PIC16C5X / PIC16CXXX Math Utility Routines

Author: *Amar Palacherla*
Microchip Technology Inc.

***PLEASE NOTE:** This application note uses the old Microchip Math Routine format. It is intended for reference purposes only and is being provided for those of you still implementing Binary Coded Decimal(BCD) routines. For any new designs, please refer to application notes contained in Microchip's Embedded Control Handbook Volume II - Math Library.*

INTRODUCTION

This application note provides some utility math routines for Microchip's PIC16C5X and PIC16CXXX series of 8-bit microcontrollers. The following math outlines are provided:

- 8x8 unsigned multiply
- 16x16 double precision multiply
- Fixed Point Division (Table 3)
- 16x16 double precision addition
- 16x16 double precision subtraction
- BCD (Binary Coded Decimal) to binary conversion routines
- Binary to BCD conversion routines
- BCD addition
- BCD subtraction
- Square root

These are written in native assembly language and the listing files are provided. They are also available on a disk (MS-DOS®). All the routines provided can be called as subroutines. Most of the routines have two different versions: one optimized for speed and the other optimized for code size. The calling sequence of each routine is explained at the beginning of each listing file.

SINGLE PRECISION UNSIGNED MULTIPLICATION (8x8)

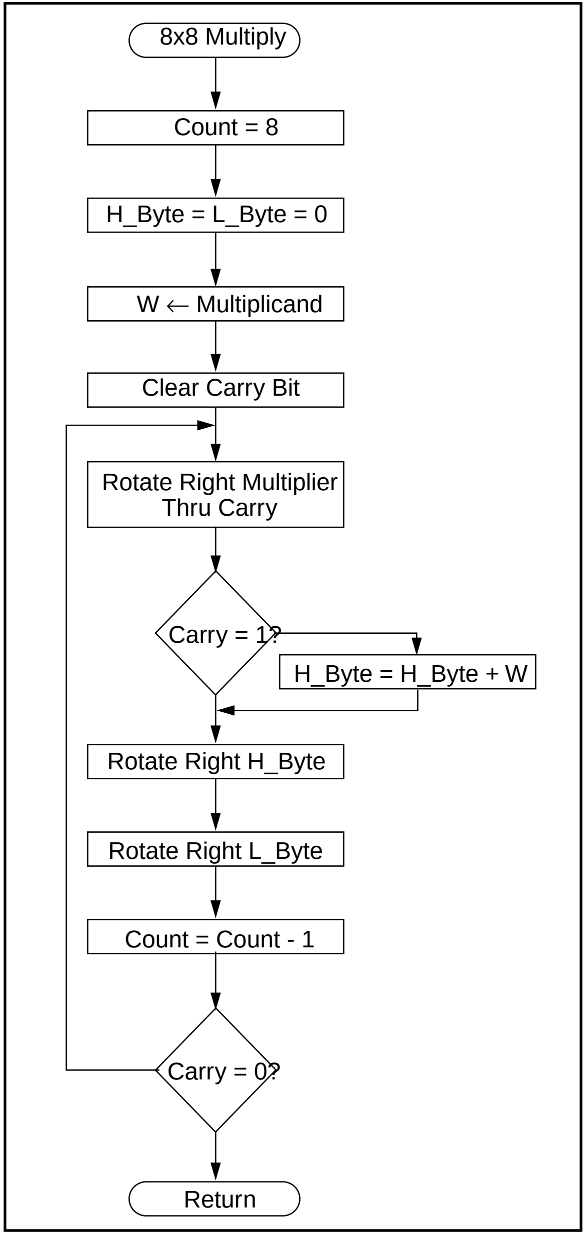
This routine computes the product of two 8-bit unsigned numbers and produces a 16-bit result. Two routines are provided: one routine is optimized for speed (by writing a straight line code) and the other routine has been written to reduce the code size (a looped code). The listing of these routines are given in Appendices A and B. The performance specs for the routines are shown in Table 1.

MS-DOS is a registered trademark of Microsoft Corporation.

TABLE 1: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
Speed Efficient	35	37
Code Efficient	16	71

FIGURE 1: Flowchart for Unsigned 8x8 Multiply



DOUBLE PRECISION MULTIPLY

This routine computes the product of two 16-bit numbers and produces a 32-bit result. Both signed and unsigned arithmetic are supported. Two routines are provided: one routine is optimized for speed (by writing a straight line code) the other routine has been written to reduce code size (a looped code). The listing of these routines are given in Appendices C and D. The performance specs for routines are shown in Table 2.

TABLE 2: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
Speed Efficient	240	233
Code Efficient	33	333

The code in Appendices C and D has been setup for unsigned arithmetic and the performance specs in the table above is for unsigned arithmetic. If signed arithmetic is desired, edit the line with “`SIGNED equ FALSE`” to “`SIGNED equ TRUE`” then re-assemble the code.

In case of signed multiply, both operands are assumed to be 16-bit 2’s complement numbers.

Conditional assembly is supported by MPASM. If you have an older version, please contact the Microchip Technology sales office nearest you.

DOUBLE PRECISION DIVISION

Fixed Point Divide Routines

Fixed point division is fundamentally a conditional shift and subtract operation, resulting in a quotient and a remainder, with standard methods related to simple binary long division by hand calculation. Typically, a processor with n-bit operands uses a fixed accumulator of 2n bits containing the dividend. In standard restoring division, the dividend is left shifted by one bit and the divisor is subtracted from the high half of the accumulator, referred to as the partial remainder. If the result is positive, the divisor was less than or equal to the partial remainder and the corresponding quotient bit in the LSb of the accumulator is set to one. If the result is negative, the divisor was greater than the partial remainder and the dividend is restored by adding back the divisor to the high half of the accumulator and setting the LSb to zero. This process is repeated for each of the n bits in the divisor, resulting in an n-bit quotient in the low half of the accumulator and the n-bit remainder in the high half, and requiring n subtractions and on average n/2 additions [1].

Nonrestoring division, requiring a total of at most n+1 subtractions and additions, offers potential for speed improvement by allowing a negative partial remainder during the calculation with a final addition of the divisor if the final remainder is negative. After the first left shift of the dividend, the divisor is subtracted and the corresponding quotient bit as well as the next add or subtract operation is determined by the carry bit [1].

Unfortunately, no simple method exists for performing two’s complement binary division, thereby requiring negate operations during a preprocessing phase. It is important to note that with the dividend initially loaded into the accumulator, an overflow of the final quotient will result if the high half of the dividend is greater than or equal to the divisor [1], indicating that the n-bit range of the quotient will be exceeded.

Because of the inherent byte structure of the PICmicro™ family of microcontrollers, a more creative and efficient implementation of the above algorithms is possible. In what follows, partial remainder is initialized at zero and is separated from the dividend, thereby avoiding any alignment logic overhead and yielding a quotient with the same number of bits as the dividend and a remainder with the same number as the divisor. Furthermore, routines are named in format FXDxxyyz, where xx is the number of bits in the dividend, yy is the number of bits in the divisor, and z indicates a signed or unsigned routine. Macros are used for core sections of each routine, thereby permitting simple switching between restoring and nonrestoring methods. The signed macros are exclusively a variation of the nonrestoring method, taking advantage of the zero MSb of the arguments after the preprocessing negation. Both restoring and nonrestoring macros are included for the unsigned case, with selection based on best worst case or best average performance as desired. For example, the unsigned macros exhibit the following performance data:

		# of Cycles (TCY)		
		32/16	**16/16**	**16/8**
restore	**max.**	561	240	193
	ave.	481	208	173
nonrestore	**max.**	481	240	190
	ave.	466	233	183

This demonstrates that while the nonrestoring algorithm is preferred for the 32/16 case, the restoring method is preferred for the 16/16 case, with the choice for the 16/8 case a function of user requirements. These optimization complications are a result of trade-offs between the number of instructions required for the restore operations verses the added logic requirements. Finally, additional routines with tacit MSb equal to zero in each argument are included, yielding significant speed improvement. These routines can also be called in the signed case when the arguments are known to be positive for a small benefit.

Routines

It is useful to note that the additional routines FXD3115U, FXD1515U, and FXD1507U can be called in a signed divide application in the special case where AARG > 0 and BARG > 0, thereby offering some improvement in performance.

Data RAM Requirements

The following contiguous data RAM locations are used by the fixed point divide routines:

ACC+B0	=	AARG+B0	AARG and ACC
ACC+B1	=	AARG+B1	
ACC+B2	=	AARG+B2	
ACC+B3	=	AARG+B3	
ACC+B4	=	REM+B0	remainder
ACC+B5	=	REM+B1	
SIGN			sign in MSb
BARG+B0			BARG
BARG+B1			
TEMP+B0			temporary storage
TEMP+B1			

where Bx = x.

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.

TABLE 3: Fixed Point Divide Routines

Routine	Cycles	Function
FXD3216S	414	32-bit/16-bit -> 32.16 signed fixed point divide
FXD3216U	485	32-bit/16-bit -> 32.16 unsigned fixed point divide
FXD3215U	390	32-bit/15-bit -> 32.15 unsigned fixed point divide
FXD3115U	383	31-bit/15-bit -> 31.15 unsigned fixed point divide
FXD1616S	214	16-bit/16-bit -> 16.16 signed fixed point divide
FXD1616U	244	16-bit/16-bit -> 16.16 unsigned fixed point divide
FXD1615U	197	16-bit/15-bit -> 16.15 unsigned fixed point divide
FXD1515U	191	15-bit/15-bit -> 15.15 unsigned fixed point divide
FXD1608S	146	16-bit/08-bit -> 16.08 signed fixed point divide
FXD1608U	196	16-bit/08-bit -> 16.08 unsigned fixed point divide
FXD1607U	130	16-bit/07-bit -> 16.07 unsigned fixed point divide
FXD1507U	125	15-bit/07-bit -> 15.07 unsigned fixed point divide

TABLE 4: PIC16CXXX Fixed Point Divide Performance Data

Routine	Max. Cycles	Min. Cycles	Program Memory	Data Memory
16 / 8 Signed	146	135	146	5
16 / 8 Unsigned	196	156	195	4
16 / 7 Unsigned	130	130	129	4
15 / 7 Unsigned	125	125	124	4
16 / 16 Unsigned	214	187	241	7
16 / 16 Unsigned	244	180	243	6
16 / 15 Unsigned	197	182	216	6
16 / 15 Unsigned	191	177	218	6
32 / 16 Unsigned	414	363	476	9
32 / 16 Unsigned	485	459	608	9
32 / 15 Unsigned	390	359	451	8
31 / 15 Unsigned	383	353	442	8

DOUBLE PRECISION ADDITION & SUBTRACTION

This routine adds or subtracts two 16-bit numbers and produces a 16-bit result. This routine is used by other double precision routines. The listing of these routines is given in Appendix E. The performance specs for the routines are shown below:

TABLE 5: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
Addition	7	8
Subtraction	14	17

BCD TO BINARY CONVERSION

This routine converts a five digit BCD number to a 16-bit binary number. The listing of this routine is given in Appendix F. The performance spec for the routine is shown below:

TABLE 6: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
BCD to Binary	30	121

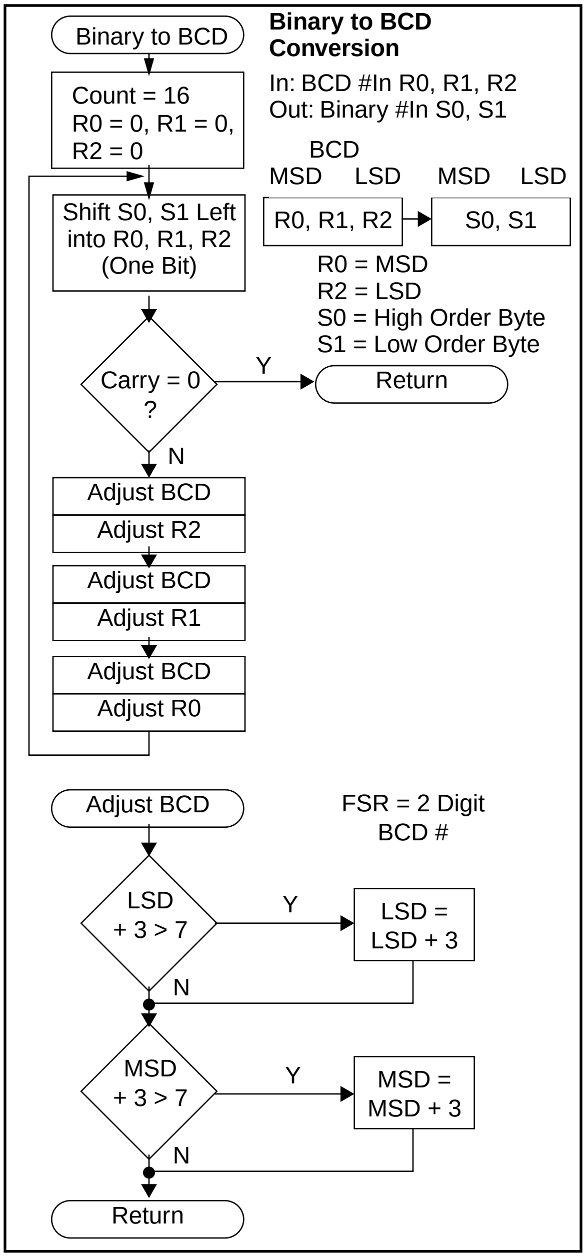
BINARY TO BCD CONVERSION

Two routines are provided: one routine converts a 16-bit binary number to a five-digit BCD number and the other routine converts an 8-bit binary number to a two-digit BCD number. The listing of these routines are given in Appendices G and H. The performance specs for the routines are shown below:

TABLE 7: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
Binary (8-Bit) to BCD	10	81 (Worst Case)
Binary (16-Bit) to BCD	30	719

FIGURE 2: Flowchart for Binary to BCD Conversion



BCD ADDITION & SUBTRACTION

These two routines perform a two-digit unsigned BCD addition and subtraction. The results are the sum (or difference) in one file register and with a overflow carry-bit in another file register. The performance specs for the routines are shown below:

TABLE 8: PERFORMANCE SPECS

Spec	Program Memory	Instruction Cycles
BCD Addition	29	23 (Worst Case)
BCD Subtraction	31	21 (Worst Case)

The listing files for the above two routines are given in Appendices J and K. The flow charts for BCD addition and BCD subtraction are given in Figure 3 and Figure 4, respectively.

FIGURE 3: Flowchart for BCD Addition

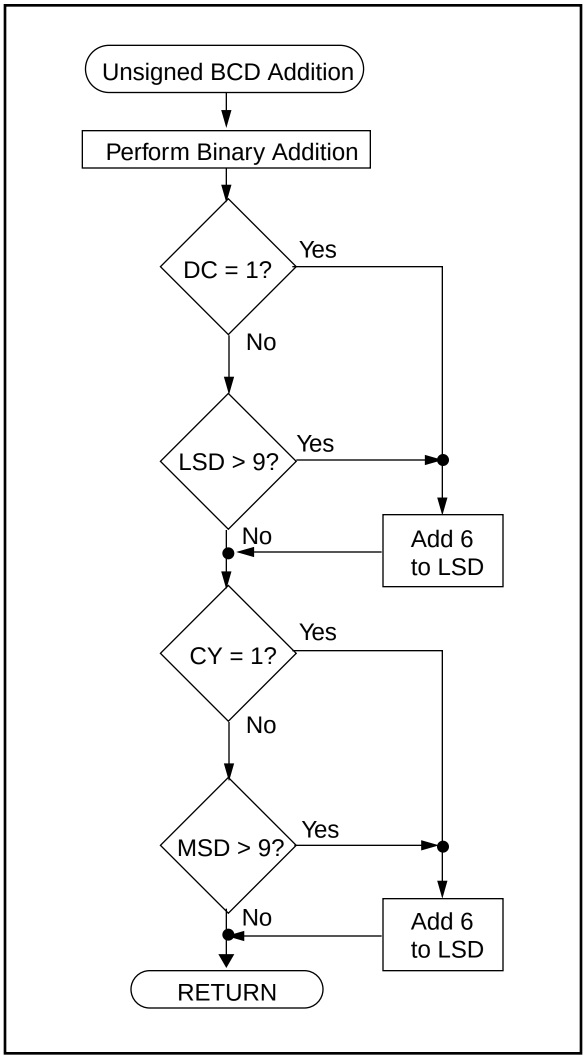
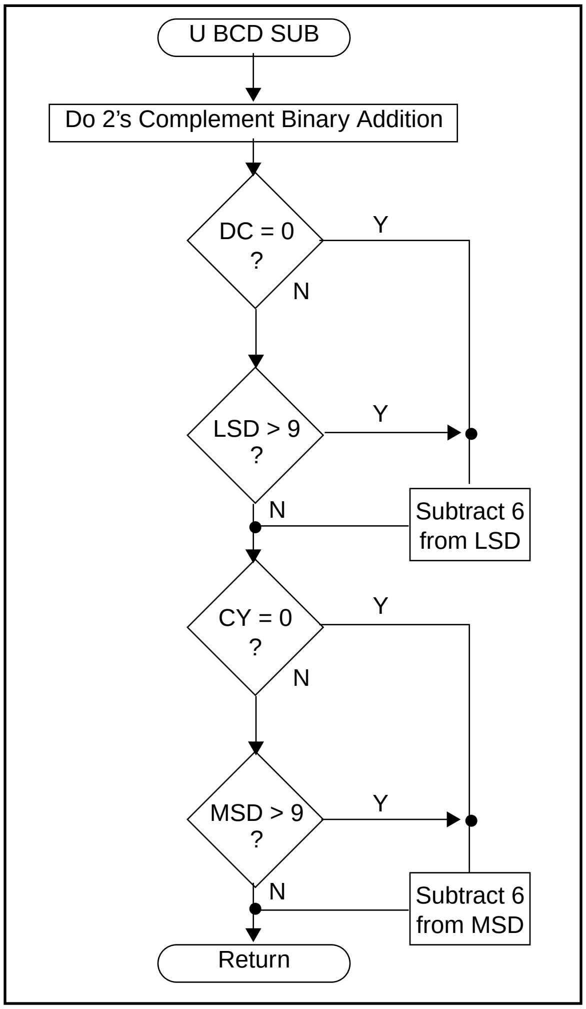


FIGURE 4: Flowchart for BCD Subtraction



SQUARE ROOT

Often in many applications one needs to find the square root of a number. Of many numerical methods to find the square root of a number, the Newton-Raphson method is very attractive because of its fast convergence rate. In this method the square root of a number, "N", is obtained from the approximate solution of:

$f(Y) = Y^2 – N = 0$

The function "f(Y)" can be expanded about Y_0 using first order Taylor polynomial expansion as:

Equation1: f(Y)=f(Y0)+(Y–Y0)f'(Y0)+(Y–Y0)2f"(Y0)/2!+....

If X is a root of f(Y), then f(X) = 0:

f(X) = f(Y0) + (x – Y0) f' (Y0) + (X - Y0) 2f" (Y0) / 2 ! + ... = 0

If Y0 is an approximate root of f(Y), then higher order terms are negligible. Therefore:

Equation 2: $f(Y_0) + (X – Y_0) f' (Y_0)$ [i.e., X = Y0 – f(Y0) / f' (Y0)]

Thus, X is a better approximation for Y0. From this, the sequence {Xn} can be generated:

Equation 3: Xn = Xn – 1 – f(Xn – 1) / f' (Xn – 1), n ≥ 1

From equation 1 and equation 3 we get,

Equation 4: Xn = 0.5* {Xn – 1 + N/Xn – 1}

The initial approximate root of N is taken to be N/2. If the approximate range of N is known a priori, then the total number of iterations may be cut down by starting with a better approximate root than N/2.

This program, as listed in Appendix K, computes the square root of a 16-bit number. This routine uses double precision math routines (division and addition) as described in the previous pages of this application note. The divide routines are integrated into the source listing. For fixed point divide routines, see Appendices L - O.

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX A:

```
MPASM 01.40 Released         MULT8X8S.ASM   1-16-1997  12:54:42         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;*******************************************************************
                    00004 ;                   8x8 Software Multiplier
                    00005 ;               ( Code Efficient : Looped Code )
                    00006 ;*******************************************************************
                    00007 ;
                    00008 ;   The 16 bit result is stored in 2 bytes
                    00009 ;
                    00010 ; Before calling the subroutine “ mpy “, the multiplier should
                    00011 ; be loaded in location “ mulplr “, and the multiplicand in
                    00012 ; “ mulcnd “ . The 16 bit result is stored in locations
                    00013 ; H_byte & L_byte.
                    00014 ;
                    00015 ;       Performance :
                    00016 ;                       Program Memory  :  15 locations
                    00017 ;                       # of cycles     :  71
                    00018 ;                       Scratch RAM     :   0 locations
                    00019 ;
                    00020 ;
                    00021 ;       Program:          MULT8x8S.ASM
                    00022 ;       Revision Date:
                    00023 ;                         1-13-97    Compatibility with MPASMWIN 1.40
                    00024 ;
                    00025 ; This routine is optimized for code efficiency (looped code)
                    00026 ; For time efficiency code refer to “mult8x8F.asm”(straight line code)
                    00027 ;*******************************************************************
                    00028 ;
  00000009          00029 mulcnd  equ     09      ; 8 bit multiplicand
  00000010          00030 mulplr  equ     10      ; 8 bit multiplier
  00000012          00031 H_byte  equ     12      ; High byte of the 16 bit result
  00000013          00032 L_byte  equ     13      ; Low byte of the 16 bit result
  00000014          00033 count   equ     14      ; loop counter
                    00034 ;
                    00035 ;
                    00036         include         “p16c5x.inc”
                    00001         LIST
                    00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                    00224         LIST
                    00037
  00000001          00038 Same    equ     1
                    00039
                    00040 ;
                    00041 ; *****************************         Begin Multiplier Routine
0000 0072           00042 mpy_S   clrf    H_byte
0001 0073           00043         clrf    L_byte
0002 0C08           00044         movlw   8
0003 0034           00045         movwf   count
0004 0209           00046         movf    mulcnd,W
0005 0403           00047         bcf     STATUS,C    ; Clear the carry bit in the status Reg.
0006 0330           00048 loop    rrf     mulplr, F
0007 0603           00049         btfsc   STATUS,C
0008 01F2           00050         addwf   H_byte,Same
0009 0332           00051         rrf     H_byte,Same
```

```
000A 0333           00052         rrf     L_byte,Same
000B 02F4           00053         decfsz  count, F
000C 0A06           00054         goto    loop
                    00055 ;
000D 0800           00056         retlw   0
                    00057 ;
                    00058 ;********************************************************************
                    00059 ;               Test Program
                    00060 ;*********************************************************************
000E 0CFF           00061 main    movlw   0FF
000F 0030           00062         movwf   mulplr       ; multiplier (in mulplr) = 0FF
0010 0CFF           00063         movlw   0FF          ; multiplicand(W Reg )   = 0FF
0011 0029           00064         movwf   mulcnd
                    00065 ;
0012 0900           00066         call    mpy_S        ; The result 0FF*0FF = FE01 is in locations
                    00067 ;                            ; H_byte & L_byte
                    00068 ;
0013 0A13           00069 self    goto    self
                    00070 ;
01FF                00071         org     01FF
01FF 0A0E           00072         goto    main
                    00073 ;
                    00074         END
MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXX------------ ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    21
Program Memory Words Free:   491

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX B:

```
MPASM 01.40 Released         MULT8X8F.ASM   1-16-1997  12:54:10         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;********************************************************************
                    00004 ;                   8x8 Software Multiplier
                    00005 ;               ( Fast Version : Straight Line Code )
                    00006 ;********************************************************************
                    00007 ;
                    00008 ;   The 16 bit result is stored in 2 bytes
                    00009 ;
                    00010 ; Before calling the subroutine “ mpy “, the multiplier should
                    00011 ; be loaded in location “ mulplr “, and the multiplicand in
                    00012 ; “ mulcnd “ . The 16 bit result is stored in locations
                    00013 ; H_byte & L_byte.
                    00014 ;
                    00015 ;       Performance :
                    00016 ;                       Program Memory  :  35 locations
                    00017 ;                       # of cycles     :  37
                    00018 ;                       Scratch RAM     :   0 locations
                    00019 ;
                    00020 ;
                    00021 ;       Program:          MULT8x8F.ASM
                    00022 ;       Revision Date:
                    00023 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                    00024 ;
                    00025 ; This routine is optimized for speed efficiency (straight line code)
                    00026 ; For code efficiency, refer to “mult8x8S.asm” (looped code)
                    00027 ;********************************************************************
                    00028 ;
  00000009          00029 mulcnd  equ     09      ; 8 bit multiplicand
  00000010          00030 mulplr  equ     10      ; 8 bit multiplier
  00000012          00031 H_byte  equ     12      ; High byte of the 16 bit result
  00000013          00032 L_byte  equ     13      ; Low byte of the 16 bit result
                    00033 ;
                    00034 ;
                    00035         include         “p16c5x.inc”
                    00001         LIST
                    00002 ; P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                    00224         LIST
                    00036
  00000001          00037 Same    equ     1
                    00038
                    00039 ;
                    00040 ;****   Define a macro for adding & right shifting  **
                    00041 ;
                    00042 mult    MACRO   bit          ; Begin macro
                    00043         btfsc   mulplr,bit
                    00044         addwf   H_byte,Same
                    00045         rrf     H_byte,Same
                    00046         rrf     L_byte,Same
                    00047         ENDM                 ; End of macro
                    00048 ;
                    00049 ; *****************************  Begin Multiplier Routine
0000 0072           00050 mpy_F clrf    H_byte
0001 0073           00051       clrf    L_byte
```

```
0002 0209           00052       movf    mulcnd,W   ; move the multiplicand to W reg.
0003 0403           00053       bcf     STATUS,C   ; Clear the carry bit in the status Reg.
                    00054       mult    0
0004 0610               M       btfsc   mulplr,0
0005 01F2               M       addwf   H_byte,Same
0006 0332               M       rrf     H_byte,Same
0007 0333               M       rrf     L_byte,Same
                    00055       mult    1
0008 0630               M       btfsc   mulplr,1
0009 01F2               M       addwf   H_byte,Same
000A 0332               M       rrf     H_byte,Same
000B 0333               M       rrf     L_byte,Same
                    00056       mult    2
000C 0650               M       btfsc   mulplr,2
000D 01F2               M       addwf   H_byte,Same
000E 0332               M       rrf     H_byte,Same
000F 0333               M       rrf     L_byte,Same
                    00057       mult    3
0010 0670               M       btfsc   mulplr,3
0011 01F2               M       addwf   H_byte,Same
0012 0332               M       rrf     H_byte,Same
0013 0333               M       rrf     L_byte,Same
                    00058       mult    4
0014 0690               M       btfsc   mulplr,4
0015 01F2               M       addwf   H_byte,Same
0016 0332               M       rrf     H_byte,Same
0017 0333               M       rrf     L_byte,Same
                    00059       mult    5
0018 06B0               M       btfsc   mulplr,5
0019 01F2               M       addwf   H_byte,Same
001A 0332               M       rrf     H_byte,Same
001B 0333               M       rrf     L_byte,Same
                    00060       mult    6
001C 06D0               M       btfsc   mulplr,6
001D 01F2               M       addwf   H_byte,Same
001E 0332               M       rrf     H_byte,Same
001F 0333               M       rrf     L_byte,Same
                    00061       mult    7
0020 06F0               M       btfsc   mulplr,7
0021 01F2               M       addwf   H_byte,Same
0022 0332               M       rrf     H_byte,Same
0023 0333               M       rrf     L_byte,Same
                    00062 ;
0024 0800           00063      retlw   0
                    00064 ;
                    00065 ;********************************************************************
                    00066 ;               Test Program
                    00067 ;*********************************************************************
0025 0CFF           00068 main  movlw   0FF
0026 0030           00069       movwf   mulplr   ; multiplier (in mulplr)  = 0FF
0027 0CFF           00070       movlw   0FF
0028 0029           00071       movwf   mulcnd   ; multiplicand(in mulcnd) = 0FF
                    00072 ;
0029 0900           00073       call    mpy_F    ; The result 0FF*0FF = FE01 is in locations
                    00074 ;                      ; H_byte & L_byte
                    00075 ;
002A 0A2A           00076 self goto    self
                    00077 ;
01FF                00078      org     01FF
01FF 0A25           00079      goto    main
                    00080 ;
                    00081         END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXX----- ----------------
```

```
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    44
Program Memory Words Free:   468


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip’s Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX C:DOUBLE PRECISION MULTIPLICATION LISTING (LOOPED)

```
MPASM 01.40 Released         DBL_MPYS.ASM   1-16-1997  12:53:00         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54,  n = 66
                    00002 ;
                    00003 ;*******************************************************************
                    00004 ;                    Double Precision Multiplication
                    00005 ;
                    00006 ;               ( Optimized for Code Size : Looped Code )
                    00007 ;
                    00008 ;*******************************************************************;
                    00009 ; Multiplication: ACCb(16 bits)*ACCa(16 bits) -> ACCb,ACCc (32 bits)
                    00010 ;  (a) Load the 1st operand in location ACCaLO & ACCaHI (16 bits)
                    00011 ;  (b) Load the 2nd operand in location ACCbLO & ACCbHI (16 bits)
                    00012 ;  (c) CALL D_mpy
                    00013 ;  (d) The 32 bit result is in location (ACCbHI,ACCbLO,ACCcHI,ACCcLO)
                    00014 ;
                    00015 ;   Performance :
                    00016 ;               Program Memory  :   033
                    00017 ;               Clock Cycles    :   333
                    00018 ;
                    00019 ;    Note : The above timing is the worst case timing, when the
                    00020 ;           register ACCb = FFFF. The speed may be improved if
                    00021 ;           the register ACCb contains a number ( out of the two
                    00022 ;           numbers ) with less number of 1s.
                    00023 ;           The performance specs are for Unsigned arithmetic (i.e,
                    00024 ;           with “SIGNED equ  FALSE“).
                    00025 ;
                    00026 ;           The performance specs are for Unsigned arithmetic (i.e,
                    00027 ;           with “SIGNED equ  FALSE“).
                    00028 ;
                    00029 ;
                    00030 ;       Program:        DBL_MPYS.ASM
                    00031 ;       Revision Date:
                    00032 ;                       1-13-97   Compatibility with MPASMWIN 1.40
                    00033 ;
                    00034 ;*******************************************************************;
                    00035 ;
  00000010          00036 ACCaLO  equ     0x10
  00000011          00037 ACCaHI  equ     0x11
  00000012          00038 ACCbLO  equ     0x12
  00000013          00039 ACCbHI  equ     0x13
  00000014          00040 ACCcLO  equ     0x14
  00000015          00041 ACCcHI  equ     0x15
  00000016          00042 ACCdLO  equ     0x16
  00000017          00043 ACCdHI  equ     0x17
  00000018          00044 temp    equ     0x18
  00000019          00045 sign    equ     0x19
  0000001F          00046 Flags   equ     0x1F
                    00047 ;
                    00048         include “p16c5x.inc”
                    00001         LIST
                    00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                    00224         LIST
                    00049
  000001FF          00050 PIC54   equ     1FFH    ; Define Reset Vector
  00000001          00051 TRUE    equ     1
  00000000          00052 FALSE   equ     0
  00000007          00053 MSB     equ     7
                    00054
0000                00055         org     0
                    00056 ;*******************************************************************
  00000001          00057 SIGNED  equ     TRUE            ; Set This To ‘TRUE’ if the routines
                    00058 ;                               ; for Multiplication & Division needs
                    00059 ;                               ; to be assembled as Signed Integer
                    00060 ;                               ; Routines. If ‘FALSE’ the above two
                    00061 ;                               ; routines ( D_mpy & D_div ) use
                    00062 ;                               ; unsigned arithmetic.
                    00063 ;*******************************************************************
                    00064 ;         Double Precision Addition ( ACCb + ACCa -> ACCb )
                    00065 ;
0000 041F           00066 D_add   bcf     Flags,C ;Clear temp Carry bit
0001 0210           00067         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0002 01F2           00068         addwf   ACCbLO, F       ;add lsb
0003 0603           00069         btfsc   STATUS,C        ;add in carry
0004 02B3           00070         incf    ACCbHI, F
0005 0603           00071         btfsc   STATUS,C
0006 051F           00072         bsf     Flags,C
0007 0211           00073         movf    ACCaHI,W
0008 01F3           00074         addwf   ACCbHI, F         ;add msb
0009 061F           00075         btfsc   Flags,C
000A 0503           00076         bsf     STATUS,C
000B 0800           00077         retlw   0
                    00078 ;*******************************************************************
                    00079 ;               Double Precision Multiply ( 16x16 -> 32 )
                    00080 ;         ( ACCb*ACCa -> ACCb,ACCc ) : 32 bit output with high word
                    00081 ;  in ACCb ( ACCbHI,ACCbLO ) and low word in ACCc ( ACCcHI,ACCcLO ).
                    00082 ;
000C                00083 D_mpyS                          ;results in ACCb(16 msb’s) and ACCc(16
lsb’s)
                    00084 ;
                    00085      IF   SIGNED
000C 0935           00086      CALL    S_SIGN
                    00087      ENDIF
                    00088 ;
000D 0926           00089         call    setup
000E 0337           00090 mloop   rrf     ACCdHI, F       ;rotate d right
000F 0336           00091         rrf     ACCdLO, F
0010 0603           00092         btfsc   STATUS,C        ;need to add?
0011 0900           00093         call    D_add
0012 0333           00094         rrf     ACCbHI, F
0013 0332           00095         rrf     ACCbLO, F
0014 0335           00096         rrf     ACCcHI, F
0015 0334           00097         rrf     ACCcLO, F
0016 02F8           00098         decfsz  temp, F         ;loop until all bits checked
0017 0A0E           00099         goto    mloop
                    00100 ;
                    00101    IF    SIGNED
0018 07F9           00102         btfss   sign,MSB
0019 0800           00103         retlw   0
001A 0274           00104         comf    ACCcLO, F       ; negate ACCa ( -ACCa -> ACCa )
001B 02B4           00105         incf    ACCcLO, F
001C 0643           00106         btfsc   STATUS,Z
001D 00F5           00107         decf    ACCcHI, F
001E 0275           00108         comf    ACCcHI, F
001F 0643           00109         btfsc   STATUS,Z
0020 0272           00110 neg_B   comf    ACCbLO, F       ; negate ACCb
0021 02B2           00111         incf    ACCbLO, F
0022 0643           00112         btfsc   STATUS,Z
0023 00F3           00113         decf    ACCbHI, F
0024 0273           00114         comf    ACCbHI, F
0025 0800           00115         retlw   0
                    00116    ELSE
```

```
                            retlw   0
                        ENDIF
                    ;
                    ;*******************************************************************
                    ;
0026 0C10           setup   movlw   .16             ; for 16 shifts
0027 0038                   movwf   temp
0028 0213                   movf    ACCbHI,W        ; move ACCb to ACCd
0029 0037                   movwf   ACCdHI
002A 0212                   movf    ACCbLO,W
002B 0036                   movwf   ACCdLO
002C 0073                   clrf    ACCbHI
002D 0072                   clrf    ACCbLO
002E 0800                   retlw   0
                    ;
                    ;*******************************************************************
                    ;
002F 0270           neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
0030 02B0                   incf    ACCaLO, F
0031 0643                   btfsc   STATUS,Z
0032 00F1                   decf    ACCaHI, F
0033 0271                   comf    ACCaHI, F
0034 0800                   retlw   0
                    ;
                    ;*******************************************************************
                    ;  Assemble this section only if Signed Arithmetic Needed
                    ;
                         IF    SIGNED
                    ;
0035 0211           S_SIGN  movf    ACCaHI,W
0036 0193                   xorwf   ACCbHI,W
0037 0039                   movwf   sign
0038 07F3                   btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
0039 0A3F                   goto    chek_A
                    ;
003A 0272                   comf    ACCbLO, F        ; negate ACCb
003B 02B2                   incf    ACCbLO, F
003C 0643                   btfsc   STATUS,Z
003D 00F3                   decf    ACCbHI, F
003E 0273                   comf    ACCbHI, F
                    ;
003F 07F1           chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
0040 0800                   retlw   0
0041 0A2F                   goto    neg_A
                    ;
                         ENDIF
                    ;
                    ;*******************************************************************
                    ;                       Test Program
                    ;*******************************************************************
                    ;    Load constant values to ACCa & ACCb for testing
                    ;
0042 0C01           main    movlw   1
0043 0031                   movwf   ACCaHI
0044 0CFF                   movlw   0FF             ; loads ACCa = 01FF
0045 0030                   movwf   ACCaLO
                    ;
0046 0C7F                   movlw   0x7F
0047 0033                   movwf   ACCbHI
0048 0CFF                   movlw   0xFF            ; loads ACCb = 7FFF
0049 0032                   movwf   ACCbLO
                    ;
004A 090C                   call    D_mpyS          ; Here (ACCb,ACCc) = 00FF 7E01
                    ;
004B 0A4B           self    goto    self
                    ;
```

```
01FF                00183         org     PIC54
01FF 0A42           00184         goto    main
                    00185         END

MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)


0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXX---- ---------------- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    77
Program Memory Words Free:   435


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX D:DOUBLE PRECISION MULTIPLICATION LISTINGS (FAST)

```
MPASM 01.40 Released         DBL_MPYF.ASM   1-16-1997  12:52:26         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;*******************************************************************
                00004 ;                       Double Precision Multiplication
                00005 ;
                00006 ;               ( Optimized for Speed : straight Line Code )
                00007 ;
                00008 ;*******************************************************************;
                00009 ;Multiplication : ACCb(16 bits) * ACCa(16 bits) -> ACCb,ACCc (32 bits)
                00010 ;  (a) Load the 1st operand in location ACCaLO & ACCaHI (16 bits)
                00011 ;  (b) Load the 2nd operand in location ACCbLO & ACCbHI (16 bits)
                00012 ;  (c) CALL D_mpy
                00013 ;  (d) The 32 bit result is in location (ACCbHI,ACCbLO,ACCcHI,ACCcLO)
                00014 ;
                00015 ;   Performance :
                00016 ;               Program Memory  :       240
                00017 ;               Clock Cycles    :       233
                00018 ;
                00019 ;   Note : The above timing is the worst case timing, when the
                00020 ;          register ACCb = FFFF. The speed may be improved if
                00021 ;          the register ACCb contains a number (out of the two
                00022 ;          numbers) with less number of 1s.
                00023 ;
                00024 ;          The performance specs are for Unsigned arithmetic (i.e,
                00025 ;          with “SIGNED equ  FALSE“).
                00026 ;
                00027 ;       Program:        DBL_MPYF.ASM
                00028 ;       Revision Date:
                00029 ;                       1-13-97  Compatibility with MPASMWIN 1.40
                00030 ;
                00031 ;*******************************************************************;
                00032 ;
  00000010      00033 ACCaLO  equ     10
  00000011      00034 ACCaHI  equ     11
  00000012      00035 ACCbLO  equ     12
  00000013      00036 ACCbHI  equ     13
  00000014      00037 ACCcLO  equ     14
  00000015      00038 ACCcHI  equ     15
  00000016      00039 ACCdLO  equ     16
  00000017      00040 ACCdHI  equ     17
  00000018      00041 temp    equ     18
  00000019      00042 sign    equ     19
                00043 ;
                00044         include “p16c5x.inc”
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                00224         LIST
                00045
  000001FF      00046 PIC54   equ     1FFH    ; Define Reset Vector
  00000001      00047 TRUE    equ     1
  00000000      00048 FALSE   equ     0
                00049
0000            00050         org     0
                00051 ;*******************************************************************
  00000000      00052 SIGNED  equ     FALSE           ; Set This To ‘TRUE’ if the routines
                00053 ;                               ; for Multiplication & Division needs
                00054 ;                               ; to be assembled as Signed Integer
                00055 ;                               ; Routines. If ‘FALSE’ the above two
                00056 ;                               ; routines ( D_mpy & D_div ) use
                00057 ;                               ; unsigned arithmetic.
                00058 ;*******************************************************************
                00059 ;       multiplication macro
                00060 ;
                00061 mulMac  MACRO
                00062         LOCAL   NO_ADD
                00063 ;
                00064         rrf     ACCdHI, F       ;rotate d right
                00065         rrf     ACCdLO, F
                00066         btfss   STATUS,C        ; need to add?
                00067         goto    NO_ADD          ; no addition necessary
                00068         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
                00069         addwf   ACCbLO, F       ;add lsb
                00070         btfsc   STATUS,C        ; add in carry
                00071         incf    ACCbHI, F
                00072         movf    ACCaHI,W
                00073         addwf   ACCbHI, F       ;add msb
                00074 NO_ADD  rrf     ACCbHI, F
                00075         rrf     ACCbLO, F
                00076         rrf     ACCcHI, F
                00077         rrf     ACCcLO, F
                00078 ;
                00079         ENDM
                00080 ;
                00081 ;*******************************************************************;
                00082 ;               Double Precision Multiply ( 16x16 -> 32 )
                00083 ;         ( ACCb*ACCa -> ACCb,ACCc ) : 32 bit output with high word
                00084 ;  in ACCb ( ACCbHI,ACCbLO ) and low word in ACCc ( ACCcHI,ACCcLO ).
                00085 ;
0000            00086 D_mpyF              ;results in ACCb(16 msb’s) and ACCc(16 lsb’s)
                00087 ;
                00088      IF   SIGNED
                00089      CALL    S_SIGN
                00090      ENDIF
                00091 ;
0000 09E2       00092         call    setup
                00093 ;
                00094 ; use the mulMac macro 16 times
                00095 ;
                00096         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0001 0337           M         rrf     ACCdHI, F       ;rotate d right
0002 0336           M         rrf     ACCdLO, F
0003 0703           M         btfss   STATUS,C        ; need to add?
0004 0A0B           M         goto    NO_ADD          ; no addition necessary
0005 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0006 01F2           M         addwf   ACCbLO, F       ; add lsb
0007 0603           M         btfsc   STATUS,C        ; add in carry
0008 02B3           M         incf    ACCbHI, F
0009 0211           M         movf    ACCaHI,W
000A 01F3           M         addwf   ACCbHI, F       ; add msb
000B 0333           M NO_ADD  rrf     ACCbHI, F
000C 0332           M         rrf     ACCbLO, F
000D 0335           M         rrf     ACCcHI, F
000E 0334           M         rrf     ACCcLO, F
                    M ;
                00097         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
000F 0337           M         rrf     ACCdHI, F       ; rotate d right
```

```
0010 0336           M         rrf     ACCdLO, F
0011 0703           M         btfss   STATUS,C        ; need to add?
0012 0A19           M         goto    NO_ADD          ; no addition necessary
0013 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0014 01F2           M         addwf   ACCbLO, F       ;add lsb
0015 0603           M         btfsc   STATUS,C        ; add in carry
0016 02B3           M         incf    ACCbHI, F
0017 0211           M         movf    ACCaHI,W
0018 01F3           M         addwf   ACCbHI, F       ; add msb
0019 0333           M NO_ADD  rrf     ACCbHI, F
001A 0332           M         rrf     ACCbLO, F
001B 0335           M         rrf     ACCcHI, F
001C 0334           M         rrf     ACCcLO, F
                    M ;
                00098         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
001D 0337           M         rrf     ACCdHI, F       ; rotate d right
001E 0336           M         rrf     ACCdLO, F
001F 0703           M         btfss   STATUS,C        ; need to add?
0020 0A27           M         goto    NO_ADD          ; no addition necessary
0021 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0022 01F2           M         addwf   ACCbLO, F       ; add lsb
0023 0603           M         btfsc   STATUS,C        ; add in carry
0024 02B3           M         incf    ACCbHI, F
0025 0211           M         movf    ACCaHI,W
0026 01F3           M         addwf   ACCbHI, F       ; add msb
0027 0333           M NO_ADD  rrf     ACCbHI, F
0028 0332           M         rrf     ACCbLO, F
0029 0335           M         rrf     ACCcHI, F
002A 0334           M         rrf     ACCcLO, F
                    M ;
                00099         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
002B 0337           M         rrf     ACCdHI, F       ; rotate d right
002C 0336           M         rrf     ACCdLO, F
002D 0703           M         btfss   STATUS,C        ; need to add?
002E 0A35           M         goto    NO_ADD          ; no addition necessary
002F 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0030 01F2           M         addwf   ACCbLO, F       ; add lsb
0031 0603           M         btfsc   STATUS,C        ; add in carry
0032 02B3           M         incf    ACCbHI, F
0033 0211           M         movf    ACCaHI,W
0034 01F3           M         addwf   ACCbHI, F       ; add msb
0035 0333           M NO_ADD  rrf     ACCbHI, F
0036 0332           M         rrf     ACCbLO, F
0037 0335           M         rrf     ACCcHI, F
0038 0334           M         rrf     ACCcLO, F
                    M ;
                00100         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0039 0337           M         rrf     ACCdHI, F       ; rotate d right
003A 0336           M         rrf     ACCdLO, F
003B 0703           M         btfss   STATUS,C        ; need to add?
003C 0A43           M         goto    NO_ADD          ; no addition necessary
003D 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
003E 01F2           M         addwf   ACCbLO, F       ; add lsb
003F 0603           M         btfsc   STATUS,C        ; add in carry
0040 02B3           M         incf    ACCbHI, F
0041 0211           M         movf    ACCaHI,W
0042 01F3           M         addwf   ACCbHI, F       ; add msb
0043 0333           M NO_ADD  rrf     ACCbHI, F
0044 0332           M         rrf     ACCbLO, F
0045 0335           M         rrf     ACCcHI, F
```

```
0046 0334           M         rrf     ACCcLO, F
                    M ;
                00101         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0047 0337           M         rrf     ACCdHI, F       ;rotate d right
0048 0336           M         rrf     ACCdLO, F
0049 0703           M         btfss   STATUS,C        ; need to add?
004A 0A51           M         goto    NO_ADD          ; no addition necessary
004B 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
004C 01F2           M         addwf   ACCbLO, F       ; add lsb
004D 0603           M         btfsc   STATUS,C        ; add in carry
004E 02B3           M         incf    ACCbHI, F
004F 0211           M         movf    ACCaHI,W
0050 01F3           M         addwf   ACCbHI, F       ; add msb
0051 0333           M NO_ADD  rrf     ACCbHI, F
0052 0332           M         rrf     ACCbLO, F
0053 0335           M         rrf     ACCcHI, F
0054 0334           M         rrf     ACCcLO, F
                    M ;
                00102         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0055 0337           M         rrf     ACCdHI, F       ; rotate d right
0056 0336           M         rrf     ACCdLO, F
0057 0703           M         btfss   STATUS,C        ; need to add?
0058 0A5F           M         goto    NO_ADD          ; no addition necessary
0059 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
005A 01F2           M         addwf   ACCbLO, F       ; add lsb
005B 0603           M         btfsc   STATUS,C        ; add in carry
005C 02B3           M         incf    ACCbHI, F
005D 0211           M         movf    ACCaHI,W
005E 01F3           M         addwf   ACCbHI, F       ; add msb
005F 0333           M NO_ADD  rrf     ACCbHI, F
0060 0332           M         rrf     ACCbLO, F
0061 0335           M         rrf     ACCcHI, F
0062 0334           M         rrf     ACCcLO, F
                    M ;
                00103         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0063 0337           M         rrf     ACCdHI, F       ; rotate d right
0064 0336           M         rrf     ACCdLO, F
0065 0703           M         btfss   STATUS,C        ; need to add?
0066 0A6D           M         goto    NO_ADD          ; no addition necessary
0067 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0068 01F2           M         addwf   ACCbLO, F       ; add lsb
0069 0603           M         btfsc   STATUS,C        ; add in carry
006A 02B3           M         incf    ACCbHI, F
006B 0211           M         movf    ACCaHI,W
006C 01F3           M         addwf   ACCbHI, F       ; add msb
006D 0333           M NO_ADD  rrf     ACCbHI, F
006E 0332           M         rrf     ACCbLO, F
006F 0335           M         rrf     ACCcHI, F
0070 0334           M         rrf     ACCcLO, F
                    M ;
                00104         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
0071 0337           M         rrf     ACCdHI, F       ; rotate d right
0072 0336           M         rrf     ACCdLO, F
0073 0703           M         btfss   STATUS,C        ; need to add?
0074 0A7B           M         goto    NO_ADD          ; no addition necessary
0075 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0076 01F2           M         addwf   ACCbLO, F       ; add lsb
0077 0603           M         btfsc   STATUS,C        ; add in carry
```

```
0078 02B3           M       incf    ACCbHI, F
0079 0211           M       movf    ACCaHI,W
007A 01F3           M       addwf   ACCbHI, F       ; add msb
007B 0333           M NO_ADD  rrf     ACCbHI, F
007C 0332           M       rrf     ACCbLO, F
007D 0335           M       rrf     ACCcHI, F
007E 0334           M       rrf     ACCcLO, F
                    M ;
                00105       mulMac
  0000              M       LOCAL   NO_ADD
                    M ;
007F 0337           M       rrf     ACCdHI, F       ; rotate d right
0080 0336           M       rrf     ACCdLO, F
0081 0703           M       btfss   STATUS,C        ; need to add?
0082 0A89           M       goto    NO_ADD          ; no addition necessary
0083 0210           M       movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0084 01F2           M       addwf   ACCbLO, F       ; add lsb
0085 0603           M       btfsc   STATUS,C        ;  add in carry
0086 02B3           M       incf    ACCbHI, F
0087 0211           M       movf    ACCaHI,W
0088 01F3           M       addwf   ACCbHI, F       ; add msb
0089 0333           M NO_ADD  rrf     ACCbHI, F
008A 0332           M       rrf     ACCbLO, F
008B 0335           M       rrf     ACCcHI, F
008C 0334           M       rrf     ACCcLO, F
                    M ;
                00106       mulMac
  0000              M       LOCAL   NO_ADD
                    M ;
008D 0337           M       rrf     ACCdHI, F       ; rotate d right
008E 0336           M       rrf     ACCdLO, F
008F 0703           M       btfss   STATUS,C        ; need to add?
0090 0A97           M       goto    NO_ADD          ; no addition necessary
0091 0210           M       movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
0092 01F2           M       addwf   ACCbLO, F       ; add lsb
0093 0603           M       btfsc   STATUS,C        ; add in carry
0094 02B3           M       incf    ACCbHI, F
0095 0211           M       movf    ACCaHI,W
0096 01F3           M       addwf   ACCbHI, F       ; add msb
0097 0333           M NO_ADD  rrf     ACCbHI, F
0098 0332           M       rrf     ACCbLO, F
0099 0335           M       rrf     ACCcHI, F
009A 0334           M       rrf     ACCcLO, F
                    M ;
                00107       mulMac
  0000              M       LOCAL   NO_ADD
                    M ;
009B 0337           M       rrf     ACCdHI, F       ; rotate d right
009C 0336           M       rrf     ACCdLO, F
009D 0703           M       btfss   STATUS,C        ; need to add?
009E 0AA5           M       goto    NO_ADD          ; no addition necessary
009F 0210           M       movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
00A0 01F2           M       addwf   ACCbLO, F       ; add lsb
00A1 0603           M       btfsc   STATUS,C        ; add in carry
00A2 02B3           M       incf    ACCbHI, F
00A3 0211           M       movf    ACCaHI,W
00A4 01F3           M       addwf   ACCbHI, F       ; add msb
00A5 0333           M NO_ADD  rrf     ACCbHI, F
00A6 0332           M       rrf     ACCbLO, F
00A7 0335           M       rrf     ACCcHI, F
00A8 0334           M       rrf     ACCcLO, F
                    M ;
                00108       mulMac
  0000              M       LOCAL   NO_ADD
                    M ;
00A9 0337           M       rrf     ACCdHI, F       ; rotate d right
```

```
00AA 0336           M         rrf     ACCdLO, F
00AB 0703           M         btfss   STATUS,C        ; need to add?
00AC 0AB3           M         goto    NO_ADD          ; no addition necessary
00AD 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
00AE 01F2           M         addwf   ACCbLO, F       ; add lsb
00AF 0603           M         btfsc   STATUS,C        ; add in carry
00B0 02B3           M         incf    ACCbHI, F
00B1 0211           M         movf    ACCaHI,W
00B2 01F3           M         addwf   ACCbHI, F       ; add msb
00B3 0333           M NO_ADD  rrf     ACCbHI, F
00B4 0332           M         rrf     ACCbLO, F
00B5 0335           M         rrf     ACCcHI, F
00B6 0334           M         rrf     ACCcLO, F
                    M ;
                00109         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
00B7 0337           M         rrf     ACCdHI, F       ; rotate d right
00B8 0336           M         rrf     ACCdLO, F
00B9 0703           M         btfss   STATUS,C        ; need to add?
00BA 0AC1           M         goto    NO_ADD          ; no addition necessary
00BB 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
00BC 01F2           M         addwf   ACCbLO, F       ; add lsb
00BD 0603           M         btfsc   STATUS,C        ; add in carry
00BE 02B3           M         incf    ACCbHI, F
00BF 0211           M         movf    ACCaHI,W
00C0 01F3           M         addwf   ACCbHI, F       ; add msb
00C1 0333           M NO_ADD  rrf     ACCbHI, F
00C2 0332           M         rrf     ACCbLO, F
00C3 0335           M         rrf     ACCcHI, F
00C4 0334           M         rrf     ACCcLO, F
                    M ;
                00110         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
00C5 0337           M         rrf     ACCdHI, F       ; rotate d right
00C6 0336           M         rrf     ACCdLO, F
00C7 0703           M         btfss   STATUS,C        ; need to add?
00C8 0ACF           M         goto    NO_ADD          ; no addition necessary
00C9 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
00CA 01F2           M         addwf   ACCbLO, F       ; add lsb
00CB 0603           M         btfsc   STATUS,C        ; add in carry
00CC 02B3           M         incf    ACCbHI, F
00CD 0211           M         movf    ACCaHI,W
00CE 01F3           M         addwf   ACCbHI, F       ; add msb
00CF 0333           M NO_ADD  rrf     ACCbHI, F
00D0 0332           M         rrf     ACCbLO, F
00D1 0335           M         rrf     ACCcHI, F
00D2 0334           M         rrf     ACCcLO, F
                    M ;
                00111         mulMac
  0000              M         LOCAL   NO_ADD
                    M ;
00D3 0337           M         rrf     ACCdHI, F       ; rotate d right
00D4 0336           M         rrf     ACCdLO, F
00D5 0703           M         btfss   STATUS,C        ; need to add?
00D6 0ADD           M         goto    NO_ADD          ; no addition necessary
00D7 0210           M         movf    ACCaLO,W        ; Addition ( ACCb + ACCa -> ACCb )
00D8 01F2           M         addwf   ACCbLO, F       ; add lsb
00D9 0603           M         btfsc   STATUS,C        ; add in carry
00DA 02B3           M         incf    ACCbHI, F
00DB 0211           M         movf    ACCaHI,W
00DC 01F3           M         addwf   ACCbHI, F       ; add msb
00DD 0333           M NO_ADD  rrf     ACCbHI, F
00DE 0332           M         rrf     ACCbLO, F
00DF 0335           M         rrf     ACCcHI, F
```

```
00E0 0334           M         rrf     ACCcLO, F
                    M ;
                00112 ;
                00113     IF    SIGNED
                00114         btfss   sign,MSB
                00115         retlw   0
                00116         comf    ACCcLO          ; negate ACCa ( -ACCa -> ACCa )
                00117         incf    ACCcLO
                00118         btfsc   STATUS,Z
                00119         decf    ACCcHI
                00120         comf    ACCcHI
                00121         btfsc   STATUS,Z
                00122 neg_B   comf    ACCbLO          ; negate ACCb
                00123         incf    ACCbLO
                00124         btfsc   STATUS,Z
                00125         decf    ACCbHI
                00126         comf    ACCbHI
                00127         retlw   0
                00128     ELSE
00E1 0800       00129         retlw   0
                00130     ENDIF
                00131 ;
                00132 ;*******************************************************************
                00133 ;
00E2 0C10       00134 setup   movlw   .16             ; for 16 shifts
00E3 0038       00135         movwf   temp
00E4 0213       00136         movf    ACCbHI,W          ;move ACCb to ACCd
00E5 0037       00137         movwf   ACCdHI
00E6 0212       00138         movf    ACCbLO,W
00E7 0036       00139         movwf   ACCdLO
00E8 0073       00140         clrf    ACCbHI
00E9 0072       00141         clrf    ACCbLO
00EA 0800       00142         retlw   0
                00143 ;
                00144 ;*******************************************************************
                00145 ;
00EB 0270       00146 neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
00EC 02B0       00147         incf    ACCaLO, F
00ED 0643       00148         btfsc   STATUS,Z
00EE 00F1       00149         decf    ACCaHI, F
00EF 0271       00150         comf    ACCaHI, F
00F0 0800       00151         retlw   0
                00152 ;
                00153 ;*******************************************************************
                00154 ;  Assemble this section only if Signed Arithmetic Needed
                00155 ;
                00156      IF    SIGNED
                00157 ;
                00158 S_SIGN  movf    ACCaHI,W
                00159         xorwf   ACCbHI,W
                00160         movwf   sign
                00161         btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
                00162         goto    chek_A
                00163 ;
                00164         comf    ACCbLO           ; negate ACCb
                00165         incf    ACCbLO
                00166         btfsc   STATUS,Z
                00167         decf    ACCbHI
                00168         comf    ACCbHI
                00169 ;
                00170 chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
                00171         retlw   0
                00172         goto    neg_A
                00173 ;
                00174      ENDIF
                00175 ;
                00176 ;*******************************************************************
                00177 ;                       Test Program
                00178 ;*******************************************************************
                00179 ;    Load constant values to ACCa & ACCb for testing
                00180 ;
00F1 0C01       00181 loadAB  movlw   1
00F2 0031       00182         movwf   ACCaHI
00F3 0CFF       00183         movlw   0FF             ; loads ACCa = 01FF
00F4 0030       00184         movwf   ACCaLO
                00185 ;
00F5 0C7F       00186         movlw   07F
00F6 0033       00187         movwf   ACCbHI
00F7 0CFF       00188         movlw   0FF             ; loads ACCb = 7FFF
00F8 0032       00189         movwf   ACCbLO
00F9 0800       00190         retlw   0
                00191 ;
00FA 0000       00192 main    nop
                00193 ;
00FB 09F1       00194         call    loadAB ;result of multiplying ACCb*ACCa->(ACCb,ACCc)
00FC 0900       00195         call    D_mpyF          ; Here (ACCb,ACCc) = 00FF 7E01
                00196 ;
00FD 0AFD       00197 self    goto    self
                00198 ;
01FF            00199         org     PIC54
01FF 0AFA       00200         goto    main
                00201         END
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)


0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXX--
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   255
Program Memory Words Free:   257


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX E:DOUBLE PRECISION ADDITION AND SUBTRACTION LISTING

```
MPASM 01.40 Released          DBL_ADD.ASM   1-16-1997  12:50:38         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

               00001         LIST    P = 16C54, n = 66
               00002 ;
               00003 ;*******************************************************************
               00004 ;                 Double Precision Addition & Subtraction
               00005 ;
               00006 ;*******************************************************************;
               00007 ;   Addition :  ACCb(16 bits) + ACCa(16 bits) -> ACCb(16 bits)
               00008 ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
               00009 ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
               00010 ;      (c) CALL D_add
               00011 ;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
               00012 ;
               00013 ;   Performance :
               00014 ;               Program Memory  :  07
               00015 ;               Clock Cycles    :  08
               00016 ;*******************************************************************;
               00017 ;   Subtraction : ACCb(16 bits) - ACCa(16 bits) -> ACCb(16 bits)
               00018 ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
               00019 ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
               00020 ;      (c) CALL D_sub
               00021 ;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
               00022 ;
               00023 ;   Performance :
               00024 ;               Program Memory  :       14
               00025 ;               Clock Cycles    :       17
               00026 ;
               00027 ;
               00028 ;       Program:          DBL_ADD.ASM
               00029 ;       Revision Date:
               00030 ;                         1-13-97      Compatibility with MPASMWIN 1.40
               00031 ;
               00032 ;*******************************************************************;
               00033 ;
  00000010     00034 ACCaLO  equ     10
  00000011     00035 ACCaHI  equ     11
  00000012     00036 ACCbLO  equ     12
  00000013     00037 ACCbHI  equ     13
               00038 ;
               00039         include "p16c5x.inc"
               00001         LIST
               00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
               00224         LIST
               00040
  000001FF     00041 PIC54   equ     1FFH    ; Define Reset Vector
               00042
0000           00043         org     0
               00044 ;*******************************************************************
               00045 ;         Double Precision Subtraction ( ACCb - ACCa -> ACCb )
               00046 ;
0000 0908      00047 D_sub   call    neg_A           ; At first negate ACCa; Then add
               00048 ;
               00049 ;*******************************************************************
               00050 ;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
               00051 ;
0001 0210      00052 D_add   movf    ACCaLO,W
0002 01F2      00053         addwf   ACCbLO, F       ; add lsb
```

```
0003 0603      00054         btfsc   STATUS,C        ; add in carry
0004 02B3      00055         incf    ACCbHI, F
0005 0211      00056         movf    ACCaHI,W
0006 01F3      00057         addwf   ACCbHI, F       ; add msb
0007 0800      00058         retlw   0
               00059 ;
               00060 ;
0008 0270      00061 neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
0009 02B0      00062         incf    ACCaLO, F
000A 0643      00063         btfsc   STATUS,Z
000B 00F1      00064         decf    ACCaHI, F
000C 0271      00065         comf    ACCaHI, F
000D 0800      00066         retlw   0
               00067 ;
               00068 ;*******************************************************************
               00069 ;                       Test Program
               00070 ;*******************************************************************
               00071 ;    Load constant values to ACCa & ACCb for testing
               00072 ;
000E 0C01      00073 loadAB  movlw   1
000F 0031      00074         movwf   ACCaHI
0010 0CFF      00075         movlw   0FF             ; loads ACCa = 01FF
0011 0030      00076         movwf   ACCaLO
               00077 ;
0012 0C7F      00078         movlw   07F
0013 0033      00079         movwf   ACCbHI
0014 0CFF      00080         movlw   0FF             ; loads ACCb = 7FFF
0015 0032      00081         movwf   ACCbLO
0016 0800      00082         retlw   0
               00083 ;
0017 0000      00084 main    nop
               00085 ;
0018 090E      00086         call    loadAB          ; result of adding ACCb+ACCa->ACCb
0019 0901      00087         call    D_add           ; Here Accb = 81FE
               00088 ;
001A 090E      00089         call    loadAB     ; result of subtracting ACCb - ACCa->ACCb
001B 0900      00090         call    D_sub           ; Here Accb = 7E00
               00091 ;
001C 0A1C      00092 self    goto    self
               00093 ;
01FF           00094         org     PIC54
01FF 0A17      00095         goto    main
               00096         END

MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXX--- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    30
Program Memory Words Free:   482



Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip’s Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX F:BCD TO BINARY CONVERSION LISTING

```
MPASM 01.40 Released          BCD2BIN.ASM   1-16-1997  12:49:30         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P = 16C54, n = 66
                    00002 ;
                    00003 ;**********************************************************************
                    00004 ;               BCD To Binary Conversion
                    00005 ;
                    00006 ;       This routine converts a 5 digit BCD number to a 16 bit binary
                    00007 ; number.
                    00008 ;       The input 5 digit BCD numbers are asumed to be in locations
                    00009 ; R0, R1 & R2 with R0 containing the MSD in its right most nibble.
                    00010 ;
                    00011 ;       The 16 bit binary number is output in registers H_byte & L_byte
                    00012 ; ( high byte & low byte repectively ).
                    00013 ;
                    00014 ;       The method used for conversion is :
                    00015 ;               input number X = abcde ( the 5 digit BCD number )
                    00016 ;               X = abcde = 10[10[10[10a+b]+c]+d]+e
                    00017 ;
                    00018 ;   Performance :
                    00019 ;               Program Memory  :       30
                    00020 ;               Clock Cycles    :       121
                    00021 ;
                    00022 ;
                    00023 ;       Program:          BCD2BIN.ASM
                    00024 ;       Revision Date:
                    00025 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                    00026 ;
                    00027 ;*******************************************************************;
                    00028 ;
  00000010          00029 H_byte  equ     10
  00000011          00030 L_byte  equ     11
  00000012          00031 R0      equ     12              ; RAM Assignments
  00000013          00032 R1      equ     13
  00000014          00033 R2      equ     14
                    00034 ;
  00000015          00035 H_temp  equ     15      ; temporary register
  00000016          00036 L_temp  equ     16      ; temporary register
                    00037 ;
                    00038 ;
                    00039         INCLUDE         “p16c5x.inc”
                    00001         LIST
                    00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                    00224         LIST
                    00040 ;
                    00041 ;
0000 0E0F           00042 mpy10b  andlw   0F
0001 01F1           00043         addwf   L_byte, F
0002 0603           00044         btfsc   STATUS,C
0003 02B0           00045         incf    H_byte, F
0004 0403           00046 mpy10a  bcf     STATUS,C        ; multiply by 2
0005 0351           00047         rlf     L_byte,W
0006 0036           00048         movwf   L_temp
0007 0350           00049         rlf     H_byte,W        ; (H_temp,L_temp) = 2*N
0008 0035           00050         movwf   H_temp
                    00051 ;
```

```
0009 0403           00052         bcf     STATUS,C        ; multiply by 2
000A 0371           00053         rlf     L_byte, F
000B 0370           00054         rlf     H_byte, F
000C 0403           00055         bcf     STATUS,C        ; multiply by 2
000D 0371           00056         rlf     L_byte, F
000E 0370           00057         rlf     H_byte, F
000F 0403           00058         bcf     STATUS,C        ; multiply by 2
0010 0371           00059         rlf     L_byte, F
0011 0370           00060         rlf     H_byte, F       ; (H_byte,L_byte) = 8*N
                    00061 ;
0012 0216           00062         movf    L_temp,W
0013 01F1           00063         addwf   L_byte, F
0014 0603           00064         btfsc   STATUS,C
0015 02B0           00065         incf    H_byte, F
0016 0215           00066         movf    H_temp,W
0017 01F0           00067         addwf   H_byte, F
0018 0800           00068         retlw   0               ; (H_byte,L_byte) = 10*N
                    00069 ;
                    00070 ;
0019 0070           00071 BCDtoB  clrf    H_byte
001A 0212           00072         movf    R0,W
001B 0E0F           00073         andlw   0F
001C 0031           00074         movwf   L_byte
001D 0904           00075         call    mpy10a          ; result = 10a+b
                    00076 ;
001E 0393           00077         swapf   R1,W
001F 0900           00078         call    mpy10b          ; result = 10[10a+b]
                    00079 ;
0020 0213           00080         movf    R1,W
0021 0900           00081         call    mpy10b          ; result = 10[10[10a+b]+c]
                    00082 ;
0022 0394           00083         swapf   R2,W
0023 0900           00084         call    mpy10b          ; result = 10[10[10[10a+b]+c]+d]
                    00085 ;
0024 0214           00086         movf    R2,W
0025 0E0F           00087         andlw   0F
0026 01F1           00088         addwf   L_byte, F
0027 0603           00089         btfsc   STATUS,C
0028 02B0           00090         incf    H_byte, F       ; result = 10[10[10[10a+b]+c]+d]+e
0029 0800           00091         retlw   0               ; BCD to binary conversion done
                    00092 ;
                    00093 ;
                    00094 ;********************************************************************
                    00095 ;               Test Program
                    00096 ;*********************************************************************
002A 0C06           00097 main    movlw   06
002B 0032           00098         movwf   R0      ; Set R0 = 06
002C 0C55           00099         movlw   55
002D 0033           00100         movwf   R1      ; Set R1 = 55
002E 0C35           00101         movlw   35
002F 0034           00102         movwf   R2      ; Set R2 = 35      ( R0, R1, R2 = 6,55,35 )
                    00103 ;
0030 0919           00104         call    BCDtoB  ; After conversion H_Byte = FF & L_Byte = FF
                    00105 ;
0031 0A31           00106 self    goto    self
                    00107 ;
01FF                00108         org     1FF
01FF 0A2A           00109         goto    main
                    00110 ;
                    00111         END
MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XX--------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.
```

```
Program Memory Words Used:    51
Program Memory Words Free:   461

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX G:BINARY (8-BIT) TO BCD CONVERSION

```
MPASM 01.40 Released          BIN8BCD.ASM   1-16-1997  12:50:05         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;********************************************************************
                00004 ;                  Binary To BCD Conversion Routine
                00005 ;
                00006 ;       This routine converts the 8 bit binary number in the W Register
                00007 ; to a 2 digit BCD number.
                00008 ;       The least significant digit is returned in location LSD and
                00009 ; the most significant digit is returned in location MSD.
                00010 ;
                00011 ;   Performance :
                00012 ;               Program Memory  :       10
                00013 ;               Clock Cycles    :       81 (worst case when W = 63 Hex )
                00014 ;                                          ( i.e max Decimal number 99 )
                00015 ;
                00016 ;       Program:          BIN8BCD.ASM
                00017 ;       Revision Date:
                00018 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                00019 ;
                00020 ;*******************************************************************
                00021 ;
  00000010      00022 LSD     equ     10
  00000011      00023 MSD     equ     11
                00024 ;
                00025         INCLUDE         “p16c5x.inc”
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                00224         LIST
                00026 ;
0000 0071       00027 BinBCD  clrf    MSD
0001 0030       00028         movwf   LSD
0002 0C0A       00029 gtenth  movlw   .10
0003 0090       00030         subwf   LSD,W
0004 0703       00031         BTFSS   STATUS,C
0005 0A09       00032         goto    over
0006 0030       00033         movwf   LSD
0007 02B1       00034         incf    MSD, F
0008 0A02       00035         goto    gtenth
0009 0800       00036 over    retlw   0
                00037 ;*******************************************************************
                00038 ;
000A 0C63       00039 main    movlw   63              ; W reg = 63 Hex
000B 0900       00040         call    BinBCD          ; after conversion, MSD = 9 & LSD = 9
000C 0A0C       00041 self    goto    self            ; ( 63 Hex = 99 Decimal )
                00042 ;
01FF            00043         org     1FF
01FF 0A0A       00044         goto    main
                00045 ;
                00046         END
0000 : XXXXXXXXXXXXX--- ---------------- ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.
```

```
Program Memory Words Used:    14
Program Memory Words Free:   498


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX H:BINARY (16-BIT) TO BCD LISTING

```
MPASM 01.40 Released         B16TOBCD.ASM   1-16-1997  12:48:00         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;********************************************************************
                00004 ;                  Binary To BCD Conversion Routine
                00005 ;      This routine converts a 16 Bit binary Number to a 5 Digit
                00006 ; BCD Number. This routine is useful since PIC16C55 & PIC16C57
                00007 ; have  two 8 bit ports and one 4 bit port ( total of 5 BCD digits)
                00008 ;
                00009 ;       The 16 bit binary number is input in locations H_byte and
                00010 ; L_byte with the high byte in H_byte.
                00011 ;       The 5 digit BCD number is returned in R0, R1 and R2 with R0
                00012 ; containing the MSD in its right most nibble.
                00013 ;
                00014 ;   Performance :
                00015 ;               Program Memory  :       35
                00016 ;               Clock Cycles    :       885
                00017 ;
                00018 ;
                00019 ;       Program:          B16TOBCD.ASM
                00020 ;       Revision Date:
                00021 ;                         1-13-97    Compatibility with MPASMWIN 1.40
                00022 ;
                00023 ;*******************************************************************;
                00024 ;
  00000016      00025 count  equ      16
  00000017      00026 temp   equ      17
                00027 ;
  00000010      00028 H_byte  equ     10
  00000011      00029 L_byte  equ     11
  00000012      00030 R0      equ     12              ; RAM Assignments
  00000013      00031 R1      equ     13
  00000014      00032 R2      equ     14
                00033 ;
                00034         include         “p16c5x.inc”
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                00224         LIST
                00035 ;
0000 0403       00036 B2_BCD  bcf     STATUS,0                ; clear the carry bit
0001 0C10       00037         movlw   .16
0002 0036       00038         movwf   count
0003 0072       00039         clrf    R0
0004 0073       00040         clrf    R1
0005 0074       00041         clrf    R2
0006 0371       00042 loop16  rlf     L_byte, F
0007 0370       00043         rlf     H_byte, F
0008 0374       00044         rlf     R2, F
0009 0373       00045         rlf     R1, F
000A 0372       00046         rlf     R0, F
                00047 ;
000B 02F6       00048         decfsz  count, F
000C 0A0E       00049         goto    adjDEC
000D 0800       00050         RETLW   0
                00051 ;
```

```
000E 0C14       00052 adjDEC  movlw   R2
000F 0024       00053         movwf   FSR
0010 0918       00054         call    adjBCD
                00055 ;
0011 0C13       00056         movlw   R1
0012 0024       00057         movwf   FSR
0013 0918       00058         call    adjBCD
                00059 ;
0014 0C12       00060         movlw   R0
0015 0024       00061         movwf   FSR
0016 0918       00062         call    adjBCD
                00063 ;
0017 0A06       00064         goto    loop16
                00065 ;
0018 0C03       00066 adjBCD  movlw   3
0019 01C0       00067         addwf   0,W
001A 0037       00068         movwf   temp
001B 0677       00069         btfsc   temp,3          ; test if result > 7
001C 0020       00070         movwf   0
001D 0C30       00071         movlw   30
001E 01C0       00072         addwf   0,W
001F 0037       00073         movwf   temp
0020 06F7       00074         btfsc   temp,7          ; test if result > 7
0021 0020       00075         movwf   0               ; save as MSD
0022 0800       00076         RETLW   0
                00077 ;
                00078 ;********************************************************************
                00079 ;               Test Program
                00080 ;*********************************************************************
0023 0CFF       00081 main    movlw   0FF
0024 0030       00082         movwf   H_byte
0025 0031       00083         movwf   L_byte          ; The 16 bit binary number = FFFF
0026 0900       00084         call    B2_BCD          ; After conversion the Decimal Number
                00085 ;                               ; in R0,R1,R2 = 06,55,35
                00086 ;
0027 0A27       00087 self    goto    self
                00088 ;
01FF            00089         org     1FF
01FF 0A23       00090         goto    main
                00091 ;
                00092         END

MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXX-------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    41
Program Memory Words Free:   471

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX I:UNSIGNED BCD SUBTRACTION LISTING

```
MPASM 01.40 Released          BCD_SUB.ASM   1-16-1997  12:49:00         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;*******************  Unsigned BCD Subtraction   ***************
                00004 ;
                00005 ;       This routine performs a 2 Digit Unsigned BCD Subtraction.
                00006 ; It is assumed that the two BCD numbers to be subtracted are in
                00007 ; locations Num_1 & Num_2. The result is the difference of Num_1 & Num_2
                00008 ; ( Num_2 - Num_1) and is stored in location Num_2 and the overflow carry
                00009 ; is returned in location Num_1.
                00010 ;
                00011 ;   Performance :
                00012 ;               Program Memory  :       31
                00013 ;               Clock Cycles    :       21  ( worst case )
                00014 ;
                00015 ;
                00016 ;       Program:          BCD_SUB.ASM
                00017 ;       Revision Date:
                00018 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                00019 ;
                00020 ;*******************************************************************
                00021 ;
  00000008      00022 Num_1   equ     8       ; Overflow flow carry overwrites Num_1
  00000008      00023 result  equ     8
                00024 ;
  00000009      00025 Num_2   equ     9       ; Num_2 - Num_1 overwrites Num_2
  00000009      00026 O_flow  equ     9
                00027 ;
                00028         include         “p16c5x.inc”
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                00224         LIST
                00029 ;
0000 0208       00030 BCDSub  movf    Num_1,W
0001 00A9       00031         subwf   Num_2, F
0002 0068       00032         clrf    Num_1
0003 0368       00033         rlf     Num_1, F
0004 0723       00034         btfss   STATUS,DC
0005 0A0C       00035         goto    adjst1
0006 0769       00036         btfss   Num_2,3         ; Adjust LSD of Result
0007 0A0E       00037         goto    Over_1
0008 0649       00038         btfsc   Num_2,2
0009 0A0C       00039         goto    adjst1          ; Adjust LSD of Result
000A 0729       00040         btfss   Num_2,1
000B 0A0E       00041         goto    Over_1          ; No : Go for MSD
000C 0C06       00042 adjst1  movlw   6
000D 00A9       00043         subwf   Num_2, F
000E 0708       00044 Over_1  btfss   Num_1,0         ; CY = 0 ?
000F 0A17       00045         goto    adjst2          ; Yes, adjust MSD of result
0010 0068       00046         clrf    Num_1
0011 07E9       00047         btfss   Num_2,7         ; No, test for MSD >9
0012 0800       00048         RETLW   0
0013 06C9       00049         btfsc   Num_2,6
0014 0A17       00050         goto    adjst2
0015 07A9       00051         btfss   Num_2,5
```

```
0016 0800       00052         RETLW   0
0017 0C60       00053 adjst2  movlw   60              ; add 6 to MSD
0018 00A9       00054         subwf   Num_2, F
0019 0068       00055         clrf    Num_1
001A 0703       00056         btfss   STATUS,C        ; test if underflow
001B 0800       00057         RETLW   0
001C 0C01       00058         movlw   1
001D 0028       00059         movwf   Num_1
001E 0800       00060 Over    RETLW   0
                00061 ;
                00062 ;********************************************************************
                00063 ;               Test Program
                00064 ;*********************************************************************
001F 0C23       00065 main    movlw   23
0020 0028       00066         movwf   Num_1      ; Set Num_1 = 23
0021 0C99       00067         movlw   99
0022 0029       00068         movwf   Num_2      ; Set Num_2 = 99
0023 0900       00069         call    BCDSub     ; After subtraction, Num_2 = 76 ( 99-23 )
                00070 ;                          ;  and Num_1 = 0 ( indicates positive result )
                00071 ;
0024 0C99       00072         movlw   99
0025 0028       00073         movwf   Num_1      ; Set Num_1 = 99
0026 0C00       00074         movlw   0
0027 0029       00075         movwf   Num_2      ; Set Num_2 = 0
                00076 ;
0028 0900       00077         call    BCDSub     ; After subtraction, Num_2 = 1
                00078 ;                          ;  and Num_1 = 1 ( indicates negative result )
                00079 ;                          ;  -1  <-  ( -99 )
                00080 ;
0029 0A29       00081 self    goto    self
                00082 ;
01FF            00083         org     1FF
01FF 0A1F       00084         goto    main
                00085 ;
                00086         END
MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXX------ ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    43
Program Memory Words Free:   469

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

APPENDIX J:SQUARE ROOT METHOD

```
MPASM 01.40 Released             SQRT.ASM   1-16-1997  12:55:13         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;*******************************************************************
                00004 ;
                00005 ;            Square Root By Newton Raphson Method
                00006 ;
                00007 ;  This routine computes the square root of a 16 bit number(with
                00008 ;  low byte in NumLo & high byte in NumHi ). After loading NumLo &
                00009 ;  NumHi with the desired number whose square root is to be computed,
                00010 ;  branch to location Sqrt ( by “GOTO  Sqrt” ). “ CALL  Sqrt” cannot
                00011 ;  be issued because the Sqrt function makes calls to Math routines
                00012 ;  and the stack is completely used up.
                00013 ;  The result = sqrt(NumHi,NumLo) is returned in location SqrtLo.
                00014 ;  The total number of iterations is set to ten. If more iterations
                00015 ;  are desired, change “LupCnt equ .10” to the desired value. Also,
                00016 ;  the initial guess value of the square root is given set as
                00017 ;  input/2 ( in subroutine “init” ). The user may modify this scheme
                00018 ;  if a better initial approximation value is known. A good initial
                00019 ;  guess will help the algorithm converge at a faster rate and thus
                00020 ;  less number of iterations required.
                00021 ;  Two utility math routines are used by this program : D_divS
                00022 ;  and D_add. These two routines are listed as seperate routines
                00023 ;  under double precision Division and double precision addtion
                00024 ;  respectively.
                00025 ;
                00026 ;  Note : If square root of an 8 bit number is desired, it is probably
                00027 ;         better to have a table look scheme rather than using numerical
                00028 ;         methods.
                00029 ;
                00030 ;
                00031 ;
                00032 ;   Performance :
                00033 ;               Program Memory  :       27  (excluding Math Routines
                00034 ;                                            D_divS & D_add )
                00035 ;               Clock Cycles    :       3600 ( approximately )
                00036 ;
                00037 ;
                00038 ;       Program:          SQRT.ASM
                00039 ;       Revision Date:
                00040 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                00041 ;
                00042 ;       To assemble this program, two routines, namely “D_add” &
                00043 ;       “D_divS” must be included into this program. These two routines
                00044 ;       are listed as separate programs in files “DBL_ADD.ASM” &
                00045 ;       “DBL_DIVS.ASM” respectively.
                00046 ;
                00047 ;*******************************************************************
                00048         include “p16c5x.inc”
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver.n 3.30 Microchip Technology, Inc.
                00224         LIST
                00049
  000001FF      00050 PIC54   equ     1FFH    ; Define Reset Vector
  00000001      00051 TRUE    equ     1
  00000000      00052 FALSE   equ     0
                00053
0000            00054         org     0
                00055 ;
  0000000A      00056 LupCnt  equ     .10             ; Number of iterations
                00057 ;
  00000010      00058 ACCaLO  equ     10
  00000011      00059 ACCaHI  equ     11
  00000013      00060 ACCbLO  equ     13
  00000014      00061 ACCbHI  equ     14
  00000014      00062 ACCcLO  equ     14
  00000015      00063 ACCcHI  equ     15
  00000016      00064 ACCdLO  equ     16
  00000017      00065 ACCdHI  equ     17
  00000018      00066 temp    equ     18
  00000019      00067 sign    equ     19
                00068 ;
  00000010      00069 SqrtLo  equ     ACCaLO
  00000011      00070 SqrtHi  equ     ACCaHI
                00071 ;
  0000001D      00072 NumLo   equ     1D
  0000001E      00073 NumHi   equ     1E
  0000001F      00074 count   equ     1F
                00075 ;
                00076 ;
0000            00077 init
0000 0C0A       00078         movlw   LupCnt
0001 003F       00079         movwf   count
0002 021E       00080         movf    NumHi,W
0003 0031       00081         movwf   SqrtHi
0004 021D       00082         movf    NumLo,W         ; set initial guess root = NUM/2
0005 0030       00083         movwf   SqrtLo
0006 0403       00084         bcf     STATUS,C
0007 0331       00085         rrf     SqrtHi, F
0008 0330       00086         rrf     SqrtLo, F
0009 0800       00087         retlw   0
                00088 ;
000A 0403       00089 div2    bcf     STATUS,C
000B 0314       00090         rrf     ACCbHI,W
000C 0031       00091         movwf   SqrtHi
000D 0313       00092         rrf     ACCbLO,W
000E 0030       00093         movwf   SqrtLo
000F 0800       00094         retlw   0
                00095 ;
                00096 ;*******************************************************************
                00097 ;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
                00098 ;
0010 0210       00099 D_add   movf    ACCaLO,W
0011 01F3       00100         addwf   ACCbLO, F       ;add lsb
0012 0603       00101         btfsc   STATUS,C    ;add in carry
0013 02B4       00102         incf    ACCbHI, F
0014 0211       00103         movf    ACCaHI,W
0015 01F4       00104         addwf   ACCbHI, F       ;add msb
0016 0800       00105         retlw   0
                00106 ;
                00107 ;*******************************************************************
  00000000      00108 SIGNED  equ     FALSE           ; Set This To ‘TRUE’ if the routines
                00109 ;                               ; for Multiplication & Division needs
                00110 ;                               ; to be assembled as Signed Integer
                00111 ;                               ; Routines. If ‘FALSE’ the above two
                00112 ;                               ; routines ( D_mpy & D_div ) use
                00113 ;                               ; unsigned arithmetic.
                00114 ;*******************************************************************
                00115 ;       Double Precision Divide ( 16/16 -> 16 )
                00116 ;
                00117 ;         ( ACCb/ACCa -> ACCb with remainder in ACCc ) : 16 bit output
                ; with Quotiont in ACCb (ACCbHI,ACCbLO) and Remainder in
                ; ACCc (ACCcHI,ACCcLO).
                ;   NOTE: Before calling this routine, the user should make sure that
                ;         the Numerator(ACCb) is greater than Denominator(ACCa). If
                ;         the case is not true, the user should scale either Numerator
                ;         or Denominator or both such that Numerator is greater than
                ;         the Denominator.
                ;
                ;
0017            D_divS
                ;
                     IF   SIGNED
                     CALL    S_SIGN
                     ENDIF
                ;
0017 0933               call    setup
0018 0075               clrf    ACCcHI
0019 0074               clrf    ACCcLO
001A 0403       dloop   bcf     STATUS,C
001B 0376               rlf     ACCdLO, F
001C 0377               rlf     ACCdHI, F
001D 0374               rlf     ACCcLO, F
001E 0375               rlf     ACCcHI, F
001F 0211               movf    ACCaHI,W
0020 0095               subwf   ACCcHI,W          ; check if a>c
0021 0743               btfss   STATUS,Z
0022 0A25               goto    nochk
0023 0210               movf    ACCaLO,W
0024 0094               subwf   ACCcLO,W          ; if msb equal then check lsb
0025 0703       nochk   btfss   STATUS,C          ; carry set if c>a
0026 0A2E               goto    nogo
0027 0210               movf    ACCaLO,W          ; c-a into c
0028 00B4               subwf   ACCcLO, F
0029 0703               btfss   STATUS,C
002A 00F5               decf    ACCcHI, F
002B 0211               movf    ACCaHI,W
002C 00B5               subwf   ACCcHI, F
002D 0503               bsf     STATUS,C         ; shift a 1 into b (result)
002E 0373       nogo    rlf     ACCbLO, F
002F 0374               rlf     ACCbHI, F
0030 02F8               decfsz  temp, F          ; loop untill all bits checked
0031 0A1A               goto    dloop
                ;
                    IF    SIGNED
                        btfss   sign,MSB        ; check sign if negative
                        retlw   0
                        goto    neg_B           ; negate ACCa ( -ACCa -> ACCa )
                    ELSE
0032 0800               retlw   0
                    ENDIF
                ;
                ;*******************************************************************
                ;
0033 0C10       setup   movlw   .16             ; for 16 shifts
0034 0038               movwf   temp
0035 0214               movf    ACCbHI,W        ; move ACCb to ACCd
0036 0037               movwf   ACCdHI
0037 0213               movf    ACCbLO,W
0038 0036               movwf   ACCdLO
0039 0074               clrf    ACCbHI
003A 0073               clrf    ACCbLO
003B 0800               retlw   0
                ;
                ;*******************************************************************
                ;
003C 0270       neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
```

```
003D 02B0               incf    ACCaLO, F
003E 0643               btfsc   STATUS,Z
003F 00F1               decf    ACCaHI, F
0040 0271               comf    ACCaHI, F
0041 0800               retlw   0
                ;
                ;*******************************************************************
                ;  Assemble this section only if Signed Arithmetic Needed
                ;
                     IF    SIGNED
                ;
                S_SIGN  movf    ACCaHI,W
                        xorwf   ACCbHI,W
                        movwf   sign
                        btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
                        goto    chek_A
                ;
                        comf    ACCbLO          ; negate ACCb
                        incf    ACCbLO
                        btfsc   STATUS,Z
                        decf    ACCbHI
                        comf    ACCbHI
                ;
                chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
                        retlw   0
                        goto    neg_A
                ;
                     ENDIF
                ;
                ;
0042 0900       Sqrt    call    init
0043 021D       sloop   movf    NumLo,W
0044 0033               movwf   ACCbLO
0045 021E               movf    NumHi,W
0046 0034               movwf   ACCbHI
                ;
0047 0917               call    D_divS          ; double precision division
0048 0910               call    D_add           ; double precision addition
                ;                               ; the above 2 routines are listed
                ;                               ; as seperate routines
0049 090A               call    div2
004A 02FF               decfsz  count, F
004B 0A43               goto    sloop
004C 0A52               goto    over            ; all iterations done
                ;                               ; branch back to desired location
                ;
                ;*************************************************************
                ;               Test Program
                ;*************************************************************
                ;
004D 0CF3       main    movlw   0F3
004E 003E               movwf   NumHi
004F 0CF6               movlw   0F6       ; Set input test number = 62454
0050 003D               movwf   NumLo     ;  = F3F6h
                ;
0051 0A42               goto    Sqrt      ; cannot use CALL : Math routines
                ;                         ; use up all the stack.
0052 0000       over    nop               ; all iterations done
                ;
0053 0A53       self    goto    self      ; result = 00F9h = 249
                ;                         ; exact sqrt(62454) = 249.9
                ;
01FF                    org     PIC54
01FF 0A4D               goto    main
                ;
                        END
```

```
MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXX------------ ---------------- ----------------
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:    85
Program Memory Words Free:   427


Errors   : 0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX K:DOUBLE PRECISION DIVISION LISTING (LOOPED)

```
MPASM 01.40 Released         DBL_DIVF.ASM   1-16-1997  12:51:16         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                00001         LIST    P = 16C54, n = 66
                00002 ;
                00003 ;********************************************************************
                00004 ;                       Double Precision Division
                00005 ;
                00006 ;               ( Optimized for Speed : straight Line Code )
                00007 ;
                00008 ;********************************************************************;
                00009 ; Division : ACCb(16 bits)/ACCa(16 bits)-> ACCb(16 bits) with
                00010 ;                                          Remainder in ACCc (16 bits)
                00011 ;      (a) Load the Denominator in location ACCaHI & ACCaLO ( 16 bits )
                00012 ;      (b) Load the Numerator in location ACCbHI & ACCbLO ( 16 bits )
                00013 ;      (c) CALL D_div
                00014 ;      (d) The 16 bit result is in location ACCbHI & ACCbLO
                00015 ;      (e) The 16 bit Remainder is in locations ACCcHI & ACCcLO
                00016 ;
                00017 ;   Performance :
                00018 ;               Program Memory  :       370
                00019 ;               Clock Cycles    :       263
                00020 ;
                00021 ;       NOTE :
                00022 ;               The performance specs are for Unsigned arithmetic (i.e,
                00023 ;               with "SIGNED equ  FALSE").
                00024 ;
                00025 ;
                00026 ;       Program:          DBL_DIVF.ASM
                00027 ;       Revision Date:
                00028 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                00029 ;
                00030 ;********************************************************************;
                00031 ;
  00000010      00032 ACCaLO  equ     10
  00000011      00033 ACCaHI  equ     11
  00000012      00034 ACCbLO  equ     12
  00000013      00035 ACCbHI  equ     13
  00000014      00036 ACCcLO  equ     14
  00000015      00037 ACCcHI  equ     15
  00000016      00038 ACCdLO  equ     16
  00000017      00039 ACCdHI  equ     17
  00000018      00040 temp    equ     18
  00000019      00041 sign    equ     19
                00042 ;
                00043         include "p16c5x.inc"
                00001         LIST
                00002 ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                00224         LIST
                00044
  000001FF      00045 PIC54   equ     1FFH    ; Define Reset Vector
  00000001      00046 TRUE    equ     1
  00000000      00047 FALSE   equ     0
                00048
0000            00049         org     0
                00050 ;********************************************************************
  00000000      00051 SIGNED  equ     FALSE           ; Set This To 'TRUE' if the routines
```

```
                ;                               ; for Multiplication & Division needs
                ;                               ; to be assembled as Signed Integer
                ;                               ; Routines. If ‘FALSE’ the above two
                ;                               ; routines ( D_mpy & D_div ) use
                ;                               ; unsigned arithmetic.
                ;*******************************************************************;
                ;       division macro
                ;
                divMac  MACRO
                        LOCAL   NOCHK
                        LOCAL   NOGO
                ;
                        bcf     STATUS,C
                        rlf     ACCdLO, F
                        rlf     ACCdHI, F
                        rlf     ACCcLO, F
                        rlf     ACCcHI, F
                        movf    ACCaHI,W
                        subwf   ACCcHI,W          ; check if a>c
                        btfss   STATUS,Z
                        goto    NOCHK
                        movf    ACCaLO,W
                        subwf   ACCcLO,W          ; if msb equal then check lsb
                NOCHK   btfss   STATUS,C          ; carry set if c>a
                        goto    NOGO
                        movf    ACCaLO,W          ; c-a into c
                        subwf   ACCcLO, F
                        btfss   STATUS,C
                        decf    ACCcHI, F
                        movf    ACCaHI,W
                        subwf   ACCcHI, F
                        bsf     STATUS,C         ; shift a 1 into b (result)
                NOGO    rlf     ACCbLO, F
                        rlf     ACCbHI, F
                ;
                        ENDM
                ;
                ;*******************************************************************
                ;       Double Precision Divide ( 16/16 -> 16 )
                ;
                ;         ( ACCb/ACCa -> ACCb with remainder in ACCc ) : 16 bit output
                ; with Quotiont in ACCb (ACCbHI,ACCbLO) and Remainder in ACCc
                ; (ACCcHI,ACCcLO).
                ;   NOTE: Before calling this routine, the user should make sure that
                ;         the Numerator(ACCb) is greater than Denominator(ACCa). If
                ;         the case is not true, the user should scale either Numerator
                ;         or Denominator or both such that Numerator is greater than
                ;         the Denominator.
                ;
                ;
0000 0C10       setup   movlw   .16               ; for 16 shifts
0001 0038               movwf   temp
0002 0213               movf    ACCbHI,W          ; move ACCb to ACCd
0003 0037               movwf   ACCdHI
0004 0212               movf    ACCbLO,W
0005 0036               movwf   ACCdLO
0006 0073               clrf    ACCbHI
0007 0072               clrf    ACCbLO
0008 0800               retlw   0
                ;
                ;*******************************************************************
                ;
0009 0270       neg_A   comf    ACCaLO, F          ; negate ACCa ( -ACCa -> ACCa )
000A 02B0               incf    ACCaLO, F
000B 0643               btfsc   STATUS,Z
000C 00F1               decf    ACCaHI, F
```

```
000D 0271       00118         comf    ACCaHI, F
000E 0800       00119         retlw   0
                00120 ;
                00121 ;*******************************************************************
                00122
                00123 ;
000F            00124 D_divF
                00125 ;
                00126      IF   SIGNED
                00127      CALL    S_SIGN
                00128      ENDIF
                00129 ;
000F 0900       00130         call    setup
0010 0075       00131         clrf    ACCcHI
0011 0074       00132         clrf    ACCcLO
                00133 ;
                00134 ; use the mulMac macro 16 times
                00135 ;
                00136         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0012 0403           M         bcf     STATUS,C
0013 0376           M         rlf     ACCdLO, F
0014 0377           M         rlf     ACCdHI, F
0015 0374           M         rlf     ACCcLO, F
0016 0375           M         rlf     ACCcHI, F
0017 0211           M         movf    ACCaHI,W
0018 0095           M         subwf   ACCcHI,W         ; check if a>c
0019 0743           M         btfss   STATUS,Z
001A 0A1D           M         goto    NOCHK
001B 0210           M         movf    ACCaLO,W
001C 0094           M         subwf   ACCcLO,W         ; if msb equal then check lsb
001D 0703           M NOCHK   btfss   STATUS,C         ; carry set if c>a
001E 0A26           M         goto    NOGO
001F 0210           M         movf    ACCaLO,W         ; c-a into c
0020 00B4           M         subwf   ACCcLO, F
0021 0703           M         btfss   STATUS,C
0022 00F5           M         decf    ACCcHI, F
0023 0211           M         movf    ACCaHI,W
0024 00B5           M         subwf   ACCcHI, F
0025 0503           M         bsf     STATUS,C         ; shift a 1 into b (result)
0026 0372           M NOGO    rlf     ACCbLO, F
0027 0373           M         rlf     ACCbHI, F
                    M ;
                00137         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0028 0403           M         bcf     STATUS,C
0029 0376           M         rlf     ACCdLO, F
002A 0377           M         rlf     ACCdHI, F
002B 0374           M         rlf     ACCcLO, F
002C 0375           M         rlf     ACCcHI, F
002D 0211           M         movf    ACCaHI,W
002E 0095           M         subwf   ACCcHI,W         ; check if a>c
002F 0743           M         btfss   STATUS,Z
0030 0A33           M         goto    NOCHK
0031 0210           M         movf    ACCaLO,W
0032 0094           M         subwf   ACCcLO,W         ; if msb equal then check lsb
0033 0703           M NOCHK   btfss   STATUS,C         ; carry set if c>a
0034 0A3C           M         goto    NOGO
0035 0210           M         movf    ACCaLO,W         ; c-a into c
0036 00B4           M         subwf   ACCcLO, F
0037 0703           M         btfss   STATUS,C
0038 00F5           M         decf    ACCcHI, F
```

```
0039 0211           M         movf    ACCaHI,W
003A 00B5           M         subwf   ACCcHI, F
003B 0503           M         bsf     STATUS,C        ; shift a 1 into b (result)
003C 0372           M NOGO    rlf     ACCbLO, F
003D 0373           M         rlf     ACCbHI, F
                    M ;
                00138         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
003E 0403           M         bcf     STATUS,C
003F 0376           M         rlf     ACCdLO, F
0040 0377           M         rlf     ACCdHI, F
0041 0374           M         rlf     ACCcLO, F
0042 0375           M         rlf     ACCcHI, F
0043 0211           M         movf    ACCaHI,W
0044 0095           M         subwf   ACCcHI,W        ; check if a>c
0045 0743           M         btfss   STATUS,Z
0046 0A49           M         goto    NOCHK
0047 0210           M         movf    ACCaLO,W
0048 0094           M         subwf   ACCcLO,W        ; if msb equal then check lsb
0049 0703           M NOCHK   btfss   STATUS,C        ; carry set if c>a
004A 0A52           M         goto    NOGO
004B 0210           M         movf    ACCaLO,W        ; c-a into c
004C 00B4           M         subwf   ACCcLO, F
004D 0703           M         btfss   STATUS,C
004E 00F5           M         decf    ACCcHI, F
004F 0211           M         movf    ACCaHI,W
0050 00B5           M         subwf   ACCcHI, F
0051 0503           M         bsf     STATUS,C        ; shift a 1 into b (result)
0052 0372           M NOGO    rlf     ACCbLO, F
0053 0373           M         rlf     ACCbHI, F
                    M ;
                00139         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0054 0403           M         bcf     STATUS,C
0055 0376           M         rlf     ACCdLO, F
0056 0377           M         rlf     ACCdHI, F
0057 0374           M         rlf     ACCcLO, F
0058 0375           M         rlf     ACCcHI, F
0059 0211           M         movf    ACCaHI,W
005A 0095           M         subwf   ACCcHI,W          ; check if a>c
005B 0743           M         btfss   STATUS,Z
005C 0A5F           M         goto    NOCHK
005D 0210           M         movf    ACCaLO,W
005E 0094           M         subwf   ACCcLO,W         ; if msb equal then check lsb
005F 0703           M NOCHK   btfss   STATUS,C         ; carry set if c>a
0060 0A68           M         goto    NOGO
0061 0210           M         movf    ACCaLO,W         ; c-a into c
0062 00B4           M         subwf   ACCcLO, F
0063 0703           M         btfss   STATUS,C
0064 00F5           M         decf    ACCcHI, F
0065 0211           M         movf    ACCaHI,W
0066 00B5           M         subwf   ACCcHI, F
0067 0503           M         bsf     STATUS,C         ; shift a 1 into b (result)
0068 0372           M NOGO    rlf     ACCbLO, F
0069 0373           M         rlf     ACCbHI, F
                    M ;
                00140         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
006A 0403           M         bcf     STATUS,C
006B 0376           M         rlf     ACCdLO, F
```

```
006C 0377           M         rlf     ACCdHI, F
006D 0374           M         rlf     ACCcLO, F
006E 0375           M         rlf     ACCcHI, F
006F 0211           M         movf    ACCaHI,W
0070 0095           M         subwf   ACCcHI,W          ; check if a>c
0071 0743           M         btfss   STATUS,Z
0072 0A75           M         goto    NOCHK
0073 0210           M         movf    ACCaLO,W
0074 0094           M         subwf   ACCcLO,W          ; if msb equal then check lsb
0075 0703           M NOCHK   btfss   STATUS,C          ; carry set if c>a
0076 0A7E           M         goto    NOGO
0077 0210           M         movf    ACCaLO,W          ; c-a into c
0078 00B4           M         subwf   ACCcLO, F
0079 0703           M         btfss   STATUS,C
007A 00F5           M         decf    ACCcHI, F
007B 0211           M         movf    ACCaHI,W
007C 00B5           M         subwf   ACCcHI, F
007D 0503           M         bsf     STATUS,C          ; shift a 1 into b (result)
007E 0372           M NOGO    rlf     ACCbLO, F
007F 0373           M         rlf     ACCbHI, F
                    M ;
                00141         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0080 0403           M         bcf     STATUS,C
0081 0376           M         rlf     ACCdLO, F
0082 0377           M         rlf     ACCdHI, F
0083 0374           M         rlf     ACCcLO, F
0084 0375           M         rlf     ACCcHI, F
0085 0211           M         movf    ACCaHI,W
0086 0095           M         subwf   ACCcHI,W          ; check if a>c
0087 0743           M         btfss   STATUS,Z
0088 0A8B           M         goto    NOCHK
0089 0210           M         movf    ACCaLO,W
008A 0094           M         subwf   ACCcLO,W          ; if msb equal then check lsb
008B 0703           M NOCHK   btfss   STATUS,C          ; carry set if c>a
008C 0A94           M         goto    NOGO
008D 0210           M         movf    ACCaLO,W           ;c-a into c
008E 00B4           M         subwf   ACCcLO, F
008F 0703           M         btfss   STATUS,C
0090 00F5           M         decf    ACCcHI, F
0091 0211           M         movf    ACCaHI,W
0092 00B5           M         subwf   ACCcHI, F
0093 0503           M         bsf     STATUS,C          ; shift a 1 into b (result)
0094 0372           M NOGO    rlf     ACCbLO, F
0095 0373           M         rlf     ACCbHI, F
                    M ;
                00142         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0096 0403           M         bcf     STATUS,C
0097 0376           M         rlf     ACCdLO, F
0098 0377           M         rlf     ACCdHI, F
0099 0374           M         rlf     ACCcLO, F
009A 0375           M         rlf     ACCcHI, F
009B 0211           M         movf    ACCaHI,W
009C 0095           M         subwf   ACCcHI,W          ; check if a>c
009D 0743           M         btfss   STATUS,Z
009E 0AA1           M         goto    NOCHK
009F 0210           M         movf    ACCaLO,W
00A0 0094           M         subwf   ACCcLO,W          ; if msb equal then check lsb
00A1 0703           M NOCHK   btfss   STATUS,C          ; carry set if c>a
00A2 0AAA           M         goto    NOGO
00A3 0210           M         movf    ACCaLO,W          ; c-a into c
```

```
00A4 00B4           M         subwf   ACCcLO, F
00A5 0703           M         btfss   STATUS,C
00A6 00F5           M         decf    ACCcHI, F
00A7 0211           M         movf    ACCaHI,W
00A8 00B5           M         subwf   ACCcHI, F
00A9 0503           M         bsf     STATUS,C          ; shift a 1 into b (result)
00AA 0372           M NOGO    rlf     ACCbLO, F
00AB 0373           M         rlf     ACCbHI, F
                    M ;
                00143         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
00AC 0403           M         bcf     STATUS,C
00AD 0376           M         rlf     ACCdLO, F
00AE 0377           M         rlf     ACCdHI, F
00AF 0374           M         rlf     ACCcLO, F
00B0 0375           M         rlf     ACCcHI, F
00B1 0211           M         movf    ACCaHI,W
00B2 0095           M         subwf   ACCcHI,W          ; check if a>c
00B3 0743           M         btfss   STATUS,Z
00B4 0AB7           M         goto    NOCHK
00B5 0210           M         movf    ACCaLO,W
00B6 0094           M         subwf   ACCcLO,W         ; if msb equal then check lsb
00B7 0703           M NOCHK   btfss   STATUS,C         ; carry set if c>a
00B8 0AC0           M         goto    NOGO
00B9 0210           M         movf    ACCaLO,W         ; c-a into c
00BA 00B4           M         subwf   ACCcLO, F
00BB 0703           M         btfss   STATUS,C
00BC 00F5           M         decf    ACCcHI, F
00BD 0211           M         movf    ACCaHI,W
00BE 00B5           M         subwf   ACCcHI, F
00BF 0503           M         bsf     STATUS,C         ; shift a 1 into b (result)
00C0 0372           M NOGO    rlf     ACCbLO, F
00C1 0373           M         rlf     ACCbHI, F
                    M ;
                00144         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
00C2 0403           M         bcf     STATUS,C
00C3 0376           M         rlf     ACCdLO, F
00C4 0377           M         rlf     ACCdHI, F
00C5 0374           M         rlf     ACCcLO, F
00C6 0375           M         rlf     ACCcHI, F
00C7 0211           M         movf    ACCaHI,W
00C8 0095           M         subwf   ACCcHI,W          ; check if a>c
00C9 0743           M         btfss   STATUS,Z
00CA 0ACD           M         goto    NOCHK
00CB 0210           M         movf    ACCaLO,W
00CC 0094           M         subwf   ACCcLO,W         ; if msb equal then check lsb
00CD 0703           M NOCHK   btfss   STATUS,C         ; carry set if c>a
00CE 0AD6           M         goto    NOGO
00CF 0210           M         movf    ACCaLO,W         ; c-a into c
00D0 00B4           M         subwf   ACCcLO, F
00D1 0703           M         btfss   STATUS,C
00D2 00F5           M         decf    ACCcHI, F
00D3 0211           M         movf    ACCaHI,W
00D4 00B5           M         subwf   ACCcHI, F
00D5 0503           M         bsf     STATUS,C         ; shift a 1 into b (result)
00D6 0372           M NOGO    rlf     ACCbLO, F
00D7 0373           M         rlf     ACCbHI, F
                    M ;
                00145         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
```

```
                    M ;
00D8 0403           M         bcf     STATUS,C
00D9 0376           M         rlf     ACCdLO, F
00DA 0377           M         rlf     ACCdHI, F
00DB 0374           M         rlf     ACCcLO, F
00DC 0375           M         rlf     ACCcHI, F
00DD 0211           M         movf    ACCaHI,W
00DE 0095           M         subwf   ACCcHI,W           ; check if a>c
00DF 0743           M         btfss   STATUS,Z
00E0 0AE3           M         goto    NOCHK
00E1 0210           M         movf    ACCaLO,W
00E2 0094           M         subwf   ACCcLO,W           ; if msb equal then check lsb
00E3 0703           M NOCHK   btfss   STATUS,C           ; carry set if c>a
00E4 0AEC           M         goto    NOGO
00E5 0210           M         movf    ACCaLO,W           ; c-a into c
00E6 00B4           M         subwf   ACCcLO, F
00E7 0703           M         btfss   STATUS,C
00E8 00F5           M         decf    ACCcHI, F
00E9 0211           M         movf    ACCaHI,W
00EA 00B5           M         subwf   ACCcHI, F
00EB 0503           M         bsf     STATUS,C           ; shift a 1 into b (result)
00EC 0372           M NOGO    rlf     ACCbLO, F
00ED 0373           M         rlf     ACCbHI, F
                    M ;
                00146         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
00EE 0403           M         bcf     STATUS,C
00EF 0376           M         rlf     ACCdLO, F
00F0 0377           M         rlf     ACCdHI, F
00F1 0374           M         rlf     ACCcLO, F
00F2 0375           M         rlf     ACCcHI, F
00F3 0211           M         movf    ACCaHI,W
00F4 0095           M         subwf   ACCcHI,W          ; check if a>c
00F5 0743           M         btfss   STATUS,Z
00F6 0AF9           M         goto    NOCHK
00F7 0210           M         movf    ACCaLO,W
00F8 0094           M         subwf   ACCcLO,W          ; if msb equal then check lsb
00F9 0703           M NOCHK   btfss   STATUS,C          ; carry set if c>a
00FA 0B02           M         goto    NOGO
00FB 0210           M         movf    ACCaLO,W          ; c-a into c
00FC 00B4           M         subwf   ACCcLO, F
00FD 0703           M         btfss   STATUS,C
00FE 00F5           M         decf    ACCcHI, F
00FF 0211           M         movf    ACCaHI,W
0100 00B5           M         subwf   ACCcHI, F
0101 0503           M         bsf     STATUS,C          ; shift a 1 into b (result)
0102 0372           M NOGO    rlf     ACCbLO, F
0103 0373           M         rlf     ACCbHI, F
                    M ;
                00147         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0104 0403           M         bcf     STATUS,C
0105 0376           M         rlf     ACCdLO, F
0106 0377           M         rlf     ACCdHI, F
0107 0374           M         rlf     ACCcLO, F
0108 0375           M         rlf     ACCcHI, F
0109 0211           M         movf    ACCaHI,W
010A 0095           M         subwf   ACCcHI,W          ; check if a>c
010B 0743           M         btfss   STATUS,Z
010C 0B0F           M         goto    NOCHK
010D 0210           M         movf    ACCaLO,W
010E 0094           M         subwf   ACCcLO,W          ; if msb equal then check lsb
```

```
010F 0703           M NOCHK   btfss   STATUS,C        ; carry set if c>a
0110 0B18           M         goto    NOGO
0111 0210           M         movf    ACCaLO,W        ; c-a into c
0112 00B4           M         subwf   ACCcLO, F
0113 0703           M         btfss   STATUS,C
0114 00F5           M         decf    ACCcHI, F
0115 0211           M         movf    ACCaHI,W
0116 00B5           M         subwf   ACCcHI, F
0117 0503           M         bsf     STATUS,C        ; shift a 1 into b (result)
0118 0372           M NOGO    rlf     ACCbLO, F
0119 0373           M         rlf     ACCbHI, F
                    M ;
                00148         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
011A 0403           M         bcf     STATUS,C
011B 0376           M         rlf     ACCdLO, F
011C 0377           M         rlf     ACCdHI, F
011D 0374           M         rlf     ACCcLO, F
011E 0375           M         rlf     ACCcHI, F
011F 0211           M         movf    ACCaHI,W
0120 0095           M         subwf   ACCcHI,W         ; check if a>c
0121 0743           M         btfss   STATUS,Z
0122 0B25           M         goto    NOCHK
0123 0210           M         movf    ACCaLO,W
0124 0094           M         subwf   ACCcLO,W        ; if msb equal then check lsb
0125 0703           M NOCHK   btfss   STATUS,C        ; carry set if c>a
0126 0B2E           M         goto    NOGO
0127 0210           M         movf    ACCaLO,W        ; c-a into c
0128 00B4           M         subwf   ACCcLO, F
0129 0703           M         btfss   STATUS,C
012A 00F5           M         decf    ACCcHI, F
012B 0211           M         movf    ACCaHI,W
012C 00B5           M         subwf   ACCcHI, F
012D 0503           M         bsf     STATUS,C        ; shift a 1 into b (result)
012E 0372           M NOGO    rlf     ACCbLO, F
012F 0373           M         rlf     ACCbHI, F
                    M ;
                00149         divMac
  0000              M         LOCAL   NOCHK
  0000              M         LOCAL   NOGO
                    M ;
0130 0403           M         bcf     STATUS,C
0131 0376           M         rlf     ACCdLO, F
0132 0377           M         rlf     ACCdHI, F
0133 0374           M         rlf     ACCcLO, F
0134 0375           M         rlf     ACCcHI, F
0135 0211           M         movf    ACCaHI,W
0136 0095           M         subwf   ACCcHI,W        ; check if a>c
0137 0743           M         btfss   STATUS,Z
0138 0B3B           M         goto    NOCHK
0139 0210           M         movf    ACCaLO,W
013A 0094           M         subwf   ACCcLO,W        ; if msb equal then check lsb
013B 0703           M NOCHK   btfss   STATUS,C        ; carry set if c>a
013C 0B44           M         goto    NOGO
013D 0210           M         movf    ACCaLO,W        ; c-a into c
013E 00B4           M         subwf   ACCcLO, F
013F 0703           M         btfss   STATUS,C
0140 00F5           M         decf    ACCcHI, F
0141 0211           M         movf    ACCaHI,W
0142 00B5           M         subwf   ACCcHI, F
0143 0503           M         bsf     STATUS,C        ; shift a 1 into b (result)
0144 0372           M NOGO    rlf     ACCbLO, F
0145 0373           M         rlf     ACCbHI, F
                    M ;
```

```
              00150         divMac
  0000            M         LOCAL   NOCHK
  0000            M         LOCAL   NOGO
                  M ;
0146 0403         M         bcf     STATUS,C
0147 0376         M         rlf     ACCdLO, F
0148 0377         M         rlf     ACCdHI, F
0149 0374         M         rlf     ACCcLO, F
014A 0375         M         rlf     ACCcHI, F
014B 0211         M         movf    ACCaHI,W
014C 0095         M         subwf   ACCcHI,W          ; check if a>c
014D 0743         M         btfss   STATUS,Z
014E 0B51         M         goto    NOCHK
014F 0210         M         movf    ACCaLO,W
0150 0094         M         subwf   ACCcLO,W          ; if msb equal then check lsb
0151 0703         M NOCHK   btfss   STATUS,C          ; carry set if c>a
0152 0B5A         M         goto    NOGO
0153 0210         M         movf    ACCaLO,W          ; c-a into c
0154 00B4         M         subwf   ACCcLO, F
0155 0703         M         btfss   STATUS,C
0156 00F5         M         decf    ACCcHI, F
0157 0211         M         movf    ACCaHI,W
0158 00B5         M         subwf   ACCcHI, F
0159 0503         M         bsf     STATUS,C          ; shift a 1 into b (result)
015A 0372         M NOGO    rlf     ACCbLO, F
015B 0373         M         rlf     ACCbHI, F
                  M ;
              00151         divMac
  0000            M         LOCAL   NOCHK
  0000            M         LOCAL   NOGO
                  M ;
015C 0403         M         bcf     STATUS,C
015D 0376         M         rlf     ACCdLO, F
015E 0377         M         rlf     ACCdHI, F
015F 0374         M         rlf     ACCcLO, F
0160 0375         M         rlf     ACCcHI, F
0161 0211         M         movf    ACCaHI,W
0162 0095         M         subwf   ACCcHI,W          ; check if a>c
0163 0743         M         btfss   STATUS,Z
0164 0B67         M         goto    NOCHK
0165 0210         M         movf    ACCaLO,W
0166 0094         M         subwf   ACCcLO,W          ; if msb equal then check lsb
0167 0703         M NOCHK   btfss   STATUS,C          ; carry set if c>a
0168 0B70         M         goto    NOGO
0169 0210         M         movf    ACCaLO,W          ; c-a into c
016A 00B4         M         subwf   ACCcLO, F
016B 0703         M         btfss   STATUS,C
016C 00F5         M         decf    ACCcHI, F
016D 0211         M         movf    ACCaHI,W
016E 00B5         M         subwf   ACCcHI, F
016F 0503         M         bsf     STATUS,C          ; shift a 1 into b (result)
0170 0372         M NOGO    rlf     ACCbLO, F
0171 0373         M         rlf     ACCbHI, F
                  M ;
              00152 ;
              00153     IF    SIGNED
              00154         btfss   sign,MSB          ; check sign if negative
              00155         retlw   0
              00156         goto    neg_B             ; negate ACCa (-ACCa -> ACCa)
              00157     ELSE
0172 0800     00158         retlw   0
              00159     ENDIF
              00160 ;
              00161 ;*******************************************************************
              00162 ;  Assemble this section only if Signed Arithmetic Needed
              00163 ;
```

```
                00164      IF    SIGNED
                00165 ;
                00166 S_SIGN  movf    ACCaHI,W
                00167         xorwf   ACCbHI,W
                00168         movwf   sign
                00169         btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
                00170         goto    chek_A
                00171 ;
                00172         comf    ACCbLO            ; negate ACCb
                00173         incf    ACCbLO
                00174         btfsc   STATUS,Z
                00175         decf    ACCbHI
                00176         comf    ACCbHI
                00177 ;
                00178 chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
                00179         retlw   0
                00180         goto    neg_A
                00181 ;
                00182      ENDIF
                00183 ;
                00184 ;*******************************************************************
                00185 ;                       Test Program
                00186 ;*******************************************************************
                00187 ;    Load constant values to ACCa & ACCb for testing
                00188 ;
0173 0C01       00189 main    movlw   1
0174 0031       00190         movwf   ACCaHI
0175 0CFF       00191         movlw   0FF             ; loads ACCa = 01FF
0176 0030       00192         movwf   ACCaLO
                00193 ;
0177 0C7F       00194         movlw   07F
0178 0033       00195         movwf   ACCbHI
0179 0CFF       00196         movlw   0FF             ; loads ACCb = 7FFF
017A 0032       00197         movwf   ACCbLO
                00198
017B 090F       00199         call    D_divF          ; remainder in ACCc. Here ACCb = 0040 &
                00200                                                     ; ACCc=003F
017C 0B7C       00201 self    goto    self
                00202 ;
01FF            00203         org     PIC54
01FF 0B73       00204         goto    main
                00205         END
MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)


0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXX---
01C0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   382
Program Memory Words Free:   130


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX L:DOUBLE PRECISION DIVISION LISTING (FAST)

```
MPASM 01.40 Released         DBL_DIVS.ASM   1-16-1997  12:51:51         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                                 LIST    P = 16C54, n = 66
                          ;
                          ;*******************************************************************
                          ;                       Double Precision Division
                          ;
                          ;               ( Optimized for Code Size : Looped Code )
                          ;
                          ;*******************************************************************;
                          ; Division : ACCb(16 bits) / ACCa(16 bits) -> ACCb(16 bits) with
                          ;                                          Remainder in ACCc (16 bits)
                          ;    (a) Load the Denominator in location ACCaHI & ACCaLO ( 16 bits )
                          ;    (b) Load the Numerator in location ACCbHI & ACCbLO ( 16 bits )
                          ;    (c) CALL D_div
                          ;    (d) The 16 bit result is in location ACCbHI & ACCbLO
                          ;    (e) The 16 bit Remainder is in locations ACCcHI & ACCcLO
                          ;
                          ;   Performance :
                          ;               Program Memory  :       037
                          ;               Clock Cycles    :       310
                          ;
                          ;        NOTE :
                          ;               The performance specs are for Unsigned arithmetic
                          ;               ( i.e,with “SIGNED equ  FALSE “).
                          ;
                          ;
                          ;       Program:          DBL_DIVS.ASM
                          ;       Revision Date:
                          ;                         1-13-97      Compatibility with MPASMWIN 1.40
                          ;
                          ;*******************************************************************;
                          ;
  00000010                ACCaLO  equ     10
  00000011                ACCaHI  equ     11
  00000012                ACCbLO  equ     12
  00000013                ACCbHI  equ     13
  00000014                ACCcLO  equ     14
  00000015                ACCcHI  equ     15
  00000016                ACCdLO  equ     16
  00000017                ACCdHI  equ     17
  00000018                temp    equ     18
  00000019                sign    equ     19
                          ;
                                  include “p16c5x.inc”
                                  LIST
                          ;P16C5X.INC Standard Header File, Ver. 3.30 Microchip Technology, Inc.
                                  LIST

  000001FF                PIC54   equ     1FFH    ; Define Reset Vector
  00000001                TRUE    equ     1
  00000000                FALSE   equ     0

0000                              org     0
                          ;*******************************************************************
  00000000                SIGNED  equ     FALSE           ; Set This To ‘TRUE’ if the routines
```

```
                    ;                               ; for Multiplication & Division needs
                    ;                               ; to be assembled as Signed Integer
                    ;                               ; Routines. If ‘FALSE’ the above two
                    ;                               ; routines ( D_mpy & D_div ) use
                    ;                               ; unsigned arithmetic.
                    ;*******************************************************************
                    ;       Double Precision Divide ( 16/16 -> 16 )
                    ;
                    ; (ACCb/ACCa -> ACCb with remainder in ACCc) : 16 bit output
                    ; with Quotiont in ACCb (ACCbHI,ACCbLO) and Remainder in ACCc
                    ; (ACCcHI,ACCcLO).
                    ; NOTE: Before calling this routine, the user should make sure that
                    ;       the Numerator(ACCb) is greater than Denominator(ACCa). If
                    ;       the case is not true, the user should scale either Numerator
                    ;       or Denominator or both such that Numerator is greater than
                    ;       the Denominator.
                    ;
                    ;
0000                D_divS
                    ;
                         IF   SIGNED
                         CALL    S_SIGN
                         ENDIF
                    ;
0000 091C                   call    setup
0001 0075                   clrf    ACCcHI
0002 0074                   clrf    ACCcLO
0003 0403           dloop   bcf     STATUS,C
0004 0376                   rlf     ACCdLO, F
0005 0377                   rlf     ACCdHI, F
0006 0374                   rlf     ACCcLO, F
0007 0375                   rlf     ACCcHI, F
0008 0211                   movf    ACCaHI,W
0009 0095                   subwf   ACCcHI,W        ; check if a>c
000A 0743                   btfss   STATUS,Z
000B 0A0E                   goto    nochk
000C 0210                   movf    ACCaLO,W
000D 0094                   subwf   ACCcLO,W        ; if msb equal then check lsb
000E 0703           nochk   btfss   STATUS,C        ; carry set if c>a
000F 0A17                   goto    nogo
0010 0210                   movf    ACCaLO,W        ; c-a into c
0011 00B4                   subwf   ACCcLO, F
0012 0703                   btfss   STATUS,C
0013 00F5                   decf    ACCcHI, F
0014 0211                   movf    ACCaHI,W
0015 00B5                   subwf   ACCcHI, F
0016 0503                   bsf     STATUS,C        ; shift a 1 into b (result)
0017 0372           nogo    rlf     ACCbLO, F
0018 0373                   rlf     ACCbHI, F
0019 02F8                   decfsz  temp, F         ; loop untill all bits checked
001A 0A03                   goto    dloop
                    ;
                        IF    SIGNED
                            btfss   sign,MSB        ; check sign if negative
                            retlw   0
                            goto    neg_B           ; negate ACCa ( -ACCa -> ACCa )
                        ELSE
001B 0800                   retlw   0
                        ENDIF
                    ;
                    ;*******************************************************************
                    ;
001C 0C10           setup   movlw   .16             ; for 16 shifts
001D 0038                   movwf   temp
001E 0213                   movf    ACCbHI,W        ; move ACCb to ACCd
001F 0037                   movwf   ACCdHI
```

```
0020 0212           00118         movf    ACCbLO,W
0021 0036           00119         movwf   ACCdLO
0022 0073           00120         clrf    ACCbHI
0023 0072           00121         clrf    ACCbLO
0024 0800           00122         retlw   0
                    00123 ;
                    00124 ;*******************************************************************
                    00125 ;
0025 0270           00126 neg_A   comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
0026 02B0           00127         incf    ACCaLO, F
0027 0643           00128         btfsc   STATUS,Z
0028 00F1           00129         decf    ACCaHI, F
0029 0271           00130         comf    ACCaHI, F
002A 0800           00131         retlw   0
                    00132 ;
                    00133 ;*******************************************************************
                    00134 ;  Assemble this section only if Signed Arithmetic Needed
                    00135 ;
                    00136      IF    SIGNED
                    00137 ;
                    00138 S_SIGN  movf    ACCaHI,W
                    00139         xorwf   ACCbHI,W
                    00140         movwf   sign
                    00141         btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
                    00142         goto    chek_A
                    00143 ;
                    00144         comf    ACCbLO            ; negate ACCb
                    00145         incf    ACCbLO
                    00146         btfsc   STATUS,Z
                    00147         decf    ACCbHI
                    00148         comf    ACCbHI
                    00149 ;
                    00150 chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
                    00151         retlw   0
                    00152         goto    neg_A
                    00153 ;
                    00154      ENDIF
                    00155 ;
                    00156 ;*******************************************************************
                    00157 ;                       Test Program
                    00158 ;*******************************************************************
                    00159 ;    Load constant values to ACCa & ACCb for testing
                    00160 ;
002B 0C01           00161 main    movlw   1
002C 0031           00162         movwf   ACCaHI
002D 0CFF           00163         movlw   0FF             ; loads ACCa = 01FF
002E 0030           00164         movwf   ACCaLO
                    00165 ;
002F 0C7F           00166         movlw   07F
0030 0033           00167         movwf   ACCbHI
0031 0CFF           00168         movlw   0FF             ; loads ACCb = 7FFF
0032 0032           00169         movwf   ACCbLO
                    00170 ;
0033 0900           00171         call    D_divS          ; remainder in ACCc. Here ACCb =0040 &
ACCc=003F
                    00172 ;
0034 0A34           00173 self    goto    self
                    00174 ;
01FF                00175         org     PIC54
01FF 0A2B           00176         goto    main
                    00177         END
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)


0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXX-----------
01C0 : ---------------- ---------------- ---------------- ---------------X
```

```
All other memory blocks unused.

Program Memory Words Used:    54
Program Memory Words Free:   458


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX M:

```
    LIST
; P16C5X.INC  Standard Header File, Version 3.30    Microchip Technology, Inc.
    NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the 16C5X microcontrollers.  These names are taken to match
; the data sheets as closely as possible.  The microcontrollers included
; in this file are:

;    16C52
;    16C54
;    16CR54
;    16C54A
;    16CR54A
;    16C55
;    16C56
;    16C57
;    16CR57A
;    16CR57B
;    16C58A
;    16CR58A

; There is one group of symbols that is valid for all microcontrollers.
; Each microcontroller in this family also has its own section of special
; symbols.  Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /P16C54A
;       2. LIST directive in the source file
;               LIST   P=16C54A
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;      Revision History
;
;==========================================================================

;Rev:   Date:   Reason:

;3.30   07/16/96 Aligned processors with MPASM v1.40
;3.2004/09/96 Added 16C54B, 16CR56B, 16C58B
;3.10   12/14/95 Added 16C52
;3.01   11/29/95 Removed 16CR55
;3.00   10/16/95 Added new processors for MPASM v1.30
;2.04   07/26/95 Reformatted for readability
;2.03   06/21/95 Removed leading spaces

;==========================================================================
;
;       Generic Definitions
;
;==========================================================================

W                            EQU     H'0000'
F                            EQU     H'0001'

;----- Register Files -----------------------------------------------------
```

```
INDF                         EQU     H’0000’
TMR0                         EQU     H’0001’
PCL                          EQU     H’0002’
STATUS                       EQU     H’0003’
FSR                          EQU     H’0004’
PORTA                        EQU     H’0005’
PORTB                        EQU     H’0006’

;----- STATUS Bits --------------------------------------------------------

PA2                          EQU     H’0007’
PA1                          EQU     H’0006’
PA0                          EQU     H’0005’
NOT_TO                       EQU     H’0004’
NOT_PD                       EQU     H’0003’
Z                            EQU     H’0002’
DC                           EQU     H’0001’
C                            EQU     H’0000’

;----- OPTION Bits --------------------------------------------------------

T0CS                         EQU     H’0005’
T0SE                         EQU     H’0004’
PSA                          EQU     H’0003’
PS2                          EQU     H’0002’
PS1                          EQU     H’0001’
PS0                          EQU     H’0000’

;==========================================================================
;
;       Processor-dependent Definitions
;
;==========================================================================

        IFDEF __16C52
           __MAXRAM H’01F’
           #define __CONFIG_2
        ENDIF

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

        IFDEF __16C54
           __MAXRAM H’01F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16CR54
           __MAXRAM H’01F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16C54A
           __MAXRAM H’01F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16CR54A
           __MAXRAM H’01F’
           #define __CONFIG_1
        ENDIF
```

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

        IFDEF __16C55
                                                ; Register Files
PORTC                     EQU     H’0007’
           __MAXRAM H’01F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16C56
           __MAXRAM H’01F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16C57
                                                ; Register Files
PORTC                     EQU     H’0007’
           __MAXRAM H’07F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16CR57A
                                                ; Register Files
PORTC                     EQU     H’0007’
           __MAXRAM H’07F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16CR57B
                                                ; Register Files
PORTC                     EQU     H’0007’
           __MAXRAM H’07F’
           #define __CONFIG_1
        ENDIF

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

        IFDEF __16C58A
           __MAXRAM H’07F’
           #define __CONFIG_0
        ENDIF

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

        IFDEF __16CR58A
           __MAXRAM H’07F’
           #define __CONFIG_1
        ENDIF

;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

        IFDEF __CONFIG_0
_CP_ON                       EQU     H’0FF7’
```

```
_CP_OFF                      EQU     H’0FFF’
_WDT_ON                      EQU     H’0FFF’
_WDT_OFF                     EQU     H’0FFB’
_LP_OSC                      EQU     H’0FFC’
_XT_OSC                      EQU     H’0FFD’
_HS_OSC                      EQU     H’0FFE’
_RC_OSC                      EQU     H’0FFF’
           #undefine __CONFIG_0
        ENDIF


        IFDEF __CONFIG_1
_CP_ON                       EQU     H’0007’
_CP_OFF                      EQU     H’0FFF’
_WDT_ON                      EQU     H’0FFF’
_WDT_OFF                     EQU     H’0FFB’
_LP_OSC                      EQU     H’0FFC’
_XT_OSC                      EQU     H’0FFD’
_HS_OSC                      EQU     H’0FFE’
_RC_OSC                      EQU     H’0FFF’
           #undefine __CONFIG_1
        ENDIF

        IFDEF __CONFIG_2
_CP_ON                       EQU     H’0FF7’
_CP_OFF                      EQU     H’0FFF’
_XT_OSC                      EQU     H’0FFD’
_RC_OSC                      EQU     H’0FFF’
           #undefine __CONFIG_2
        ENDIF

    LIST
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX N:INCLUDE FILE FOR FIXED POINT ROUTINE

```
               processor16C71

;       define assembler constants

B0             equ 0
B1             equ 1
B2             equ 2
B3             equ 3
B4             equ 4
B5             equ 5

MSB            equ 7
LSB            equ 0

W              equ 0

;       define special function registers

               cblock 0x00; page 0 registers
                      INDF,RTCC,PCL,STATUS,FSR,TRISA,TRISB,ZZZZ,
                      ADCON0,ADRES,PCLATH,INTCON
               endc

               cblock 0x00; page 1 registers
                      INDF,OPTION,PCL,STATUS,FSR,PORTA,PORTB,ZZZZ,
                      ADCON1,ADRES,PCLATH,INTCON
               endc

;       define beginning of general purpose RAM

RAMSTART       equ    0x0C
RAMSTOP        equ    0x2F

;       define commonly used bits

;       STATUS bit definitions

               #define_C  STATUS,0
               #define_DC STATUS,1
               #define_Z  STATUS,2
               #define_PD STATUS,3
               #define_TO STATUS,4
               #define_RP0STATUS,5
               #define_PA0STATUS,5
               #define_RP1STATUS,6
               #define_PA1STATUS,6
               #define_IRPSTATUS,7
               #define_PA2STATUS,7
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX O:

```
;       16/8 PIC16 FIXED POINT DIVIDE ROUTINES  VERSION 1.5

;       Input:  fixed point arguments in AARG and BARG

;       Output: quotient AARG/BARG followed by remainder in REM

;       All timings are worst case cycle counts

;       It is useful to note that the additional routine FXD1507U
;       can be called in a signed divide application in the special case
;       where AARG > 0 and BARG > 0, thereby offering some improvement in
;       performance.

;         Routine            Clocks     Function

;       FXD1608S    188 16 bit/8 bit -> 16.08 signed fixed point divide

;       FXD1608U    294 16 bit/8 bit -> 16.08 unsigned fixed point divide

;       FXD1607U    174 16 bit/7 bit -> 16.07 unsigned fixed point divide

;       FXD1507U    166 15 bit/7 bit -> 15.07 unsigned fixed point divide

;       The above timings are based on the looped macros. If space permits,
;       approximately 41-50 clocks can be saved by using the unrolled macros.


                list    r=dec,x=on,t=off,p=16C71

                include <PIC16.INC>

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

;       Define divide register variables

ACC             equ     0x0D    ; most significant byte of contiguous 4 byte accumulator

SIGN            equ     0x13    ; save location for sign in MSB

TEMP            equ     0x19    ; temporary storage

;       Define binary operation arguments

AARG            equ     0x0D    ; most significant byte of argument A

BARG            equ     0x16    ; most significant byte of argument B

REM             equ     0x11    ; most significant byte of remainder

LOOPCOUNT       equ     0x14    ; loop counter

;       Note:   ( AARG+B0, AARG+B1 )  and  ( ACC+B0, ACC+B1)
;               reference the same storage locations, and similarly for
;               ( REM+B0, REM+B1 ) and ( ACC+B4, ACC+B5 )


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

;       16/08 BIT Division Macros
```

```
SDIV1608L       macro

;       Max Timing:     3+5+2+5*11+10+10+6*11+10+2 = 163 clks

;       Min Timing:     3+5+2+5*11+10+10+6*11+10+2 = 163 clks

;       PM: 42                              DM: 5

               MOVF          BARG+B0,W
               SUBWF         REM+B0
               RLF           ACC+B0

               RLF           ACC+B0,W
               RLF           REM+B0
               MOVF          BARG+B0,W
               ADDWF         REM+B0
               RLF           ACC+B0

               MOVLW         6
               MOVWF         LOOPCOUNT

LOOPS1608A     RLF           ACC+B0,W
               RLF           REM+B0
               MOVF          BARG+B0,W

               BTFSC         ACC+B0,LSB
               SUBWF         REM+B0
               BTFSS         ACC+B0,LSB
               ADDWF         REM+B0
               RLF           ACC+B0

               DECFSZ        LOOPCOUNT
               GOTO          LOOPS1608A

               RLF           ACC+B1,W
               RLF           REM+B0
               MOVF          BARG+B0,W

               BTFSC         ACC+B0,LSB
               SUBWF         REM+B0
               BTFSS         ACC+B0,LSB
               ADDWF         REM+B0
               RLF           ACC+B1

               MOVLW         7
               MOVWF         LOOPCOUNT

LOOPS1608B     RLF           ACC+B1,W
               RLF           REM+B0
               MOVF          BARG+B0,W

               BTFSC         ACC+B1,LSB
               SUBWF         REM+B0
               BTFSS         ACC+B1,LSB
               ADDWF         REM+B0
               RLF           ACC+B1

               DECFSZ        LOOPCOUNT
               GOTO          LOOPS1608B

               BTFSS         ACC+B1,LSB
               ADDWF         REM+B0

               endm

UDIV1608L  macro
```

```
;       Max Timing: 2+7*12+11+3+7*24+23 = 291 clks

;       Min Timing: 2+7*11+10+3+7*17+16 = 227 clks

;       PM: 39                                  DM: 7

                MOVLW           8
                MOVWF           LOOPCOUNT

LOOPU1608A      RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                SUBWF           REM+B0

                BTFSC           _C
                GOTO            UOK68A
                ADDWF           REM+B0
                BCF             _C
UOK68A          RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1608A

                CLRF            TEMP

                MOVLW           8
                MOVWF           LOOPCOUNT

LOOPU1608B      RLF             ACC+B1,W
                RLF             REM+B0
                RLF             TEMP
                MOVF            BARG+B0,W
                SUBWF           REM+B0
                CLRF            ACC+B5
                CLRW
                BTFSS           _C
                INCFSZ          ACC+B5,W
                SUBWF           TEMP

                BTFSC           _C
                GOTO            UOK68B
                MOVF            BARG+B0,W
                ADDWF           REM+B0
                CLRF            ACC+B5
                CLRW
                BTFSC           _C
                INCFSZ          ACC+B5,W
                ADDWF           TEMP

                BCF             _C
UOK68B          RLF             ACC+B1

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1608B

                endm

UDIV1607L       macro

;       Max Timing:     7+6*11+10+10+6*11+10+2 = 171 clks

;       Min Timing:     7+6*11+10+10+6*11+10+2 = 171 clks

;       PM: 39                                  DM: 5

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU1607A      RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1607A

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU1607B      RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B1,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1607B

                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0

                endm

UDIV1507L       macro

;       Max Timing:     3+5+2+5*11+10+10+6*11+10+2 = 163 clks

;       Min Timing:     3+5+2+5*11+10+10+6*11+10+2 = 163 clks

;       PM: 42                                DM: 5

                MOVF            BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                ADDWF           REM+B0
                RLF             ACC+B0
                MOVLW           6
                MOVWF           LOOPCOUNT

LOOPU1507A      RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1507A

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU1507B      RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B1,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1507B

                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0

                endm

SDIV1608        macro

;       Max Timing:     3+5+14*8+2 = 122 clks

;       Min Timing:     3+5+14*8+2 = 122 clks

;       PM: 122                         DM: 4

                variable i

                MOVF            BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                ADDWF           REM+B0
                RLF             ACC+B0
                i = 2

                while i < 8

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B0

                i=i+1

                endw

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i = 9

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B1,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i=i+1

                endw

                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0

                endm

UDIV1608  macro

;       restore = 9/21 clks,  nonrestore = 8/14 clks

;       Max Timing: 8*9+1+8*21 = 241 clks

;       Min Timing: 8*8+1+8*14 = 177 clks

;       PM: 241                                 DM: 6

                variable         i

                i = 0

                while i < 8

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                SUBWF           REM+B0

                BTFSC           _C
                GOTO            UOK68#v(i)
                ADDWF           REM+B0
                BCF             _C
UOK68#v(i)      RLF             ACC+B0

                i=i+1

                endw

                CLRF            TEMP

                i = 8

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B0
                RLF             TEMP
                MOVF            BARG+B0,W
                SUBWF           REM+B0
                CLRF            ACC+B5
                CLRW
                BTFSS           _C
                INCFSZ          ACC+B5,W
                SUBWF           TEMP

                BTFSC           _C
                GOTO            UOK68#v(i)
                MOVF            BARG+B0,W
                ADDWF           REM+B0
                CLRF            ACC+B5
                CLRW
                BTFSC           _C
                INCFSZ          ACC+B5,W
                ADDWF           TEMP

                BCF             _C
UOK68#v(i)      RLF             ACC+B1

                i=i+1

                endw

                endm

UDIV1607        macro

;       Max Timing:     5+15*8+2 = 127 clks

;       Min Timing:     5+15*8+2 = 127 clks

;       PM: 127                         DM: 4

                variable i

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                i = 1
                while i < 8

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B0

                i=i+1

                endw

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i = 9

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B1,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i=i+1

                endw

                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0

                endm

UDIV1507        macro

;       Max Timing:     3+5+14*8+2 = 122 clks

;       Min Timing:     3+5+14*8+2 = 122 clks

;       PM: 122                             DM: 4

                variable i

                MOVF            BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W
                ADDWF           REM+B0
                RLF             ACC+B0
                i = 2

                while i < 8

                RLF             ACC+B0,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B0

                i=i+1

                endw

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B0,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B0,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i = 9

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B0
                MOVF            BARG+B0,W

                BTFSC           ACC+B1,LSB
                SUBWF           REM+B0
                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0
                RLF             ACC+B1

                i=i+1

                endw

                BTFSS           ACC+B1,LSB
                ADDWF           REM+B0

                endm
;**********************************************************************************************
;**********************************************************************************************

;       16/8 Bit Signed Fixed Point Divide 16/8 -> 16.08

;       Input:  16 bit signed fixed point dividend in AARG+B0, AARG+B1
;               8 bit signed fixed point divisor in BARG+B0

;       Use:    CALL    FXD1608S

;       Output: 16 bit signed fixed point quotient in AARG+B0, AARG+B1
;               8 bit signed fixed point remainder in REM+B0

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     10+163+3 = 176 clks             A > 0, B > 0
;                                       11+163+11 = 185 clks            A > 0, B < 0
;                                       14+163+11 = 188 clks            A < 0, B > 0
;                                       15+163+3 = 181 clks             A < 0, B < 0

;       Min Timing:     10+163+3 = 176 clks             A > 0, B > 0
;                                       11+163+11 = 185 clks            A > 0, B < 0
;                                       14+163+11 = 188 clks            A < 0, B > 0
;                                       15+163+3 = 181 clks             A < 0, B < 0

;       PM: 15+42+10 = 67               DM: 6

FXD1608S        MOVF            AARG+B0,W
                XORWF           BARG+B0,W
                MOVWF           SIGN

                BTFSS           BARG+B0,MSB        ; if MSB set go & negate BARG
                GOTO            CA1608S

                COMF            BARG+B0
                INCF            BARG+B0

CA1608S         BTFSS           AARG+B0,MSB        ; if MSB set go & negate ACCa
                GOTO            C1608S

                COMF            AARG+B1
                INCF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

C1608S          CLRF            REM+B0

                SDIV1608L

                BTFSS           SIGN,MSB           ; negate (ACCc,ACCd)
                RETLW           0x00

                COMF            AARG+B1
                INCF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

                COMF            REM+B0
                INCF            REM+B0

                RETLW           0x00


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

;       16/8 Bit Unsigned Fixed Point Divide 16/8 -> 16.08

;       Input:  16 bit unsigned fixed point dividend in AARG+B0, AARG+B1
;               16 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1608U

;       Output: 16 bit unsigned fixed point quotient in AARG+B0, AARG+B1
;               16 bit unsigned fixed point remainder in REM+B0

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     1+291+2 = 294 clks

;       Min Timing:     1+227+2 = 230 clks
;       PM: 1+39+1 = 41         DM: 7

FXD1608U        CLRF            REM+B0

                UDIV1608L

                RETLW           0x00

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

;       16/7 Bit Unsigned Fixed Point Divide 16/7 -> 16.07

;       Input:  16 bit unsigned fixed point dividend in AARG+B0, AARG+B1
;               7 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1607U

;       Output: 16 bit unsigned fixed point quotient in AARG+B0, AARG+B1
;               7 bit unsigned fixed point remainder in REM+B0

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     1+171+2 = 174 clks

;       Min Timing:     1+171+2 = 174 clks

;       PM: 1+39+1 = 41         DM: 5

FXD1607U        CLRF            REM+B0

                UDIV1607L

                RETLW           0x00

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

;       15/7 Bit Unsigned Fixed Point Divide 15/7 -> 15.07

;       Input:  15 bit unsigned fixed point dividend in AARG+B0, AARG+B1
;               7 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1507U

;       Output: 15 bit unsigned fixed point quotient in AARG+B0, AARG+B1
;               7 bit unsigned fixed point remainder in REM+B0

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     1+163+2 = 166 clks

;       Min Timing:     1+163+2 = 166 clks

;       PM: 1+42+1 = 44         DM: 5

FXD1507U        CLRF            REM+B0

                UDIV1507L

                RETLW           0x00

                END
;**********************************************************************************************
;**********************************************************************************************
```

Please check the Microchip BBS for the latest version of the source code. Microchip’s Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX P:16/16 FIXED POINT DIVIDE ROUTINES

```
;       16/16 PIC16 FIXED POINT DIVIDE ROUTINES VERSION 1.5

;       Input:  fixed point arguments in AARG and BARG

;       Output: quotient AARG/BARG followed by remainder in REM

;       All timings are worst case cycle counts

;       It is useful to note that the additional routine FXD1515U
;       can be called in a signed divide application in the special case
;       where AARG > 0 and BARG > 0, thereby offering some improvement in
;       performance.

;         Routine            Clocks     Function

;       FXD1616S    319 16 bit/16 bit -> 16.16 signed fixed point divide

;       FXD1616U    373 16 bit/16 bit -> 16.16 unsigned fixed point divide

;       FXD1515U    294 15 bit/15 bit -> 15.15 unsigned fixed point divide

;       The above timings are based on the looped macros. If space permits,
;       approximately 65-69 clocks can be saved by using the unrolled macros.


                list    r=dec,x=on,t=off,p=16C71

                include <PIC16.INC>

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

;       Define divide register variables

ACC             equ     0x0D    ; most significant byte of contiguous 4 byte accumulator

SIGN            equ     0x13    ; save location for sign in MSB

TEMP            equ     0x19    ; temporary storage

;       Define binary operation arguments

AARG            equ     0x0D    ; most significant byte of argument A

BARG            equ     0x16    ; most significant byte of argument B

REM             equ     0x11    ; most significant byte of remainder

LOOPCOUNT       equ     0x14    ; loop counter

;       Note:   ( AARG+B0, AARG+B1 )  and  ( ACC+B0, ACC+B1)
;               reference the same storage locations, and similarly for
;               ( REM+B0, REM+B1 ) and ( ACC+B4, ACC+B5 )


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

;       16/16 Bit Division Macros
```

```
SDIV1616L       macro

;       Max Timing:     13+14*18+17+8 = 290 clks

;       Min Timing:     13+14*16+15+3 = 255 clks

;       PM: 42                                DM: 7

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B1
                RLF             ACC+B0

                MOVLW           15
                MOVWF           LOOPCOUNT

LOOPS1616       RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B1,LSB
                GOTO            SADD66L

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK66LL

SADD66L         ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK66LL         RLF             ACC+B1
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPS1616

                BTFSC           ACC+B1,LSB
                GOTO            SOK66L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
SOK66L

                endm

UDIV1616L       macro

;       restore = 23 clks,  nonrestore = 17 clks
;       Max Timing:     2+15*23+22 = 369 clks

;       Min Timing:     2+15*17+16 = 273 clks

;       PM: 24                                  DM: 7

                MOVLW           16
                MOVWF           LOOPCOUNT

LOOPU1616       RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0

                BTFSC           _C
                GOTO            UOK66LL
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

                BCF             _C

UOK66LL         RLF             ACC+B1
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1616

                endm

UDIV1515L       macro

;       Max Timing:     13+14*18+17+8 = 290 clks

;       Min Timing:     13+14*17+16+3 = 270 clks

;       PM: 42                                  DM: 7

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B1
                RLF             ACC+B0

                MOVLW           15
                MOVWF           LOOPCOUNT

LOOPU1515       RLF            ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            UADD55L

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK55LL

UADD55L         ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK55LL         RLF             ACC+B1
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1515

                BTFSC           ACC+B1,LSB
                GOTO            UOK55L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
UOK55L

                endm

SDIV1616        macro

;       Max Timing:     7+10+6*14+14+7*14+8 = 221 clks

;       Min Timing:     7+10+6*13+13+7*13+3 = 202 clks

;       PM: 7+10+6*18+18+7*18+8 = 277   DM: 6

                variable i

                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                RLF             ACC+B0

                i = 2

                while i < 8
                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B0,LSB
                GOTO            SADD66#v(i)

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK66#v(i)

SADD66#v(i)     ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK66#v(i)      RLF             ACC+B0

                i=i+1

                endw

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B0,LSB
                GOTO            SADD668

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK668

SADD668         ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK668          RLF             ACC+B1

                i = 9

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B1,LSB
                GOTO            SADD66#v(i)

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK66#v(i)

SADD66#v(i)     ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK66#v(i)      RLF             ACC+B1

                i=i+1

                endw

                BTFSC           ACC+B1,LSB
                GOTO            SOK66
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
SOK66

                endm

UDIV1616  macro

;       restore = 20 clks,  nonrestore = 14 clks

;       Max Timing: 16*20 = 320 clks

;       Min Timing: 16*14 = 224 clks

;       PM: 16*20 = 320         DM: 6

                variable         i

                i = 0

                while i < 16

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0

                BTFSC           _C
                GOTO            UOK66#v(i)
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

                BCF             _C

UOK66#v(i)      RLF             ACC+B1
```

```
                RLF             ACC+B0

                i=i+1

                endw

                endm

UDIV1515        macro

;       Max Timing:     7+10+6*14+14+7*14+8 = 221 clks

;       Min Timing:     7+10+6*13+13+7*13+3 = 202 clks

;       PM:     7+10+6*18+18+7*18+8 = 277       DM: 6

                variable i

                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                RLF             ACC+B0

                i = 2

                while i < 8

                RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B0,LSB
                GOTO            UADD55#v(i)

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK55#v(i)

UADD55#v(i)     ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK55#v(i)      RLF             ACC+B0

                i=i+1
                endw

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B0,LSB
                GOTO            UADD558

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK558

UADD558         ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK558          RLF             ACC+B1

                i = 9

                while i < 16

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W

                BTFSS           ACC+B1,LSB
                GOTO            UADD55#v(i)

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK55#v(i)

UADD55#v(i)     ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK55#v(i)      RLF             ACC+B1

                i=i+1

                endw

                BTFSC           ACC+B1,LSB
                GOTO            UOK55
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
UOK55
```

```
                endm

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

;       16/16 Bit Signed Fixed Point Divide 16/16 -> 16.16

;       Input:  16 bit fixed point dividend in AARG+B0, AARG+B1
;               16 bit fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1616S

;       Output: 16 bit fixed point quotient in AARG+B0, AARG+B1
;               16 bit fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     11+290+3 = 304 clks             A > 0, B > 0
;                                       15+290+14 = 319 clks            A > 0, B < 0
;                                       15+290+14 = 319 clks            A < 0, B > 0
;                                       19+290+3 = 312 clks             A < 0, B < 0

;       Min Timing:     11+255+3 = 269 clks             A > 0, B > 0
;                                       15+255+14 = 284 clks            A > 0, B < 0
;                                       15+255+14 = 284 clks            A < 0, B > 0
;                                       19+255+3 = 277 clks             A < 0, B < 0

;       PM: 19+42+13 = 74               DM: 8

FXD1616S        MOVF            AARG+B0,W
                XORWF           BARG+B0,W
                MOVWF           SIGN
                BTFSS           BARG+B0,MSB        ; if MSB set go & negate BARG
                GOTO            CA1616S

                COMF            BARG+B1
                INCF            BARG+B1
                BTFSC           _Z
                DECF            BARG+B0
                COMF            BARG+B0

CA1616S         BTFSS           AARG+B0,MSB        ; if MSB set go & negate ACCa
                GOTO            C1616S

                COMF            AARG+B1
                INCF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

C1616S          CLRF            REM+B0
                CLRF            REM+B1

                SDIV1616L

                BTFSS           SIGN,MSB          ; negate (ACCc,ACCd)
                RETLW           0x00

                COMF            AARG+B1
                INCF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

                COMF            REM+B1
                INCF            REM+B1
                BTFSC           _Z
                DECF            REM+B0
                COMF            REM+B0

                RETLW           0x00

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

;       16/16 Bit Unsigned Fixed Point Divide 16/16 -> 16.16

;       Input:  16 bit unsigned fixed point dividend in AARG+B0, AARG+B1
;               16 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1616U

;       Output: 16 bit unsigned fixed point quotient in AARG+B0, AARG+B1
;               16 bit unsigned fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     2+369+2 = 373 clks

;       Min Timing:     2+273+2 = 277 clks

;       PM: 2+24+1 = 27         DM: 7

FXD1616U        CLRF            REM+B0
                CLRF            REM+B1

                UDIV1616L

                RETLW           0x00

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

;       15/15 Bit Unsigned Fixed Point Divide 15/15 -> 15.15

;       Input:  15 bit unsigned fixed point dividend in AARG+B0, AARG+B1
;               15 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD1515U

;       Output: 15 bit unsigned fixed point quotient in AARG+B0, AARG+B1
;               15 bit unsigned fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     2+290+2 = 294 clks

;       Min Timing:     2+270+2 = 274 clks

;       PM: 2+42+1 = 45         DM: 7

FXD1515U        CLRF            REM+B0
                CLRF            REM+B1

                UDIV1515L

                RETLW           0x00

                END
;**********************************************************************************************
;**********************************************************************************************
```

Please check the Microchip BBS for the latest version of the source code. Microchip's Worldwide Web Address: www.microchip.com; Bulletin Board Support: MCHIPBBS using CompuServe® (CompuServe membership not required).

APPENDIX Q:32/16 FIXED POINT DIVIDE ROUTINES

```
;       32/16 PIC16 FIXED POINT DIVIDE ROUTINES VERSION 1.5

;       Input:  fixed point arguments in AARG and BARG

;       Output: quotient AARG/BARG followed by remainder in REM

;       All timings are worst case cycle counts

;       It is useful to note that the additional routine FXD3115U
;       can be called in a signed divide application in the special case
;       where AARG > 0 and BARG > 0, thereby offering some improvement in
;       performance.

;         Routine            Clocks     Function

;       FXD3216S    578 32 bit/16 bit -> 32.16 signed fixed point divide

;       FXD3216U    702 32 bit/16 bit -> 32.16 unsigned fixed point divide

;       FXD3115U    541 31 bit/15 bit -> 31.15 unsigned fixed point divide

                list    r=dec,x=on,t=off,p=16C71

                include <PIC16.INC>

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

;       Define divide register variables

ACC             equ     0x0D    ; most significant byte of contiguous 4 byte accumulator

SIGN            equ     0x13    ; save location for sign in MSB

TEMP            equ     0x19    ; temporary storage

;       Define binary operation arguments

AARG            equ     0x0D    ; most significant byte of argument A

BARG            equ     0x16    ; most significant byte of argument B

REM             equ     0x11    ; most significant byte of remainder

LOOPCOUNT       equ     0x14    ; loop counter

;       Note:   ( AARG+B0, AARG+B1 )  and  ( ACC+B0, ACC+B1)
;               reference the same storage locations, and similarly for
;               ( REM+B0, REM+B1 ) and ( ACC+B4, ACC+B5 )


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

;       32/16 Bit Division Macros

SDIV3216L       macro

;       Max Timing:     9+6*17+16+16+6*17+16+16+6*17+16+16+6*17+16+8 = 537 clks
;       Min Timing:     9+6*16+15+15+6*16+15+15+6*16+15+15+6*16+15+3 = 501 clks

;       PM: 157                                 DM: 9

                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPS3216A      RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            SADD26LA

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26LA

SADD26LA        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26LA         RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPS3216A

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            SADD26L8

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26L8

SADD26L8        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26L8         RLF             ACC+B1

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPS3216B      RLF             ACC+B1,W
```

```
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            SADD26LB

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26LB

SADD26LB        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26LB         RLF             ACC+B1

                DECFSZ          LOOPCOUNT
                GOTO            LOOPS3216B

                RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            SADD26L16

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26L16

SADD26L16       ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26L16        RLF             ACC+B2

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPS3216C      RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            SADD26LC

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26LC

SADD26LC        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26LC         RLF             ACC+B2

                DECFSZ          LOOPCOUNT
                GOTO            LOOPS3216C

                RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            SADD26L24

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26L24

SADD26L24       ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26L24        RLF             ACC+B3

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPS3216D      RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B3,LSB
                GOTO            SADD26LD

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            SOK26LD

SADD26LD        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

SOK26LD         RLF             ACC+B3

                DECFSZ          LOOPCOUNT
                GOTO            LOOPS3216D

                BTFSC           ACC+B3,LSB
                GOTO            SOK26L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
```

```
SOK26L

                endm

UDIV3216L       macro

;       Max Timing:     15+6*22+21+21+6*22+21+21+6*22+21+21+6*22+21+8 = 698 clks

;       Min Timing:     15+6*21+20+20+6*21+20+20+6*21+20+20+6*21+20+3 = 662 clks

;       PM: 233                                 DM: 11

                CLRF            TEMP

                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                RLF             ACC+B0

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3216A      RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            UADD26LA

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26LA

UADD26LA        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26LA         RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3216A

                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            UADD26L8

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26L8

UADD26L8        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26L8         RLF             ACC+B1

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3216B      RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            UADD26LB

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26LB

UADD26LB        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26LB         RLF             ACC+B1
```

```
                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3216B

                RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            UADD26L16

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26L16

UADD26L16       ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26L16        RLF             ACC+B2

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3216C      RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            UADD26LC

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26LC

UADD26LC        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26LC         RLF             ACC+B2

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3216C

                RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            UADD26L24

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26L24

UADD26L24       ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26L24        RLF             ACC+B3

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3216D      RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B3,LSB
                GOTO            UADD26LD

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSS           _C
                INCFSZ          SIGN,W
                SUBWF           TEMP
                GOTO            UOK26LD

UADD26LD        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0
                CLRF            SIGN
                CLRW
                BTFSC           _C
                INCFSZ          SIGN,W
                ADDWF           TEMP

UOK26LD         RLF             ACC+B3

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3216D

                BTFSC           ACC+B3,LSB
                GOTO            UOK26L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
UOK26L

                endm

UDIV3115L       macro

;       Max Timing:     9+6*17+16+16+6*17+16+16+6*17+16+16+6*17+16+8 = 537 clks

;       Min Timing:     9+6*16+15+15+6*16+15+15+6*16+15+15+6*16+15+3 = 501 clks

;       PM: 157                                 DM: 9

                MOVF            BARG+B1,W
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                RLF             ACC+B0

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3115A      RLF             ACC+B0,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            UADD15LA

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15LA

UADD15LA        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15LA         RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3115A
                RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B0,LSB
                GOTO            UADD15L8

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15L8

UADD15L8        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15L8         RLF             ACC+B1

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3115B      RLF             ACC+B1,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            UADD15LB

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15LB

UADD15LB        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15LB         RLF             ACC+B1

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3115B

                RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B1,LSB
                GOTO            UADD15L16

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15L16

UADD15L16       ADDWF           REM+B1
```

```
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15L16        RLF             ACC+B2

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3115C      RLF             ACC+B2,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            UADD15LC

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15LC

UADD15LC        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15LC         RLF             ACC+B2

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3115C

                RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B2,LSB
                GOTO            UADD15L24

                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15L24

UADD15L24       ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15L24        RLF             ACC+B3

                MOVLW           7
                MOVWF           LOOPCOUNT

LOOPU3115D      RLF             ACC+B3,W
                RLF             REM+B1
                RLF             REM+B0
                MOVF            BARG+B1,W
                BTFSS           ACC+B3,LSB
                GOTO            UADD15LD
                SUBWF           REM+B1
                MOVF            BARG+B0,W
                BTFSS           _C
                INCFSZ          BARG+B0,W
                SUBWF           REM+B0
                GOTO            UOK15LD

UADD15LD        ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCFSZ          BARG+B0,W
                ADDWF           REM+B0

UOK15LD         RLF             ACC+B3

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU3115D

                BTFSC           ACC+B3,LSB
                GOTO            UOK15L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
UOK15L

                endm

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

;       32/16 Bit Signed Fixed Point Divide 32/16 -> 32.16

;       Input:  32 bit fixed point dividend in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               16 bit fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD3216S

;       Output: 32 bit fixed point quotient in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               16 bit fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     11+537+3 = 551 clks             A > 0, B > 0
;                                       15+537+20 = 572 clks            A > 0, B < 0
;                                       21+537+20 = 578 clks            A < 0, B > 0
;                                       25+537+3 = 565 clks             A < 0, B < 0

;       Min Timing:     11+501+3 = 515 clks             A > 0, B > 0
;                                       15+501+20 = 536 clks            A > 0, B < 0
;                                       21+501+20 = 542 clks            A < 0, B > 0
;                                       25+501+3 = 529 clks             A < 0, B < 0

;       PM: 25+157+19 = 201             DM: 10

FXD3216S        MOVF            AARG+B0,W
                XORWF           BARG+B0,W
                MOVWF           SIGN
                BTFSS           BARG+B0,MSB        ; if MSB set go & negate BARG
                GOTO            CA3216S

                COMF            BARG+B1
                INCF            BARG+B1
                BTFSC           _Z
                DECF            BARG+B0
                COMF            BARG+B0

CA3216S         BTFSS           AARG+B0,MSB        ; if MSB set go & negate ACCa
                GOTO            C3216S

                COMF            AARG+B3
                INCF            AARG+B3
                BTFSC           _Z
                DECF            AARG+B2
                COMF            AARG+B2
                BTFSC           _Z
                DECF            AARG+B1
                COMF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

C3216S          CLRF            REM+B0
                CLRF            REM+B1

                SDIV3216L

                BTFSS           SIGN,MSB        ; negate (ACCc,ACCd)
                RETLW           0x00

                COMF            AARG+B3
                INCF            AARG+B3
                BTFSC           _Z
                DECF            AARG+B2
                COMF            AARG+B2
                BTFSC           _Z
                DECF            AARG+B1
                COMF            AARG+B1
                BTFSC           _Z
                DECF            AARG+B0
                COMF            AARG+B0

                COMF            REM+B1
                INCF            REM+B1
                BTFSC           _Z
                DECF            REM+B0
                COMF            REM+B0

                RETLW           0x00


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

;       32/16 Bit Unsigned Fixed Point Divide 32/16 -> 32.16

;       Input:  32 bit unsigned fixed point dividend in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               16 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD3216U

;       Output: 32 bit unsigned fixed point quotient in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               16 bit unsigned fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     2+698+2 = 702 clks

;       Max Timing:     2+662+2 = 666 clks
;       PM: 2+233+1 = 236               DM: 11

FXD3216U        CLRF            REM+B0
                CLRF            REM+B1

                UDIV3216L

                RETLW           0x00

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

;       31/15 Bit Unsigned Fixed Point Divide 31/15 -> 31.15

;       Input:  31 bit unsigned fixed point dividend in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               15 bit unsigned fixed point divisor in BARG+B0, BARG+B1

;       Use:    CALL    FXD3115U

;       Output: 31 bit unsigned fixed point quotient in AARG+B0, AARG+B1,AARG+B2,AARG+B3
;               15 bit unsigned fixed point remainder in REM+B0, REM+B1

;       Result: AARG, REM  <—  AARG / BARG

;       Max Timing:     2+537+2 = 541 clks

;       Min Timing:     2+501+2 = 505 clks

;       PM: 2+157+1 = 160               DM: 9

FXD3115U        CLRF            REM+B0
                CLRF            REM+B1

                UDIV3115L

                RETLW           0x00

                END
;*********************************************************************************************
;*********************************************************************************************
```

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

♻ Printed on recycled paper.

DNV Certification, Inc.
USA

ANSI★RAB
QMS
ACCREDITED

DNV MSC
The Netherlands
Accredited by the RvA

DNV

ISO 9001 / QS-9000
REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/01/02