARGB1,MSB	
NEGW	WREG, F	
BTFSC	WREG, 4	; is  e  < 2048 ?
GOTO	DOMERR32	1 1 2 2 2
	- <del></del>	

	MOVPF MOVPF	AARGB1,ZARGB0 AARGB2,ZARGB1	; save e in ZARGB0,ZARGB1
	BTFSC GOTO	EARGB0,MSB POW32HBOK	
	CLRF	WREG, F	
	INCF	ZARGB1,F	
	ADDWFC	ZARGB0,F	
	MOVFP	EEXP, WREG	
	MOVWF	AEXP	
	MOVFP	EARGB0,AARGB0	
	MOVFP	EARGB1,AARGB1	
	MOVFP	EARGB2,AARGB2	
	MOVLW	0x7B	
	MOVWF	BEXP	
	MOVLW	0x80	
	MOVWF	BARGB0	
	CLRF	BARGB1,F	
	CLRF	BARGB2,F	
	CALL	FPA32	
	MOVFP	AEXP, WREG	; save Hb in EARG
	MOVWF	EEXP	
	MOVPF	AARGB0,EARGB0	
	MOVPF	AARGB1,EARGB1	
	MOVPF	AARGB2,EARGB2	
POW32HBOK			
	MOVFP	EEXP, WREG	
	MOVWF	AEXP	
	MOVFP	EARGB0,AARGB0	
	MOVFP	EARGB1,AARGB1	
	MOVFP	EARGB2, AARGB2	
	MOVFP	AEXP, WREG	
	MOVWF	DEXP	
	MOVPF	AARGB0,DARGB0	
	MOVPF	AARGB1,DARGB1	
	MOVPF	AARGB2,DARGB2	
	BSF	FPFLAGS,RND	
	POL32	EXP232,2,0	z = 2**Hb - 1
	MOVFP	DEXP, WREG	
	MOVWF	BEXP	
	MOVFP	DARGB0, WREG	
	MOVWF	BARGB0	
	MOVFP	DARGB1,WREG	
	MOVWF	BARGB1	
	MOVFP MOVWF	DARGB2,WREG BARGB2	
	HOVWE	DANGDZ	
	CALL	FPM32	
	MOVFP	ZARGB0, WREG	
	MOVWF	ZARGB2	
	MOVFP	ZARGB1,WREG	
	MOVWF	ZARGB3	
	CLRF	GARGB3,F	
	BTFSS	ZARGB0,MSB	

INCF	GARGB3,F
BCF	_C
RRCF	ZARGB2,F
RRCF	ZARGB3,F
RRCF	ZARGB2,F
RRCF	ZARGB3,F
RRCF	ZARGB2,F
RRCF	ZARGB3,F
RRCF	ZARGB2,F
RRCF	ZARGB3,F
BTFSC	ZARGB0,MSB
INCF	ZARGB3,F
MOVFP	ZARGB3,WREG
ADDWF	GARGB3,F
MOLIED	GARGRA MREG
MOVFP	GARGB3, WREG
MULLW	0x10
MOVLW	0x10
BTFSC	GARGB3,MSB
SUBWF	PRODH,F
	515651 ID56
MOVFP	ZARGB1,WREG
SUBWF	PRODL,F
MOVFP	ZARGB0,WREG
SUBWFB	PRODH,F
MOVFP	PRODL, WREG
MOVWF	ZARGB3
MOVWF	TEMPB0
HOVWI	TEMI DO
CALL	POW32GETA
CALL	FPM32
MOVFP	ZARGB3,WREG
MOVWF	TEMPB0
CALL	POW32GETC
TSTFSZ	BEXP
INCF	BEXP,F
IIVCI	DEXI , I
CALL	FPA32
MOVFP	ZARGB3,WREG
MOVWF	TEMPB0
CALL	POW32GETA
BTFSS	DARGB3,RND
BCF	FPFLAGS,RND
CALL	FPA32
MOVFP	GARGB3,WREG
TSTFSZ	AEXP
ADDWF	AEXP,F
RETLW	0x00
MOVLW	EXPBIAS
MOVWF	AEXP
CLRF	AARGB0,F
CLRF	AARGB1,F
CLRF	AARGB2,F
RETLW	0x00

POW32ONE

```
POW32AZERO
                                   BARGB0,MSB
                                                    ; if x=0 and y<0, set overflow flag
                 BTFSS
                 RETLW
                                   0x00
                 GOTO
                                   SETFOV32
REDUCE
                                                    ; BARG = X
                 MOVFP
                                  AEXP, WREG
                                  BEXP
                 MOVWF
                 MOVPF
                                  AARGB0,BARGB0
                 MOVPF
                                  AARGB1,BARGB1
                 MOVPF
                                  AARGB2, BARGB2
                                   0x04
                 MOVLW
                 ADDWF
                                  AEXP,F
                 CALL
                                  FLOOR32
                 MOVLW
                                   0x04
                 TSTFSZ
                                  AEXP
                 SUBWF
                                  AEXP,F
                 MOVFP
                                   AEXP, WREG
                                                    ; DARG = Xa
                 MOVWF
                                   DEXP
                                  AARGB0, DARGB0
                 MOVPF
                 MOVPF
                                  AARGB1, DARGB1
                 MOVPF
                                  AARGB2, DARGB2
                 BTG
                                  AARGB0,MSB
                                                    ; AARG = Xb
                 CALL
                                   FPA32
                 RETLW
                                   0x00
POW32GETA
                 MOVLW
                                  HIGH (POW32TABLEA); access table for A
                 MOVWF
                                  TBLPTRH
                 RLNCF
                                  TEMPB0,W
                 ADDLW
                                  LOW (POW32TABLEA)
                 MOVWF
                                  TBLPTRL
                 BTFSC
                                   _C
                 INCF
                                  TBLPTRH, F
                                  0,1,BEXP
                 TABLRD
                 TLRD
                                  1,BEXP
                 TABLRD
                                  0,1,BARGB0
                                  1,BARGB1
                 TIRD
                 TABLRD
                                  0,0,BARGB2
                 RETLW
                                   0x00
POW32GETC
                 MOVLW
                                  HIGH (POW32TABLEC); access table for A
                 MOVWF
                                  TBLPTRH
                                  TEMPB0,W
                 RLNCF
                 ADDLW
                                  LOW (POW32TABLEC)
                 MOVWF
                                  TBLPTRL
                 BTFSC
                                   _C
                                  TBLPTRH, F
                 INCF
                 TABLED
                                   0,1,BEXP
                 TLRD
                                  1,BEXP
                 TABLRD
                                   0,1,BARGB0
                 TLRD
                                   1,BARGB1
                                   0,0,BARGB2
                 TABLRD
```

```
RETLW
                                  0x00
POW32GETD
                 MOVLW
                                  HIGH (POW32TABLED); access table for A
                 MOVWF
                                  TRIPTRH
                 RLNCF
                                  TEMPB0,W
                 ADDLW
                                  LOW (POW32TABLED)
                 MOVWF
                                  TBLPTRL
                                  _C
                 BTFSC
                 INCF
                                  TBLPTRH, F
                 TABLRD
                                  0,1,BEXP
                 TLRD
                                  1,BEXP
                 TABLRD
                                  0,1,BARGB0
                                  1,BARGB1
                 תאיזע
                 TABLRD
                                  0,0,BARGB2
                 RETLW
                                  0x00
        minimax rational coefficients for log2(1+z)/z on [-.0625,.0625]
                                                    ; LOG232P0 = .73551298732E+1******
LOG232P0
                 EQU
                                  0x81
LOG232P00
                 EQU
                                  0x19
LOG232P01
                                  0xB1
                 EQU
LOG232P02
                                  0xA6
                 EQU
LOG232P1
                 EQU
                                  0x80
                                                    ; LOG232P1 = .40900513905E+1
LOG232P10
                                  0x57
                 EQU
LOG232P11
                 EQU
                                  0x5A
LOG232P12
                 EQU
                                  0x68
LOG232P2
                 EQU
                                  0x7C
                                                    ; LOG232P1 = .40900513905E+1
LOG232P20
                 EOU
                                  0x24
LOG232P21
                                  0x58
                 EQU
LOG232P22
                 EQU
                                  0x44
LOG232Q0
                 EQU
                                  0x80
                                                    ; LOG232Q0 = .50982159260E+1
LOG232Q00
                                  0x55
                 EQU
LOG232Q01
                 EQU
                                  0x10
LOG232Q02
                 EQU
                                  0xA7
LOG232Q1
                 EQU
                                  0x80
                                                    ; LOG232Q1 = .53849258895E+1
LOG232Q10
                                  0x7F
                 EQU
LOG232Q11
                 EQU
                                  0xCD
LOG232Q12
                 EQU
                                  0xD0
LOG232Q2
                 EQU
                                  0x7F
                                                    ; LOG232Q2 = 1.0
LOG232Q20
                 EQU
                                  0x00
LOG232Q21
                 EQU
                                  0x00
LOG232Q22
                 EQU
                                  0x00
        minimax rational approximationz-.5*z*z+z*(z*z*P(z)/Q(z))
LOG32AP0
                 EQU
                                  0x7D
                                                    ; LOG32AP0 = .4165382203229886
                                  0x55
LOG32AP00
                 EQU
LOG32AP01
                 EQU
                                  0x44
LOG32AP02
                 EQU
                                  0x7F
LOG32AP1
                                  0x79
                                                    ; LOG32AP1 = .02090135006173772
                 EQU
LOG32AP10
                                  0x2B
                 EQU
LOG32AP11
                 EQU
                                  0x39
LOG32AP12
                 EQU
                                  0x4F
```

```
LOG32AO0
                 EOU
                                   0x7F
                                                    ; LOG32AO0 = 1.249615003891314
LOG32AQ00
                 EQU
                                   0x1F
LOG32AQ01
                 EQU
                                   0xF3
LOG32AQ02
                 EQU
                                   0x62
LOG32AQ1
                 EQU
                                   0x7F
                                                    ; LOG32AQ1 = 1.0
LOG32AQ10
                 EQU
                                   0x00
LOG32AQ11
                 EQU
                                   0x00
LOG32AQ12
                                   0x00
                 EQU
       minimax rational approximationz-.5*z*z+z*(z*z*P(z)/Q(z))
LOG32BP0
                 EQU
                                   0x7D
                                                    ; LOG32BP0 = .4165382203229886****
LOG32BP00
                 EQU
                                   0x55
LOG32BP01
                 EQU
                                   0x57
LOG32BP02
                                   0x8F
                 EQU
                                   0x79
                                                    ; LOG32BP1 = .02090135006173772
LOG32BP1
                 EQU
LOG32BP10
                 EQU
                                   0x2A
LOG32BP11
                 EQU
                                   0x72
LOG32BP12
                 EQU
                                   0xAE
LOG32BQ0
                 EQU
                                   0x7F
                                                    ; LOG32BQ0 = 1.249615003891314
LOG32BQ00
                 EQU
                                   0x20
LOG32BQ01
                                   0x01
                 EQU
LOG32BQ02
                 EQU
                                   0xAB
LOG32BQ1
                 EQU
                                   0x7F
                                                    ; LOG32BQ1 = 1.0
LOG32BQ10
                 EQU
                                   0x00
LOG32BQ11
                 EQU
                                   0x00
LOG32BQ12
                                   0x00
                 EOU
        second degree minimax polynomial coefficients for 2**(x)-1 on [-.0625,0]
EXP2320
                 EQU
                                   0x7E
                                                    ; EXP2320 = .693146757796576
EXP23200
                 EQU
                                   0x31
EXP23201
                 EQU
                                   0x72
EXP23202
                 EQU
                                   0x11
EXP2321
                 EQU
                                   0x7C
                                                    ; EXP2321 = .2401853543026017
EXP23210
                 EQU
                                   0x75
EXP23211
                 EQU
                                   0xF3
EXP23212
                                   0x26
                 EQU
EXP2322
                 EQU
                                   0x7A
                                                    ; EXP2322 = .05436330184989159
EXP23220
                 EQU
                                   0x5E
EXP23221
                 EQU
                                   0xAC
EXP23222
                 EOU
                                   0 \times 0 E
        second degree minimax polynomial coefficients for 2**(x)-1 on [-.0625,0]
EXP232A0
                 EQU
                                   0x7E
                                                    ; EXP232A0 = .693146757796576****
EXP232A00
                 EQU
                                   0x31
EXP232A01
                 EQU
                                   0x72
EXP232A02
                 EQU
                                   0x11
EXP232A1
                 EQU
                                   0x7C
                                                    ; EXP232A1 = .2401853543026017
EXP232A10
                                   0x75
                 EQU
EXP232A11
                 EQU
                                   0xF3
EXP232A12
                 EQU
                                   0x26
```

EXP232A2	EQU		0x7A	; EXP232A2 =	.05436330184989159
EXP232A20	EQU		0x5E		
EXP232A21 EXP232A22	EQU EQU		0xAC 0x0E		
EAP Z 3 Z A Z Z	БQU		OXOE		
;					
DOM 2 OFFI DI TIA					
POW32TABLEA	DATA	0x7F00			
	DATA	0x0000			
	DATA	0x7E75			
	DATA	0x257D			
	DATA	0x7E6A			
	DATA	0xC0C7			
	DATA	0x7E60			
	DATA	0xCCDF			
	DATA	0x7E57			
	DATA DATA	0x44FD 0x7E4E			
	DATA	0x7E1E			
	DATA	0x7E45			
	DATA	0x672A			
	DATA	0x7E3D			
	DATA	$0 \times 08A4$			
	DATA	0x7E35			
	DATA	0x04F3			
	DATA	0x7E2D			
	DATA	0x583F			
	DATA DATA	0x7E25 0xFED7			
	DATA	0x7E1E			
	DATA	0xF532			
	DATA	0x7E18			
	DATA	0x37F0			
	DATA	0x7E11			
	DATA	0xC3D3			
	DATA	0x7E0B			
	DATA DATA	0x95C2 0x7E05			
	DATA	0xAAC3			
	DATA	0x7E00			
	DATA	0x0000			
POW32TABLEC	DAMA	00000			
	DATA DATA	$0 \times 00000$ $0 \times 00000$			
	DATA	0x6329			
	DATA	0x2436			
	DATA	0x63C1			
	DATA	0x16DE			
	DATA	0x639E			
	DATA	0xAB59			
	DATA	0x64D4			
	DATA	0xA58A			
	DATA DATA	0x6328 0xFC24			
	DATA	0xfC24 0x630A			
	DATA	0xA837			
	DATA	0x65C1			
	DATA	0x4FE8			
	DATA	0x644F			
	DATA	0xE77A			
	DATA	0x63AD			
	DATA	0xEAF6			
	DATA	0x65AC			

```
0 \times 9 D5 E
                 DATA
                 DATA
                          0x6541
                 DATA
                          0x2342
                 DATA
                          0x6523
                 DATA
                          0x1B71
                          0x6567
                 DATA
                          0x5624
                 DATA
                 DATA
                          0x63E0
                 DATA
                          0xABA1
                          0x654F
                 DATA
                 DATA
                          0x9891
                 DATA
                          0x0000
                 DATA
                          0x0000
POW32TABLED
                 DATA
                          0x0000
                 DATA
                          0x0000
                 DATA
                          0x63B0
                 DATA
                          0xA146
                 DATA
                          0x6352
                 DATA
                          0x90BE
                 DATA
                          0x6334
                 DATA
                          0xB0DA
                                            ; +1 647CE183
                          0x647C
                 DATA
                          0xE182
                 DATA
                          0x63D1
                 DATA
                 DATA
                          0xDAF2
                 DATA
                          0x63B3
                 DATA
                          0xD0E5
                 DATA
                          0x6602
                 DATA
                          0xE5A2
                 DATA
                          0x6593
                                            ; -1 659302AE
                 DATA
                          0x02AF
                 DATA
                          0 \times 6400
                 DATA
                          0x6C56
                 DATA
                          0x6605
                 DATA
                          0x1AA9
                 DATA
                          0x669B
                 DATA
                          0x85F2
                 DATA
                          0x6689
                 DATA
                          0x2801
                 DATA
                          0x66CB
                          0x2482
                 DATA
                                            ; +1 644E0611
                 DATA
                          0 \times 644 E
                          0x0610
                 DATA
                 DATA
                          0x66C6
                 DATA
                          0xCB6A
                          0x0000
                 DATA
                 DATA
                          0x0000
        Evaluate floor(x)
;
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Use:
                 CALL
                          FLOOR32
;
        Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
        Result: AARG <-- FLOOR( AARG )
```

```
Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                 min
                                   mean
                          max
                                   35.2
        Timing: 30
                          45
                                            clks
                 min
                          max
                                   mean
                                            rms
        Error: 0x00
                          0x00
                                   0.0
                                            0.0
                                                     nsb
FLOOR32
                 CLRF
                                   AARGB3,W
                                                     ; test for zero argument
                 CPFSGT
                                   AEXP
                 RETLW
                                   0x00
                 MOVFP
                                   AARGB0, AARGB4
                                                     ; save mantissa
                                   AARGB1,AARGB5
                 MOVFP
                 MOVFP
                                   AARGB2, AARGB6
                                   EXPBIAS
                 MOVLW
                 SUBWF
                                   AEXP,W
                 BTFSC
                                   WREG, MSB
                 GOTO
                                   FLOOR32ZERO
                 SUBLW
                                   0x18-1
                 MOVWF
                                   TEMPB0
                                                     ; save number of zero bits in TEMPBO
                                                     ; divide by eight
                 BTFSC
                                   WREG, LSB+1+3
                 GOTO
                                   FLOOR32MASKH
                 BTFSC
                                   WREG, LSB+3
                 GOTO
                                   FLOOR32MASKM
FLOOR32MASKL
                                   TBLPTRH, F
                 CLRF
                 MOVFP
                                   TEMPB0, WREG
                                                     ; get remainder for mask pointer
                 ANDLW
                                   0x07
                 M'IUU'
                                   LOW (FLOOR32MASKTABLE)
                 MOVWF
                                   TBLPTRL
                 MOVLW
                                   HIGH (FLOOR32MASKTABLE); access table for F0
                 ADDWFC
                                   TBLPTRH, F
                                   0,1,WREG
                 TABLRD
                 TLRD
                                   0,WREG
                 ANDWF
                                   AARGB2,F
                 BTFSS
                                   AARGB0,MSB
                                                     ; if negative, round down
                                   0x00
                 RETLW
                 MOVWF
                                   AARGB7
                 MOVFP
                                   AARGB6, WREG
                                   AARGB2
                 CPFSEQ
                 GOTO
                                   FLOOR32RNDL
                 RETLW
                                   0x00
FLOOR32RNDL
                 COMF
                                   AARGB7,W
                 INCF
                                   WREG, F
                 ADDWF
                                   AARGB2,F
                 CLRF
                                   WREG, F
                 ADDWFC
                                   AARGB1,F
                 ADDWFC.
                                   AARGB0,F
                 BTFSS
                                                     ; has rounding caused carryout?
                                   C
                                   0x00
                 RETLW
                 RRCF
                                   AARGB0,F
                                   AARGB1,F
                 RRCF
```

```
RRCF
                                   AARGB2,F
                  INCFSZ
                                   AEXP,F
                                                     ; check for overflow
                 RETLW
                                    0 \times 00
                  GOTO
                                   SETFOV32
FLOOR32MASKM
                  CLRF
                                   TBLPTRH, F
                 MOVFP
                                   TEMPB0, WREG
                                   0x07
                 ANDLW
                 ADDLW
                                   LOW (FLOOR32MASKTABLE)
                 MOVWF
                                   TBLPTRL
                 MOVLW
                                   HIGH (FLOOR32MASKTABLE); access table for F0
                 ADDWFC
                                   TBLPTRH, F
                 TABLRD
                                   0,1,WREG
                                   0,WREG
                  TLRD
                 ANDWF
                                   AARGB1,F
                 CLRF
                                   AARGB2,F
                                   AARGB0,MSB
                 BTFSS
                                                      ; if negative, round down
                  RETLW
                                   0x00
                 MOVWF
                                   AARGB7
                                   AARGB6, WREG
                 MOVFP
                 CPFSEQ
                                   AARGB2
                  GOTO
                                   FLOOR32RNDM
                 MOVFP
                                   AARGB5, WREG
                 CPFSEQ
                                   AARGB1
                  GOTO
                                   FLOOR32RNDM
                  RETLW
                                    0 \times 00
FLOOR32RNDM
                  COMF
                                   AARGB7,W
                                   WREG, F
                  INCF
                 ADDWF
                                   AARGB1,F
                 CLRF
                                   WREG, F
                                   AARGB0,F
                 ADDWFC
                 BTFSS
                                                     ; has rounding caused carryout?
                                    _C
                 RETLW
                                   0x00
                  RRCF
                                   AARGB0,F
                 RRCF
                                   AARGB1,F
                 RRCF
                                   AARGB2,F
                  INCFSZ
                                   AEXP,F
                                                     ; check for overflow
                                   0x00
                 RETLW
                  GOTO
                                   SETFOV32
FLOOR32MASKH
                  CLRF
                                   TBLPTRH, F
                 MOVFP
                                   TEMPB0, WREG
                                   0x07
                 ANDLW
                 ADDLW
                                   LOW (FLOOR32MASKTABLE)
                 MOVWF
                                   TBLPTRL
                 MOVLW
                                   HIGH (FLOOR32MASKTABLE); access table for F0
                                   TBLPTRH, F
                 ADDWFC
                 TABLRD
                                   0,1,WREG
                  TLRD
                                   0,WREG
                                   AARGB0,F
                 ANDWF
                 CLRF
                                   AARGB1,F
                  CLRF
                                   AARGB2,F
                  BTFSS
                                   AARGB0,MSB
                                                      ; if negative, round down
                 RETLW
                                   0x00
```

```
MOVWF
                                   AARGB7
                 MOVFP
                                   AARGB6, WREG
                 CPFSEQ
                                   AARGB2
                 GOTO
                                   FLOOR32RNDH
                 MOVFP
                                   AARGB5, WREG
                 CPFSEQ
                                   AARGB1
                                   FLOOR32RNDH
                 GOTO
                 MOVFP
                                   AARGB4, WREG
                 CPFSEQ
                                   AARGB0
                                   FLOOR32RNDH
                 GOTO
                 RETLW
                                   0x00
FLOOR32RNDH
                 COMF
                                   AARGB7,W
                                   WREG, F
                 INCF
                 ADDWF
                                   AARGB0,F
                 BTFSS
                                   _C
                                                    ; has rounding caused carryout?
                 RETLW
                                   0x00
                 RRCF
                                   AARGB0,F
                 RRCF
                                   AARGB1,F
                                   AEXP,F
                                                    ; check for overflow
                 INCFSZ
                 RETLW
                                   0x00
                 GOTO
                                   SETFOV32
FLOOR32ZERO
                                   AARGB0,MSB
                 BTFSC
                 GOTO
                                   FLOOR32MINUSONE
                                  AEXP,F
                 CLRF
                 CLRF
                                   AARGB0,F
                 CLRF
                                   AARGB1,F
                 CLRF
                                   AARGB2,F
                 RETLW
                                   0x00
FLOOR32MINUSONE
                                   0x7F
                 MOVLW
                 MOVWF
                                  AEXP
                                   0x80
                 MOVLW
                 MOVWF
                                   AARGB0
                 CLRF
                                   AARGB1,F
                 CLRF
                                   AARGB2,F
                 RETLW
                                   0x00
        table for least significant byte requiring masking, using pointer from
        the remainder of the number of zero bits divided by eight.
FLOOR32MASKTABLE
                 DATA
                                   0xFF
                 DATA
                                   0xFE
                 DATA
                                   0xFC
                 DATA
                                   0xF8
                                   0xF0
                 DATA
                                   0xE0
                 DATA
                 DATA
                                   0xC0
                 DATA
                                   0x80
                                   0x00
                 DATA
        Evaluate rand(x)
        Input: 32 bit initial integer seed in RANDB0, RANDB1, RANDB2, RANDB3
        Use:
                 CALL
                          RAND32
```

```
Output: 32 bit random integer in RANDB0, RANDB1, RANDB2, RANDB3
        Result: RAND <-- RAND32( RAND )
        Timing: 4+6+2+90+15 = 117 clks
        Linear congruential random number generator
               X \leftarrow (a * X + c) \mod m
        The calculation is performed exactly, with multiplier a, increment c, and
        modulus m, selected to achieve high ratings from standard spectral tests.
        The dedicated storage in RANDBx retains the current number in the sequence
        and is not used by any other routine in the library. The initial seed, X0,
        is arbitrary and must be placed in RANDBx.
RAND32
               MOVFP
                               RANDB0, AARGB0
               MOVFP
                               RANDB1, AARGB1
               MOVFP
                               RANDB2, AARGB2
                               RANDB3, AARGB3
               MOVFP
                               0x0D
               MOVLW
                                               ; multiplier a = 1664525
               MOVWF
                               BARGB2
               MOVLW
                               0x66
               MOVWF
                               BARGB1
               MOVLW
                               0x19
               MOVWF
                               BARGB0
                               FXM3224U
               CALL
               MOVLW
                               0x01
                                               ; increment c = 1
               ADDWF
                               AARGB6,F
               CLRF
                               WREG,F
               ADDWFC
                               AARGB5,F
                               AARGB4,F
               ADDWFC
               ADDWFC
                               AARGB3,F
               ADDWFC
                               AARGB2,F
               ADDWFC
                               AARGB1,F
               ADDWFC
                               AARGB0,F
               MOVPF
                               AARGB3,RANDB0
                                              ; modulus m = 2**32
               MOVPF
                               AARGB4, RANDB1
               MOVPF
                               AARGB5, RANDB2
                               AARGB6, RANDB3
               MOVPF
               RETLW
                               0x00
Floating Point Relation A < B
               32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
        Use:
               CALL
                       TALTB32
        Output: logical result in WREG
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
```

# **AN660**

;	Result:			REG = 0x01 $WREG = 0x00$
;	Timing:		max 33	mean 11.6 clks
TALTB32		MOVFP XORWF BTFSC GOTO		AARGB0,WREG BARGB0,W WREG,MSB TALTB320
		BTFSC GOTO		AARGB0,MSB TALTB32N
TALTB32F	,	MOVFP SUBWF BTFSS RETLW BTFSS RETLW		AEXP, WREG BEXP, W _C 0x00 _Z 0x01
		MOVFP SUBWF BTFSS RETLW BTFSS RETLW		AARGB0, WREG BARGB0, W _C 0x00 _Z 0x01
		MOVFP SUBWF BTFSS RETLW BTFSS RETLW		AARGB1,WREG BARGB1,W _C 0x00 _Z 0x01
		MOVFP SUBWF BTFSS RETLW BTFSS RETLW RETLW		AARGB2,WREG BARGB2,W _C 0x00 _Z 0x01 0x00
TALTB32N	Г	MOVFP SUBWF BTFSS RETLW BTFSS RETLW		BEXP, WREG AEXP, W _C 0x00 _Z 0x01
		MOVFP SUBWF BTFSS RETLW BTFSS RETLW		BARGB0, WREG AARGB0, W _C 0x00 _Z 0x01
		MOVFP SUBWF BTFSS RETLW BTFSS RETLW		BARGB1,WREG AARGB1,W _C 0x00 _Z 0x01
		MOVFP SUBWF		BARGB2,WREG AARGB2,W

```
_C
                 BTFSS
                 RETLW
                                   0x00
                 BTFSS
                                   _z
                                   0x01
                 RETLW
                                   0x00
                 RETLW
TALTB320
                 BTFSS
                                   BARGB0, MSB
                 RETLW
                                   0x01
                                   0x00
                 RETLW
        Floating Point Relation A <= B
        Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                 32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
;
        Use:
                 CALL
                          TALEB32
        Output: logical result in WREG
;
;
        Testing on [-MAXNUM, MAXNUM] from 100000 trials:
        Result: if A <= B TRUE, WREG = 0x01
;
                 if A <= B FALSE, WREG = 0x00
                 min
;
                          max
                                   mean
        Timing: 8
                          31
                                   11.6
                                            clks
TALEB32
                 MOVFP
                                   AARGB0, WREG
                 XORWF
                                   BARGB0,W
                                   WREG, MSB
                 BTFSC
                 GOTO
                                   TALEB320
                 BTFSC
                                   AARGB0,MSB
                                   TALEB32N
                 GOTO
TALEB32P
                 MOVFP
                                   AEXP, WREG
                 SUBWF
                                   BEXP,W
                 BTFSS
                                   0x00
                 RETLW
                                   _{\rm Z}
                 BTFSS
                                   0x01
                 RETLW
                 MOVFP
                                   AARGB0, WREG
                 SUBWF
                                   BARGB0,W
                                   _C
                 BTFSS
                 RETLW
                                   0x00
                                   _z
                 BTFSS
                 RETLW
                                   0x01
                 MOVFP
                                   AARGB1,WREG
                                   BARGB1,W
                 SUBWF
                 BTFSS
                                   _C
                 RETLW
                                   0x00
                 BTFSS
                                   _{\rm Z}
                 RETLW
                                   0x01
                                   AARGB2, WREG
                 MOVFP
                                   BARGB2,W
                 SUBWF
                 BTFSS
                                   _C
                                   0x00
                 RETLW
                 RETLW
                                   0x01
```

## **AN660**

```
TALEB32N
                  MOVFP
                                    BEXP, WREG
                                    AEXP,W
                  SUBWF
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _Z
                  RETLW
                                    0x01
                  MOVFP
                                    BARGB0, WREG
                  SUBWF
                                    AARGB0,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _Z
                  RETLW
                                    0x01
                                    BARGB1, WREG
                  MOVFP
                  SUBWF
                                    AARGB1,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                    _{\rm Z}
                  RETLW
                                    0 \times 01
                  MOVFP
                                    BARGB2, WREG
                  SUBWF
                                    AARGB2,W
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  RETLW
                                    0 \times 01
TALEB320
                                    BARGB0,MSB
                  BTFSS
                  RETLW
                                    0x01
                  RETLW
                                    0x00
         Floating Point Relation A > B
         Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
                  32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
         Use:
                  CALL
                          TAGTB32
         Output: logical result in WREG
         Testing on [-MAXNUM, MAXNUM] from 100000 trials:
         Result: if A > B TRUE, WREG = 0x01
                  if A > B FALSE, WREG = 0x00
                  min
                          max
                                    mean
         Timing: 8
                           33
                                    11.6
                                            clks
                                    BARGB0, WREG
TAGTB32
                 MOVFP
                                    AARGB0,W
                  XORWF
                                    WREG, MSB
                  BTFSC
                  GOTO
                                    TAGTB320
                                    BARGBO, MSB
                  BTFSC
                  GOTO
                                    TAGTB32N
TAGTB32P
                  MOVFP
                                    BEXP, WREG
                                    AEXP,W
                  SUBWF
                  BTFSS
                                    _C
                                    0x00
                  RETLW
                  BTFSS
                                    _Z
                  RETLW
                                    0x01
```

```
MOVFP
                                    BARGB0, WREG
                                    AARGB0,W
                  SUBWF
                  BTFSS
                                     _C
                                    0x00
                  RETLW
                  BTFSS
                                     _Z
                                    0x01
                  RETLW
                  MOVFP
                                    BARGB1, WREG
                  SUBWF
                                    AARGB1,W
                                    _C
                  BTFSS
                  RETLW
                                    0x00
                                    _z
                  BTFSS
                                    0x01
                  RETLW
                                    BARGB2, WREG
                  MOVFP
                  SUBWF
                                    AARGB2,W
                  BTFSS
                                     _C
                                    0x00
                  RETLW
                  BTFSS
                                     _{\rm Z}
                                    0 \times 01
                  RETLW
                                    0x00
                  RETLW
TAGTB32N
                  MOVFP
                                    AEXP, WREG
                                    BEXP,W
                  SUBWF
                  BTFSS
                                     _C
                                    0x00
                  RETLW
                  BTFSS
                                     _z
                                    0x01
                  RETLW
                                    AARGB0, WREG
                  MOVFP
                  SUBWF
                                    BARGB0,W
                  BTFSS
                                     _C
                                    0x00
                  RETLW
                  BTFSS
                                     _Z
                                    0x01
                  RETLW
                                    AARGB1,WREG
                  MOVFP
                                    BARGB1,W
                  SUBWF
                  BTFSS
                                     _C
                  RETLW
                                    0x00
                  BTFSS
                                     _{\rm Z}
                                    0x01
                  RETLW
                  MOVFP
                                    AARGB2, WREG
                                    BARGB2,W
                  SUBWF
                  BTFSS
                                    _C
                  RETLW
                                    0x00
                  BTFSS
                                     _Z
                  RETLW
                                    0x01
                  RETLW
                                    0x00
TAGTB320
                                    AARGB0,MSB
                  BTFSS
                                    0 \times 01
                  RETLW
                                    0x00
                  RETLW
         Floating Point Relation A >= B
;
                  32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
;
                  32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
         Use:
                  CALL
                           TAGEB32
         Output: logical result in WREG
```

```
Testing on [-MAXNUM, MAXNUM] from 100000 trials:
         Result: if A >= B TRUE, WREG = 0x01
                  if A >= B FALSE, WREG = 0 \times 00
                  min
                           max
                                     mean
         Timing: 8
                                     11.6
                                              clks
                                    BARGB0, WREG
TAGEB32
                  MOVFP
                  XORWF
                                     AARGB0,W
                  BTFSC
                                     WREG, MSB
                  GOTO
                                     TAGEB320
                                     BARGB0, MSB
                  BTFSC
                  GOTO
                                     TAGEB32N
TAGEB32P
                  MOVFP
                                     BEXP, WREG
                                     AEXP,W
                  SUBWF
                  BTFSS
                                     _C
                                     0x00
                  RETLW
                  BTFSS
                                     _z
                  RETLW
                                     0x01
                  MOVFP
                                     BARGB0, WREG
                  SUBWF
                                     AARGB0,W
                  BTFSS
                                     _C
                                     0x00
                  RETLW
                  BTFSS
                                     _{\rm Z}
                                     0x01
                  RETLW
                                     BARGB1, WREG
                  MOVFP
                                     AARGB1,W
                  SUBWF
                                     _C
                  BTFSS
                                     0x00
                  RETLW
                  BTFSS
                                     _Z
                                     0x01
                  RETLW
                  MOVFP
                                     BARGB2, WREG
                  SUBWF
                                     AARGB2,W
                  BTFSS
                                     _C
                                     0x00
                  RETLW
                  RETLW
                                     0x01
TAGEB32N
                  MOVFP
                                     AEXP, WREG
                  SUBWF
                                     BEXP,W
                  BTFSS
                                     _C
                  RETLW
                                     0x00
                  BTFSS
                                     _Z
                  RETLW
                                     0x01
                                     AARGB0, WREG
                  MOVFP
                  SUBWF
                                     BARGB0,W
                  BTFSS
                                     _C
                  RETLW
                                     0x00
                  BTFSS
                                     _{\rm Z}
                                     0x01
                  RETLW
                  MOVFP
                                     AARGB1, WREG
                  SUBWF
                                     BARGB1,W
                  BTFSS
                                     _C
                  RETLW
                                     0x00
                  BTFSS
                                     _{\rm Z}
                  RETLW
                                     0x01
                                     AARGB2, WREG
                  MOVFP
```

```
SUBWF
                            BARGB2,W
              BTFSS
                            _C
              RETLW
                            0x00
              RETLW
                            0x01
TAGEB320
              BTFSS
                            AARGB0,MSB
              RETLW
              RETLW
                            0x00
********************
       Floating Point Relation A == B
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
              32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
;
       Use:
              CALL
                     TAEQB32
       Output: logical result in WREG
;
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
       Result: if A == B TRUE, WREG = 0 \times 01
              if A == B FALSE, WREG = 0x00
              min
                    max
                            mean
       Timing: 4
                    14
                           5.9
                                  clks
TAEQB32
              MOVFP
                            AEXP, WREG
              CPFSEQ
                            BEXP
              RETLW
                            0x00
                            AARGB0, WREG
              MOVFP
                           BARGB0
              CPESEO
              RETLW
                           0x00
              MOVFP
                            AARGB1,WREG
              CPFSEQ
                            BARGB1
              RETLW
                            0x00
              MOVED
                            AARGB2, WREG
                            BARGB2
              CPFSEQ
              RETLW
                            0x00
              RETLW
                            0x01
   *******************************
       Floating Point Relation A =! B
       Input: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
              32 bit floating point number in BEXP, BARGBO, BARGB1, BARGB2
       Use:
              CALL
                     TANEB32
       Output: logical result in WREG
       Testing on [-MAXNUM, MAXNUM] from 100000 trials:
       Result: if A =! B TRUE, WREG = 0x01
;
              if A =! B FALSE, WREG = 0 \times 00
              min
                     max
                            mean
       Timing: 4
                     14
                            5.9
                                   clks
TANEB32
              MOVFP
                            AEXP, WREG
              CPFSEQ
                            BEXP
                            0x01
              RETLW
```

```
MOVFP
                          AARGB0, WREG
             CPFSEO
                          BARGB0
             RETLW
                          0 \times 01
             MOVFP
                          AARGB1, WREG
             CPFSEO
                          BARGB1
             RETIM
                          0 \times 01
                          AARGB2, WREG
             MOVFP
             CPFSEQ
                          BARGB2
             RETLW
                          0x01
             RETLW
                          0x00
Nearest neighbor rounding
      Input: 40 bit floating point number in AEXP, AARGB0, AARGB1, AARGB2, AARGB3
                   RND4032
      Use:
             CALL
      Output: 32 bit floating point number in AEXP, AARGBO, AARGB1, AARGB2
      Result: AARG <-- RND( AARG )
      Testing on [-MAXNUM, MAXNUM] from 100000 trials:
                   max
                          mean
      Timing: 3
                                 clks
                   23
             min
                   max
                          mean
      Error: 0
                   0
                                 nsb
RND4032
             BTFSS
                          AARGB3,MSB ; is NSB < 0x80?
             RETLW
                          0x00
             BSF
                          _C
                                       ; set carry for rounding
                          0x80
             MOVLW
             CPFSGT
                          AARGB3
             RRCF
                          AARGB2,W
                                       ; select even if NSB = 0x80
             MOVPF
                          AARGB0,SIGN
                                       ; save sign
                          AARGB0,MSB
                                       ; make MSB explicit
             BSF
             CLRF
                          WREG, F
                                       ; round
                          AARGB2,F
             ADDWFC
             ADDWFC
                          AARGB1,F
             ADDWFC
                          AARGB0,F
                                       ; has rounding caused carryout?
             BTFSS
                          _C
                          RND40320K
             GOTO
             RRCF
                          AARGB0,F
                                       ; if so, right shift
             RRCF
                          AARGB1,F
             RRCF
                          AARGB2,F
                          EXP, F
                                       ; test for floating point overflow
             INFSNZ
             GOTO
                          SETFOV32
RND40320K
             BTFSS
                          SIGN, MSB
                                       ; clear sign bit if positive
             BCF
                          AARGB0,MSB
             RETLW
                          0x00
```

#### 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