PIC16C5X / 16CXX Math Utility Routines

INTRODUCTION

This application note provides some utility math routines for Microchip's PIC16C5X and PIC16CXX series of 8-bit microcontrollers. The following math routines are provided:

- 8 x 8 unsigned multiply
- 16 x 16 double precision multiply
- Fixed Point Division (see Table 3)
- 16 x 16 double precision addition
- 16 x 16 double precision subtraction
- BCD 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.

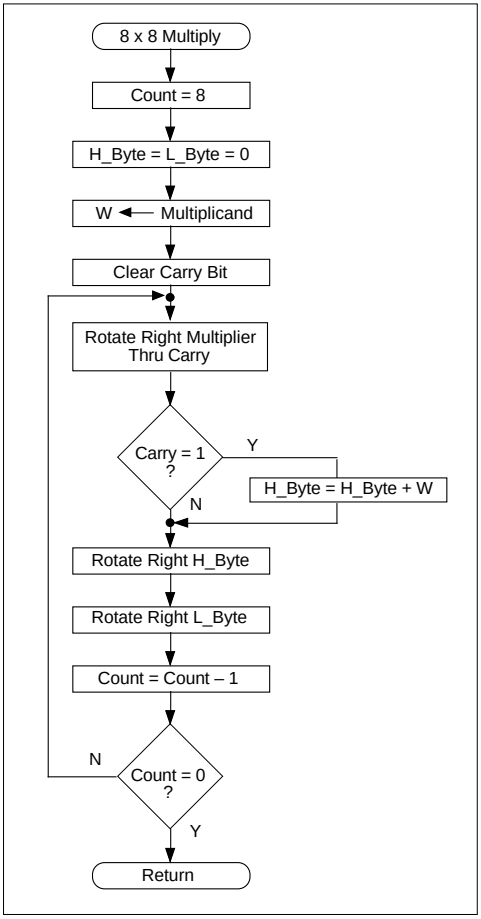
SINGLE PRECISION UNSIGNED MULTIPLICATION (8 X 8)

This routine computes the product of two unsigned 8-bit 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.

TABLE 1 - PERFORMANCE SPECS

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

FIGURE 1 - FLOW CHART FOR UN SIGNED 8 X 8 MULTIPLY



5

DOUBLE PRECISION MULTIPLICATION

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) and the other routine has been written to reduce the code size (a looped code). The listing of these routines are given in Appendices C and D. The performance specs for the 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 standarrestoring 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 methods exist 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 PIC16/17 family of microcontrollers, a more creative and efficient implementation of the above algorithms is possible. In what follows, the 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, the routines are named in the 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 the 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:

		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.

TABLE 3: FIXED POINT DIVIDE ROUTINES

Routine	Clocks	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

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.

5

TABLE 4: PIC16CXX FIXED POINT DIVIDE PERFORMANCE DATA

Routine	Max Cycles	Min. Cycles	PM	DM
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

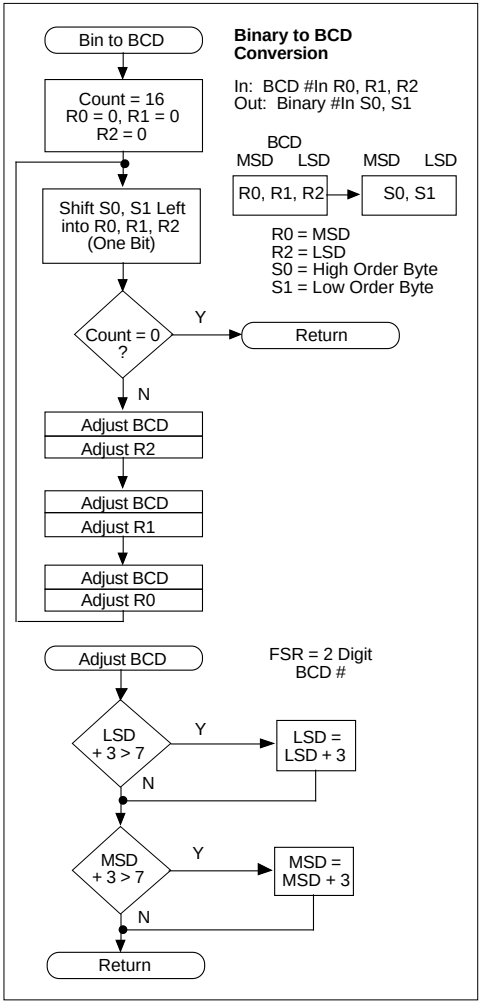
BINARY TO BCD CONVERSION

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 - FLOW CHART FOR BINARY TO BCD CONVERSION



BCD ADDITION & SUBTRACTION

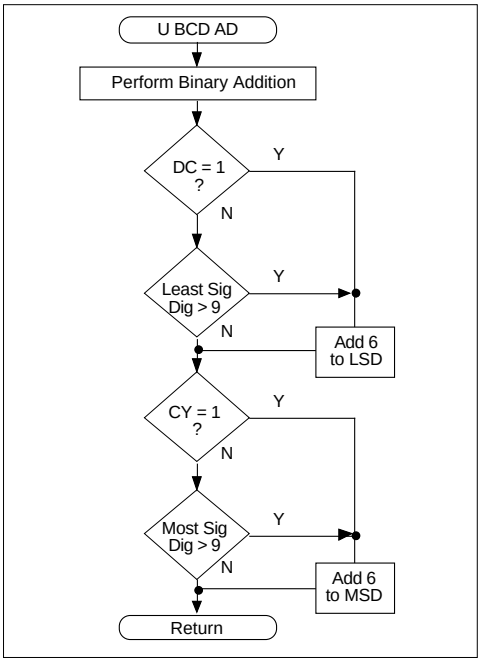
These two routines perform a two-digit unsigned BCD addition and subtraction. The results are the sum (or difference) in one File register and with a overflow carry-bit in another File register. The performance specs for the routines are shown below:

TABLE 8 - PERFORMANCE SPECS

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

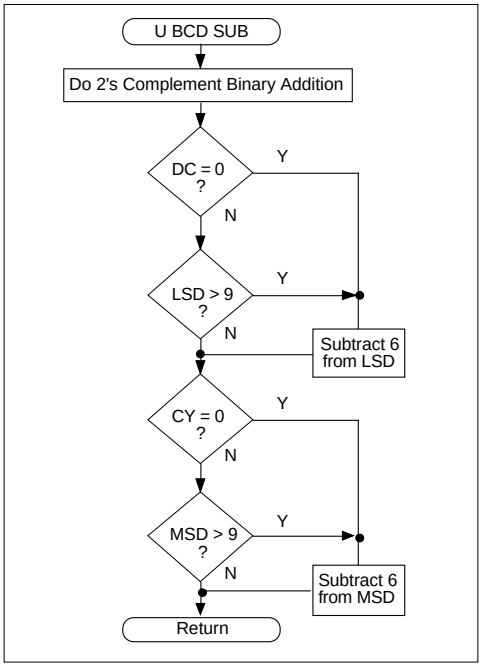
The listing files for the above two routines are given in Appendices J and K. The flow charts for BCD addition and BCD subtraction are given in Figures 3 and 4, respectively.

FIGURE 3 - FLOW CHART FOR BCD ADDITION



5

FIGURE 4 - FLOW CHART 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:

Equation 1: $f(Y) = f(Y_0) + (Y - Y_0) f'(Y_0) + (Y - Y_0)^2 f''(Y_0) / 2! + ...$

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

$$f(X) = f(Y_0) + (x - Y_0) f'(Y_0) + (X - Y_0)^2 f''(Y_0) / 2! + ... = 0$$

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., $X = Y_0 - f(Y_0) / f'(Y_0)$

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

Equation 3: $X_n = X_{n-1} - f(X_{n-1}) / f'(X_{n-1}), n \geq 1$

From equation 1 and equation 3 we get,

Equation 4: $X_n = 0.5 * \{X_{n-1} + N/X_{n-1}\}$

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.

Author: Amar Palacherla
Logic Products Division

APPENDIX A

MPASM B0.54 PAGE 1

```
                ;*****************************************************************
                ;                    8x8 Software Multiplier
                ;                ( Code Efficient : Looped Code )
                ;*****************************************************************
                ;
                ;   The 16 bit result is stored in 2 bytes
                ;
                ; Before calling the subroutine " mpy ", the multiplier should
                ; be loaded in location " mulplr ", and the multiplicand in
                ; " mulcnd " . The 16 bit result is stored in locations
                ; H_byte & L_byte.
                ;
                ;       Performance :
                ;                       Program Memory  :  15 locations
                ;                       # of cycles     :  71
                ;                       Scratch RAM     :   0 locations
                ;
                ;  This routine is optimized for code efficiency ( looped code )
                ;  For time efficiency code refer to "mult8x8F.asm" ( straight line code )
                ;*****************************************************************
                ;
                       LIST    P=16C54
0009            mulcnd  equ     09      ; 8 bit multiplicand
0010            mulplr  equ     10      ; 8 bit multiplier
0012            H_byte  equ     12      ; High byte of the 16 bit result
0013            L_byte  equ     13      ; Low byte of the 16 bit result
0014            count   equ     14      ; loop counter
                ;
                ;

                  include         "mpreg.h"
                ;*************************       PIC16C5X Header ************************
01FF            PIC54   equ     1FFH    ; Define Reset Vectors
01FF            PIC55   equ     1FFH
03FF            PIC56   equ     3FFH
07FF            PIC57   equ     7FFH
                ;
0001            RTCC    equ     1h
0002            PC      equ     2h
0003            STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004            FSR     equ     4h
                ;
0005            Port_A  equ     5h
0006            Port_B  equ     6h      ; I/O Port Assignments
0007            Port_C  equ     7h
                ;
                ;*****************************************************************************
                ;
                ;                       ; STATUS REG. Bits
0000            CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000            C       equ     0h
0001            DCARRY  equ     1h
0001            DC      equ     1h
0002            Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002            Z       equ     2h
0003            P_DOWN  equ     3h
0003            PD      equ     3h
0004            T_OUT   equ     4h
0004            TO      equ     4h
0005            PA0     equ     5h
0006            PA1     equ     6h
0007            PA2     equ     7h
```



```
                        ;
0001                    Same    equ     1h
                        ;
0000                    LSB     equ     0h
0007                    MSB     equ     7h
                        ;
0001                    TRUE    equ     1h
0001                    YES     equ     1h
0000                    FALSE   equ     0h
0000                    NO      equ     0h
                        ;
                        ;*****************************************************************************


                        ;
                        ; ****************************        Begin Multiplier Routine
0000 0072               mpy_S   clrf    H_byte
0001 0073                       clrf    L_byte
0002 0C08                       movlw   8
0003 0034                       movwf   count
0004 0209                       movf    mulcnd,w
0005 0403                       bcf     STATUS,CARRY    ; Clear the carry bit in the status Reg.
0006 0330               loop    rrf     mulplr
0007 0603                       btfsc   STATUS,CARRY
0008 01F2                       addwf   H_byte,Same
0009 0332                       rrf     H_byte,Same
000A 0333                       rrf     L_byte,Same
000B 02F4                       decfsz  count
000C 0A06                       goto    loop
                        ;
000D 0800                       retlw   0
                        ;
                        ;*******************************************************************
                        ;               Test Program
                        ;*******************************************************************
000E 0CFF               main    movlw   0FF
000F 0030                       movwf   mulplr          ; multiplier (in mulplr) = 0FF
0010 0CFF                       movlw   0FF             ; multiplicand(W Reg )   = 0FF
0011 0029                       movwf   mulcnd
                        ;
0012 0900                      call    mpy_S            ; The result 0FF*0FF = FE01 is in locations
                        ;                               ; H_byte & L_byte
                        ;
0013 0A13               self    goto    self
                        ;
                               org     01FF
01FF 0A0E                      goto    main
                        ;
                               END


Errors   :    0
Warnings :    0
```

APPENDIX B

MPASM B0.54 PAGE 1

```
                ;*****************************************************************
                ;                    8x8 Software Multiplier
                ;                ( Fast Version : Straight Line Code )
                ;*****************************************************************
                ;
                ;   The 16 bit result is stored in 2 bytes
                ;
                ; Before calling the subroutine " mpy ", the multiplier should
                ; be loaded in location " mulplr ", and the multiplicand in
                ; " mulcnd " . The 16 bit result is stored in locations
                ; H_byte & L_byte.
                ;
                ;       Performance :
                ;                       Program Memory  :  35 locations
                ;                       # of cycles     :  37
                ;                       Scratch RAM     :   0 locations
                ;
                ;  This routine is optimized for speed efficiency ( straight line code )
                ;  For code efficiency, refer to "mult8x8S.asm" ( looped code )
                ;*****************************************************************
                ;
                       LIST    P=16C54
0009            mulcnd  equ     09      ; 8 bit multiplicand
0010            mulplr  equ     10      ; 8 bit multiplier
0012            H_byte  equ     12      ; High byte of the 16 bit result
0013            L_byte  equ     13      ; Low byte of the 16 bit result
                ;
                ;
                  include         "picreg.h"
                ;************************        PIC16C5X Header ************************
01FF            PIC54   equ     1FFH    ; Define Reset Vectors
01FF            PIC55   equ     1FFH
03FF            PIC56   equ     3FFH
07FF            PIC57   equ     7FFH
                ;
0001            RTCC    equ     1h
0002            PC      equ     2h
0003            STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004            FSR     equ     4h
                ;
0005            Port_A  equ     5h
0006            Port_B  equ     6h      ; I/O Port Assignments
0007            Port_C  equ     7h
                ;
                ;*****************************************************************************
                ;
                ;                       ; STATUS REG. Bits
0000            CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000            C       equ     0h
0001            DCARRY  equ     1h
0001            DC      equ     1h
0002            Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002            Z       equ     2h
0003            P_DOWN  equ     3h
0003            PD      equ     3h
0004            T_OUT   equ     4h
0004            TO      equ     4h
0005            PA0     equ     5h
0006            PA1     equ     6h
0007            PA2     equ     7h
                ;
0001            Same    equ     1h
                ;
```

```
0000                LSB     equ     0h
0007                MSB     equ     7h
                    ;
0001                TRUE    equ     1h
0001                YES     equ     1h
0000                FALSE   equ     0h
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************
                    ;

                    ;
                    ;****   Define a macro for adding & right shifting  **
                    ;
                    mult    MACRO   bit             ; Begin macro
                      btfsc   mulplr,bit
                      addwf   H_byte,Same
                      rrf     H_byte,Same
                      rrf     L_byte,Same
                      ENDM                          ; End of macro
                    ;
                    ; ****************************         Begin Multiplier Routine
0000 0072           mpy_F   clrf    H_byte
0001 0073                   clrf    L_byte
0002 0209                   movf    mulcnd,w        ; move the multiplicand to W reg.
0003 0403                   bcf     STATUS,CARRY    ; Clear the carry bit in the status Reg.
                    mult    0
0004 0610                   btfsc   mulplr,0
0005 01F2                   addwf   H_byte,Same
0006 0332                   rrf     H_byte,Same
0007 0333                   rrf     L_byte,Same

                    mult    1
0008 0630                   btfsc   mulplr,1
0009 01F2                   addwf   H_byte,Same
000A 0332                   rrf     H_byte,Same
000B 0333                   rrf     L_byte,Same

                    mult    2
000C 0650                   btfsc   mulplr,2
000D 01F2                   addwf   H_byte,Same
000E 0332                   rrf     H_byte,Same
000F 0333                   rrf     L_byte,Same

                    mult    3
0010 0670                   btfsc   mulplr,3
0011 01F2                   addwf   H_byte,Same
0012 0332                   rrf     H_byte,Same
0013 0333                   rrf     L_byte,Same

                    mult    4
0014 0690                   btfsc   mulplr,4
0015 01F2                   addwf   H_byte,Same
0016 0332                   rrf     H_byte,Same
0017 0333                   rrf     L_byte,Same
                    mult    5
0018 06B0                   btfsc   mulplr,5
0019 01F2                   addwf   H_byte,Same
001A 0332                   rrf     H_byte,Same
001B 0333                   rrf     L_byte,Same

                    mult    6
001C 06D0                   btfsc   mulplr,6
001D 01F2                   addwf   H_byte,Same
001E 0332                   rrf     H_byte,Same
001F 0333                   rrf     L_byte,Same
```

```
                        mult    7
0020 06F0                       btfsc   mulplr,7
0021 01F2                       addwf   H_byte,Same
0022 0332                       rrf     H_byte,Same
0023 0333                       rrf     L_byte,Same

                        ;
0024 0800                       retlw   0
                        ;
                        ;*****************************************************************
                        ;               Test Program
                        ;*****************************************************************
0025 0CFF               main    movlw   0FF
0026 0030                       movwf   mulplr          ; multiplier (in mulplr)     = 0FF
0027 0CFF                       movlw   0FF
0028 0029                       movwf   mulcnd          ; multiplicand(in mulcnd )   = 0FF
                        ;
0029 0900                       call    mpy_F           ; The result 0FF*0FF = FE01 is in locations
                        ;                               ; H_byte & L_byte
                        ;
002A 0A2A               self    goto    self
                        ;
                                org     01FF
01FF 0A25                       goto    main
                        ;
                                END


Errors   :    0
Warnings :    0
```

5

APPENDIX C: DOUBLE PRECISION MULTIPLICATION LISTING (LOOPED)

```
MPASM B0.54                                                          PAGE  1

                    ;*****************************************************************
                    ;                    Double Precision Multiplication
                    ;
                    ;               ( Optimized for Code Size : Looped Code )
                    ;
                    ;*****************************************************************;
                    ;   Multiplication : ACCb(16 bits) * ACCa(16 bits) -> ACCb,ACCc ( 32 bits )
                    ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
                    ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
                    ;      (c) CALL D_mpy
                    ;      (d) The 32 bit result is in location ( ACCbHI,ACCbLO,ACCcHI,ACCcLO )
                    ;
                    ;   Performance :
                    ;               Program Memory  :       033
                    ;               Clock Cycles    :       333
                    ;
                    ;       Note : The above timing is the worst case timing, when the
                    ;               register ACCb = FFFF. The speed may be improved if
                    ;               the register ACCb contains a number ( out of the two
                    ;               numbers ) with less number of 1s.
                    ;               The performance specs are for Unsigned arithmetic ( i.e,
                    ;               with “SIGNED equ  FALSE “).
                    ;
                    ;               The performance specs are for Unsigned arithmetic ( i.e,
                    ;               with “SIGNED equ  FALSE “).
                    ;
                    ;*****************************************************************;
                    ;
                           LIST    P=16C54
0010                ACCaLO  equ     10
0011                ACCaHI  equ     11
0012                ACCbLO  equ     12
0013                ACCbHI  equ     13
0014                ACCcLO  equ     14
0015                ACCcHI  equ     15
0016                ACCdLO  equ     16
0017                ACCdHI  equ     17
0018                temp    equ     18
0019                sign    equ     19
                    ;
                      include “picreg.h”
                    ;*************************       PIC16C5X Header ************************
01FF                PIC54   equ     1FFH    ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1h
0002                PC      equ     2h
0003                STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004                FSR     equ     4h
                    ;
0005                Port_A  equ     5h
0006                Port_B  equ     6h      ; I/O Port Assignments
0007                Port_C  equ     7h
                    ;
                    ;*****************************************************************************
                    ;
                                            ; STATUS REG. Bits
0000                CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000                C       equ     0h
0001                DCARRY  equ     1h
0001                DC      equ     1h
0002                Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2h
```

```
0003                P_DOWN  equ     3h
0003                PD      equ     3h
0004                T_OUT   equ     4h
0004                TO      equ     4h
0005                PA0     equ     5h
0006                PA1     equ     6h
0007                PA2     equ     7h
                    ;
0001                Same    equ     1h
                    ;
0000                LSB     equ     0h
0007                MSB     equ     7h
                    ;
0001                TRUE    equ     1h
0001                YES     equ     1h
0000                FALSE   equ     0h
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************


                          org     0
                    ;*****************************************************************
0001                SIGNED  equ     TRUE            ; 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 Subtraction ( ACCb - ACCa -> ACCb )
                    ;
0000 0210           D_add   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0001 01F2                   addwf   ACCbLO          ;add lsb
0002 0603                   btfsc   STATUS,CARRY    ;add in carry
0003 02B3                   incf    ACCbHI
0004 0211                   movf    ACCaHI,w
0005 01F3                   addwf   ACCbHI          ;add msb
0006 0800                   retlw   0
                    ;*****************************************************************
                    ;               Double Precision Multiply ( 16x16 -> 32 )
                    ;         ( ACCb*ACCa -> ACCb,ACCc ) : 32 bit output with high word
                    ;  in ACCb ( ACCbHI,ACCbLO ) and low word in ACCc ( ACCcHI,ACCcLO ).
                    ;
                    D_mpyS                           ;results in ACCb(16 msb's) and ACCc(16
                    ;
                        IF   SIGNED
0007 0930                 CALL    S_SIGN
                        ENDIF
                    ;
0008 0921                   call    setup
0009 0337           mloop   rrf     ACCdHI          ;rotate d right
000A 0336                   rrf     ACCdLO
000B 0603                   btfsc   STATUS,CARRY    ;need to add?
000C 0900                   call    D_add
000D 0333                   rrf     ACCbHI
000E 0332                   rrf     ACCbLO
000F 0335                   rrf     ACCcHI
0010 0334                   rrf     ACCcLO
0011 02F8                   decfsz  temp            ;loop until all bits checked
0012 0A09                   goto    mloop
```

5

```
                        ;
                           IF   SIGNED
0013 07F9                       btfss   sign,MSB
0014 0800                       retlw   0
0015 0274                       comf    ACCcLO          ; negate ACCa ( -ACCa -> ACCa )
0016 02B4                       incf    ACCcLO
0017 0643                       btfsc   STATUS,Z_bit
0018 00F5                       decf    ACCcHI
0019 0275                       comf    ACCcHI
001A 0643                       btfsc   STATUS,Z_bit
001B 0272                 neg_B comf    ACCbLO          ; negate ACCb
001C 02B2                       incf    ACCbLO
001D 0643                       btfsc   STATUS,Z_bit
001E 00F3                       decf    ACCbHI
001F 0273                       comf    ACCbHI
0020 0800                       retlw   0
                           ELSE
                               retlw   0
                           ENDIF
                        ;
                        ;*****************************************************************
                        ;
0021 0C10                 setup movlw   .16             ; for 16 shifts
0022 0038                       movwf   temp
0023 0213                       movf    ACCbHI,w        ;move ACCb to ACCd
0024 0037                       movwf   ACCdHI
0025 0212                       movf    ACCbLO,w
0026 0036                       movwf   ACCdLO
0027 0073                       clrf    ACCbHI
0028 0072                       clrf    ACCbLO
0029 0800                       retlw   0
                        ;
                        ;*****************************************************************
                        ;
002A 0270                  neg_A   comf    ACCaLO          ; negate ACCa ( -ACCa -> ACCa )
002B 02B0                          incf    ACCaLO
002C 0643                          btfsc   STATUS,Z_bit
002D 00F1                          decf    ACCaHI
002E 0271                          comf    ACCaHI
002F 0800                          retlw   0
                        ;
                        ;*****************************************************************
                        ;  Assemble this section only if Signed Arithmetic Needed
                        ;
                            IF    SIGNED
                        ;
0030 0211                 S_SIGN  movf    ACCaHI,W
0031 0193                         xorwf   ACCbHI,W
0032 0039                         movwf   sign
0033 07F3                         btfss   ACCbHI,MSB      ; if MSB set go & negate ACCb
0034 0A3A                         goto    chek_A
                        ;
0035 0272                         comf    ACCbLO          ; negate ACCb
0036 02B2                         incf    ACCbLO
0037 0643                         btfsc   STATUS,Z_bit
0038 00F3                         decf    ACCbHI
0039 0273                         comf    ACCbHI
                        ;
003A 07F1                 chek_A  btfss   ACCaHI,MSB      ; if MSB set go & negate ACCa
003B 0800                         retlw   0
003C 0A2A                         goto    neg_A
                        ;
                            ENDIF
```

```
                    ;
                    ;*****************************************************************
                    ;                          Test Program
                    ;*****************************************************************
                    ;    Load constant values to ACCa & ACCb for testing
                    ;
003D 0C01           main    movlw   1
003E 0031                   movwf   ACCaHI
003F 0CFF                   movlw   0FF             ; loads ACCa = 01FF
0040 0030                   movwf   ACCaLO
                    ;
0041 0C7F                   movlw   07F
0042 0033                   movwf   ACCbHI
0043 0CFF                   movlw   0FF             ; loads ACCb = 7FFF
0044 0032                   movwf   ACCbLO
                    ;
0045 0907                   call    D_mpyS          ; Here (ACCb,ACCc) = 00FF 7E01
                    ;
0046 0A46           self    goto    self
                    ;
                            org     PIC54
01FF 0A3D                   goto    main
                            END
                    ;*****************************************************************

Errors   :    0
Warnings :    0
```

5

APPENDIX D: DOUBLE PRECISION MULTIPLICATION LISTINGS (FAST)

```
MPASM B0.54                                                    PAGE  1

                    ;*****************************************************************
                    ;                       Double Precision Multiplication
                    ;
                    ;               ( Optimized for Speed : straight Line Code )
                    ;
                    ;*****************************************************************;
                    ;   Multiplication : ACCb(16 bits) * ACCa(16 bits) -> ACCb,ACCc ( 32 bits )
                    ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
                    ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
                    ;      (c) CALL D_mpy
                    ;      (d) The 32 bit result is in location ( ACCbHI,ACCbLO,ACCcHI,ACCcLO )
                    ;
                    ;   Performance :
                    ;               Program Memory  :       240
                    ;               Clock Cycles    :       233
                    ;
                    ;       Note : The above timing is the worst case timing, when the
                    ;               register ACCb = FFFF. The speed may be improved if
                    ;               the register ACCb contains a number ( out of the two
                    ;               numbers ) with less number of 1s.
                    ;
                    ;               The performance specs are for Unsigned arithmetic ( i.e,
                    ;               with “SIGNED equ  FALSE “).

                    ;*****************************************************************;
                    ;
                           LIST    p=16c54
0010                ACCaLO  equ     10
0011                ACCaHI  equ     11
0012                ACCbLO  equ     12
0013                ACCbHI  equ     13
0014                ACCcLO  equ     14
0015                ACCcHI  equ     15
0016                ACCdLO  equ     16
0017                ACCdHI  equ     17
0018                temp    equ     18
0019                sign    equ     19
                    ;
                      include “mpreg.h”
                    ;*************************        PIC16C5X Header ************************
01FF                PIC54   equ     1FFH    ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1h
0002                PC      equ     2h
0003                STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004                FSR     equ     4h
                    ;
0005                Port_A  equ     5h
0006                Port_B  equ     6h      ; I/O Port Assignments
0007                Port_C  equ     7h
                    ;
                    ;*****************************************************************************
                    ;
                    ;                       ; STATUS REG. Bits
0000                CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000                C       equ     0h
0001                DCARRY  equ     1h
0001                DC      equ     1h
0002                Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2h
0003                P_DOWN  equ     3h
```

```
0003              PD      equ     3h
0004              T_OUT   equ     4h
0004              TO      equ     4h
0005              PA0     equ     5h
0006              PA1     equ     6h
0007              PA2     equ     7h
                  ;
0001              Same    equ     1h
                  ;
0000              LSB     equ     0h
0007              MSB     equ     7h
                  ;
0001              TRUE    equ     1h
0001              YES     equ     1h
0000              FALSE   equ     0h
0000              NO      equ     0h
                  ;
                  ;*****************************************************************************


                        org     0
                  ;*****************************************************************
0000              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.
                  ;*****************************************************************
                  ;       multiplication macro
                  ;
                  mulMac  MACRO
                    LOCAL   NO_ADD
                  ;
                    rrf     ACCdHI              ;rotate d right
                    rrf     ACCdLO
                    btfss   STATUS,CARRY        ;need to add?
                    goto    NO_ADD              ; no addition necessary
                    movf    ACCaLO,w            ; Addition ( ACCb + ACCa -> ACCb )
                    addwf   ACCbLO              ;add lsb
                    btfsc   STATUS,CARRY        ;add in carry
                    incf    ACCbHI
                    movf    ACCaHI,w
                    addwf   ACCbHI              ;add msb
                  NO_ADD  rrf     ACCbHI
                    rrf     ACCbLO
                    rrf     ACCcHI
                    rrf     ACCcLO
                  ;
                    ENDM
                  ;
                  ;*****************************************************************;
                  ;               Double Precision Multiply ( 16x16 -> 32 )
                  ;         ( ACCb*ACCa -> ACCb,ACCc ) : 32 bit output with high word
                  ;  in ACCb ( ACCbHI,ACCbLO ) and low word in ACCc ( ACCcHI,ACCcLO ).
                  ;
                  D_mpyF                        ;results in ACCb(16 msb's) and ACCc(16 lsb's)
                  ;
                      IF   SIGNED
                      CALL    S_SIGN
                      ENDIF
                  ;
0000 09E2                 call    setup
                  ;
```

5

```
                        ; use the mulMac macro 16 times
                        ;
                          mulMac
                               LOCAL   NO_ADD
                        ;
0001 0337                       rrf     ACCdHI          ;rotate d right
0002 0336                       rrf     ACCdLO
0003 0703                       btfss   STATUS,CARRY    ;need to add?
0004 0A0B                       goto    NO_ADD          ; no addition necessary
0005 0210                       movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0006 01F2                       addwf   ACCbLO          ;add lsb
0007 0603                       btfsc   STATUS,CARRY    ;add in carry
0008 02B3                       incf    ACCbHI
0009 0211                       movf    ACCaHI,w
000A 01F3                       addwf   ACCbHI            ;add msb
000B 0333                 NO_ADD  rrf     ACCbHI
000C 0332                       rrf     ACCbLO
000D 0335                       rrf     ACCcHI
000E 0334                       rrf     ACCcLO
                        ;

                          mulMac
                               LOCAL   NO_ADD
                        ;
000F 0337                       rrf     ACCdHI          ;rotate d right
0010 0336                       rrf     ACCdLO
0011 0703                       btfss   STATUS,CARRY    ;need to add?
0012 0A19                       goto    NO_ADD          ; no addition necessary
0013 0210                       movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0014 01F2                       addwf   ACCbLO          ;add lsb
0015 0603                       btfsc   STATUS,CARRY    ;add in carry
0016 02B3                       incf    ACCbHI
0017 0211                       movf    ACCaHI,w
0018 01F3                       addwf   ACCbHI          ;add msb
0019 0333                 NO_ADD  rrf     ACCbHI
001A 0332                       rrf     ACCbLO
001B 0335                       rrf     ACCcHI
001C 0334                       rrf     ACCcLO
                        ;

                          mulMac
                               LOCAL   NO_ADD
                        ;
001D 0337                       rrf     ACCdHI          ;rotate d right
001E 0336                       rrf     ACCdLO
001F 0703                       btfss   STATUS,CARRY    ;need to add?
0020 0A27                       goto    NO_ADD          ; no addition necessary
0021 0210                       movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0022 01F2                       addwf   ACCbLO          ;add lsb
0023 0603                       btfsc   STATUS,CARRY    ;add in carry
0024 02B3                       incf    ACCbHI
0025 0211                       movf    ACCaHI,w
0026 01F3                       addwf   ACCbHI          ;add msb
0027 0333                 NO_ADD  rrf     ACCbHI
0028 0332                       rrf     ACCbLO
0029 0335                       rrf     ACCcHI
002A 0334                       rrf     ACCcLO
                        ;
```

.

```
                      mulMac
                           LOCAL   NO_ADD
                    ;
002B 0337                  rrf     ACCdHI          ;rotate d right
002C 0336                  rrf     ACCdLO
002D 0703                  btfss   STATUS,CARRY    ;need to add?
002E 0A35                  goto    NO_ADD          ; no addition necessary
002F 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0030 01F2                  addwf   ACCbLO          ;add lsb
0031 0603                  btfsc   STATUS,CARRY    ;add in carry
0032 02B3                  incf    ACCbHI
0033 0211                  movf    ACCaHI,w
0034 01F3                  addwf   ACCbHI          ;add msb
0035 0333           NO_ADD rrf     ACCbHI
0036 0332                  rrf     ACCbLO
0037 0335                  rrf     ACCcHI
0038 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
0039 0337                  rrf     ACCdHI          ;rotate d right
003A 0336                  rrf     ACCdLO
003B 0703                  btfss   STATUS,CARRY    ;need to add?
003C 0A43                  goto    NO_ADD          ; no addition necessary
003D 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
003E 01F2                  addwf   ACCbLO          ;add lsb
003F 0603                  btfsc   STATUS,CARRY    ;add in carry
0040 02B3                  incf    ACCbHI
0041 0211                  movf    ACCaHI,w
0042 01F3                  addwf   ACCbHI          ;add msb
0043 0333           NO_ADD rrf     ACCbHI
0044 0332                  rrf     ACCbLO
0045 0335                  rrf     ACCcHI
0046 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
0047 0337                  rrf     ACCdHI          ;rotate d right
0048 0336                  rrf     ACCdLO
0049 0703                  btfss   STATUS,CARRY    ;need to add?
004A 0A51                  goto    NO_ADD          ; no addition necessary
004B 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
004C 01F2                  addwf   ACCbLO          ;add lsb
004D 0603                  btfsc   STATUS,CARRY    ;add in carry
004E 02B3                  incf    ACCbHI
004F 0211                  movf    ACCaHI,w
0050 01F3                  addwf   ACCbHI          ;add msb
0051 0333           NO_ADD rrf     ACCbHI
0052 0332                  rrf     ACCbLO
0053 0335                  rrf     ACCcHI
0054 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
0055 0337                  rrf     ACCdHI          ;rotate d right
0056 0336                  rrf     ACCdLO
0057 0703                  btfss   STATUS,CARRY    ;need to add?
0058 0A5F                  goto    NO_ADD          ; no addition necessary
0059 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
005A 01F2                  addwf   ACCbLO          ;add lsb
005B 0603                  btfsc   STATUS,CARRY    ;add in carry
005C 02B3                  incf    ACCbHI
005D 0211                  movf    ACCaHI,w
005E 01F3                  addwf   ACCbHI            ;add msb
```

```
005F 0333           NO_ADD  rrf     ACCbHI
0060 0332                   rrf     ACCbLO
0061 0335                   rrf     ACCcHI
0062 0334                   rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
0063 0337                   rrf     ACCdHI          ;rotate d right
0064 0336                   rrf     ACCdLO
0065 0703                   btfss   STATUS,CARRY    ;need to add?
0066 0A6D                   goto    NO_ADD          ; no addition necessary
0067 0210                   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0068 01F2                   addwf   ACCbLO          ;add lsb
0069 0603                   btfsc   STATUS,CARRY    ;add in carry
006A 02B3                   incf    ACCbHI
006B 0211                   movf    ACCaHI,w
006C 01F3                   addwf   ACCbHI          ;add msb
006D 0333           NO_ADD  rrf     ACCbHI
006E 0332                   rrf     ACCbLO
006F 0335                   rrf     ACCcHI
0070 0334                   rrf     ACCcLO
                    ;

                      mulMac
                           LOCAL   NO_ADD
                    ;
0071 0337                   rrf     ACCdHI          ;rotate d right
0072 0336                   rrf     ACCdLO
0073 0703                   btfss   STATUS,CARRY    ;need to add?
0074 0A7B                   goto    NO_ADD          ; no addition necessary
0075 0210                   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0076 01F2                   addwf   ACCbLO          ;add lsb
0077 0603                   btfsc   STATUS,CARRY    ;add in carry
0078 02B3                   incf    ACCbHI
0079 0211                   movf    ACCaHI,w
007A 01F3                   addwf   ACCbHI          ;add msb
007B 0333           NO_ADD  rrf     ACCbHI
007C 0332                   rrf     ACCbLO
007D 0335                   rrf     ACCcHI
007E 0334                   rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
007F 0337                   rrf     ACCdHI          ;rotate d right
0080 0336                   rrf     ACCdLO
0081 0703                   btfss   STATUS,CARRY    ;need to add?
0082 0A89                   goto    NO_ADD          ; no addition necessary
0083 0210                   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0084 01F2                   addwf   ACCbLO          ;add lsb
0085 0603                   btfsc   STATUS,CARRY    ;add in carry
0086 02B3                   incf    ACCbHI
0087 0211                   movf    ACCaHI,w
0088 01F3                   addwf   ACCbHI          ;add msb
0089 0333           NO_ADD  rrf     ACCbHI
008A 0332                   rrf     ACCbLO
008B 0335                   rrf     ACCcHI
008C 0334                   rrf     ACCcLO
                    ;
```

```
                      mulMac
                           LOCAL   NO_ADD
                    ;
008D 0337                  rrf     ACCdHI          ;rotate d right
008E 0336                  rrf     ACCdLO
008F 0703                  btfss   STATUS,CARRY    ;need to add?
0090 0A97                  goto    NO_ADD          ; no addition necessary
0091 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
0092 01F2                  addwf   ACCbLO          ;add lsb
0093 0603                  btfsc   STATUS,CARRY    ;add in carry
0094 02B3                  incf    ACCbHI
0095 0211                  movf    ACCaHI,w
0096 01F3                  addwf   ACCbHI          ;add msb
0097 0333           NO_ADD rrf     ACCbHI
0098 0332                  rrf     ACCbLO
0099 0335                  rrf     ACCcHI
009A 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
009B 0337                  rrf     ACCdHI          ;rotate d right
009C 0336                  rrf     ACCdLO
009D 0703                  btfss   STATUS,CARRY    ;need to add?
009E 0AA5                  goto    NO_ADD          ; no addition necessary
009F 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
00A0 01F2                  addwf   ACCbLO          ;add lsb
00A1 0603                  btfsc   STATUS,CARRY    ;add in carry
00A2 02B3                  incf    ACCbHI
00A3 0211                  movf    ACCaHI,w
00A4 01F3                  addwf   ACCbHI          ;add msb
00A5 0333           NO_ADD rrf     ACCbHI
00A6 0332                  rrf     ACCbLO
00A7 0335                  rrf     ACCcHI
00A8 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
00A9 0337                  rrf     ACCdHI          ;rotate d right
00AA 0336                  rrf     ACCdLO
00AB 0703                  btfss   STATUS,CARRY    ;need to add?
00AC 0AB3                  goto    NO_ADD          ; no addition necessary
00AD 0210                  movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
00AE 01F2                  addwf   ACCbLO          ;add lsb
00AF 0603                  btfsc   STATUS,CARRY    ;add in carry
00B0 02B3                  incf    ACCbHI
00B1 0211                  movf    ACCaHI,w
00B2 01F3                  addwf   ACCbHI          ;add msb
00B3 0333           NO_ADD rrf     ACCbHI
00B4 0332                  rrf     ACCbLO
00B5 0335                  rrf     ACCcHI
00B6 0334                  rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
00B7 0337                  rrf     ACCdHI          ;rotate d right
00B8 0336                  rrf     ACCdLO
00B9 0703                  btfss   STATUS,CARRY    ;need to add?
00BA 0AC1                  goto    NO_ADD          ;no addition necessary
00BB 0210                  movf    ACCaLO,w        ;Addition ( ACCb + ACCa -> ACCb )
00BC 01F2                  addwf   ACCbLO          ;add lsb
00BD 0603                  btfsc   STATUS,CARRY    ;add in carry
00BE 02B3                  incf    ACCbHI
00BF 0211                  movf    ACCaHI,w
00C0 01F3                  addwf   ACCbHI          ;add msb
```

```
00C1 0333           NO_ADD  rrf     ACCbHI
00C2 0332                   rrf     ACCbLO
00C3 0335                   rrf     ACCcHI
00C4 0334                   rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
00C5 0337                   rrf     ACCdHI          ;rotate d right
00C6 0336                   rrf     ACCdLO
00C7 0703                   btfss   STATUS,CARRY    ;need to add?
00C8 0ACF                   goto    NO_ADD          ; no addition necessary
00C9 0210                   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
00CA 01F2                   addwf   ACCbLO          ;add lsb
00CB 0603                   btfsc   STATUS,CARRY    ;add in carry
00CC 02B3                   incf    ACCbHI
00CD 0211                   movf    ACCaHI,w
00CE 01F3                   addwf   ACCbHI          ;add msb
00CF 0333           NO_ADD  rrf     ACCbHI
00D0 0332                   rrf     ACCbLO
00D1 0335                   rrf     ACCcHI
00D2 0334                   rrf     ACCcLO
                    ;
                      mulMac
                           LOCAL   NO_ADD
                    ;
00D3 0337                   rrf     ACCdHI          ;rotate d right
00D4 0336                   rrf     ACCdLO
00D5 0703                   btfss   STATUS,CARRY    ;need to add?
00D6 0ADD                   goto    NO_ADD          ; no addition necessary
00D7 0210                   movf    ACCaLO,w        ; Addition ( ACCb + ACCa -> ACCb )
00D8 01F2                   addwf   ACCbLO          ;add lsb
00D9 0603                   btfsc   STATUS,CARRY    ;add in carry
00DA 02B3                   incf    ACCbHI
00DB 0211                   movf    ACCaHI,w
00DC 01F3                   addwf   ACCbHI          ;add msb
00DD 0333           NO_ADD  rrf     ACCbHI
00DE 0332                   rrf     ACCbLO
00DF 0335                   rrf     ACCcHI
00E0 0334                   rrf     ACCcLO
                    ;

                    ;
                       IF    SIGNED
                           btfss   sign,MSB
                           retlw   0
                           comf    ACCcLO          ; negate ACCa ( -ACCa -> ACCa )
                           incf    ACCcLO
                           btfsc   STATUS,Z_bit
                           decf    ACCcHI
                           comf    ACCcHI
                           btfsc   STATUS,Z_bit
                    neg_B   comf    ACCbLO          ; negate ACCb
                           incf    ACCbLO
                           btfsc   STATUS,Z_bit
                           decf    ACCbHI
                           comf    ACCbHI
                           retlw   0
                       ELSE
00E1 0800                   retlw   0
                       ENDIF
```

```
                        ;
                        ;*****************************************************************
                        ;
00E2 0C10               setup   movlw   .16             ; for 16 shifts
00E3 0038                       movwf   temp
00E4 0213                       movf    ACCbHI,w        ;move ACCb to ACCd
00E5 0037                       movwf   ACCdHI
00E6 0212                       movf    ACCbLO,w
00E7 0036                       movwf   ACCdLO
00E8 0073                       clrf    ACCbHI
00E9 0072                       clrf    ACCbLO
00EA 0800                       retlw   0
                        ;
                        ;*****************************************************************
                        ;
00EB 0270               neg_A   comf    ACCaLO          ; negate ACCa ( -ACCa -> ACCa )
00EC 02B0                       incf    ACCaLO
00ED 0643                       btfsc   STATUS,Z_bit
00EE 00F1                       decf    ACCaHI
00EF 0271                       comf    ACCaHI
00F0 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_bit
                                decf    ACCbHI
                                comf    ACCbHI
                        ;
                        chek_A  btfss   ACCaHI,MSB      ; if MSB set go & negate ACCa
                                retlw   0
                                goto    neg_A
                        ;
                            ENDIF
                        ;
```

```
                        ;*****************************************************************
                        ;                       Test Program
                        ;*****************************************************************
                        ;
                        ;    Load constant values to ACCa & ACCb for testing
                        ;
00F1 0C01               loadAB  movlw   1
00F2 0031                       movwf   ACCaHI
00F3 0CFF                       movlw   0FF             ; loads ACCa = 01FF
00F4 0030                       movwf   ACCaLO
                        ;
00F5 0C7F                       movlw   07F
00F6 0033                       movwf   ACCbHI
00F7 0CFF                       movlw   0FF             ; loads ACCb = 7FFF
00F8 0032                       movwf   ACCbLO
00F9 0800                       retlw   0
                        ;
00FA 0000               main    nop
                        ;
00FB 09F1                       call    loadAB          ; result of multiplying ACCb*ACCa-
00FC 0900                       call    D_mpyF          ; Here (ACCb,ACCc) = 00FF 7E01
                        ;
00FD 0AFD               self    goto    self
                        ;
                                org     PIC54
01FF 0AFA                       goto    main
                                END
                        ;*****************************************************************


Errors   :    0
Warnings :    0
```

APPENDIX E: DOUBLE PRECISION ADDITION AND SUBTRACTION LISTING

```
MPASM B0.54                                                    PAGE  1

                    ;*****************************************************************
                    ;                 Double Precision Addition & Subtraction
                    ;
                    ;*****************************************************************;
                    ;   Addition :  ACCb(16 bits) + ACCa(16 bits) -> ACCb(16 bits)
                    ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
                    ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
                    ;      (c) CALL D_add
                    ;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
                    ;
                    ;   Performance :
                    ;               Program Memory  :       07
                    ;               Clock Cycles    :       08
                    ;*****************************************************************;
                    ;   Subtraction : ACCb(16 bits) - ACCa(16 bits) -> ACCb(16 bits)
                    ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
                    ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
                    ;      (c) CALL D_sub
                    ;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
                    ;
                    ;   Performance :
                    ;               Program Memory  :       14
                    ;               Clock Cycles    :       17
                    ;*****************************************************************;
                    ;
                           LIST  P=16C54
0010                ACCaLO  equ     10
0011                ACCaHI  equ     11
0012                ACCbLO  equ     12
0013                ACCbHI  equ     13
                    ;
                      include "mpreg.h"
                    ;*************************        PIC16C5X Header ************************
01FF                PIC54   equ     1FFH    ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1h
0002                PC      equ     2h
0003                STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004                FSR     equ     4h
                    ;
0005                Port_A  equ     5h
0006                Port_B  equ     6h      ; I/O Port Assignments
0007                Port_C  equ     7h
                    ;
                    ;*****************************************************************************
                    ;
                    ;                       ; STATUS REG. Bits
0000                CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000                C       equ     0h
0001                DCARRY  equ     1h
0001                DC      equ     1h
0002                Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2h
0003                P_DOWN  equ     3h
0003                PD      equ     3h
0004                T_OUT   equ     4h
0004                TO      equ     4h
0005                PA0     equ     5h
0006                PA1     equ     6h
0007                PA2     equ     7h
                    ;
```

```
0001                Same    equ     1h
                    ;
0000                LSB     equ     0h
0007                MSB     equ     7h
                    ;
0001                TRUE    equ     1h
0001                YES     equ     1h
0000                FALSE   equ     0h
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************
                    '

                           org     0
                    ;*****************************************************************
                    ;         Double Precision Subtraction ( ACCb - ACCa -> ACCb )
                    ;
0000 0908           D_sub   call    neg_A           ; At first negate ACCa; Then add
                    ;
                    ;*****************************************************************
                    ;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
                    ;
0001 0210           D_add   movf    ACCaLO,w
0002 01F2                   addwf   ACCbLO          ;add lsb
0003 0603                   btfsc   STATUS,CARRY    ;add in carry
0004 02B3                   incf    ACCbHI
0005 0211                   movf    ACCaHI,w
0006 01F3                   addwf   ACCbHI          ;add msb
0007 0800                   retlw   0
                    ;
                    ;
0008 0270           neg_A   comf    ACCaLO          ; negate ACCa ( -ACCa -> ACCa )
0009 02B0                   incf    ACCaLO
000A 0643                   btfsc   STATUS,Z_bit
000B 00F1                   decf    ACCaHI
000C 0271                   comf    ACCaHI
000D 0800                   retlw   0
                    ;
                    ;*****************************************************************
                    ;                           Test Program
                    ;*****************************************************************
                    ;    Load constant values to ACCa & ACCb for testing
                    ;
000E 0C01           loadAB  movlw   1
000F 0031                   movwf   ACCaHI
0010 0CFF                   movlw   0FF             ; loads ACCa = 01FF
0011 0030                   movwf   ACCaLO
                    ;
0012 0C7F                   movlw   07F
0013 0033                   movwf   ACCbHI
0014 0CFF                   movlw   0FF             ; loads ACCb = 7FFF
0015 0032                   movwf   ACCbLO
0016 0800                   retlw   0
                    ;
0017 0000           main    nop
                    ;
0018 090E                   call    loadAB          ; result of adding ACCb+ACCa->ACCb
0019 0901                   call    D_add           ; Here Accb = 81FE
                    ;
001A 090E                   call    loadAB          ; result of subtracting ACCb - ACCa->ACCb
001B 0900                   call    D_sub           ; Here Accb = 7E00
                    ;
001C 0A1C           self    goto    self
                    ;
                           org     PIC54
01FF 0A17                  goto    main
                           END
                    ;*************************************************************
Errors   :    0
Warnings :    0
```

APPENDIX F: BCD TO BINARY CONVERSION LISTING

```
MPASM B0.54                                                    PAGE  1

                    ;
                    ;*****************************************************************
                    ;               BCD To Binary Conversion
                    ;
                    ;       This routine converts a 5 digit BCD number to a 16 bit binary
                    ; number.
                    ;       The input 5 digit BCD numbers are asumed to be in locations
                    ; R0, R1 & R2 with R0 containing the MSD in its right most nibble.
                    ;
                    ;       The 16 bit binary number is output in registers H_byte & L_byte
                    ; ( high byte & low byte repectively ).
                    ;
                    ;       The method used for conversion is :
                    ;               input number X = abcde ( the 5 digit BCD number )
                    ;               X = abcde = 10[10[10[10a+b]+c]+d]+e
                    ;
                    ;   Performance :
                    ;               Program Memory  :       30
                    ;               Clock Cycles    :       121
                    ;
                    ;*****************************************************************;
                    ;
                           LIST    P=16C54
0010                H_byte  equ     10
0011                L_byte  equ     11
0012                R0      equ     12              ; RAM Assignments
0013                R1      equ     13
0014                R2      equ     14
                    ;
0015                H_temp  equ     15              ; temporary register
0016                L_temp  equ     16              ; temporary register
                    ;
                    ;
                      INCLUDE         "mpreg.h"
                    ;*************************       PIC16C5X Header ************************
01FF                PIC54   equ     1FFH            ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1h
0002                PC      equ     2h
0003                STATUS  equ     3h              ; F3 Reg is STATUS Reg.
0004                FSR     equ     4h
                    ;
0005                Port_A  equ     5h
0006                Port_B  equ     6h              ; I/O Port Assignments
0007                Port_C  equ     7h
                    ;
                    ;*****************************************************************
                    ;
                    ;                               ; STATUS REG. Bits
0000                CARRY   equ     0h              ; Carry Bit is Bit.0 of F3
0000                C       equ     0h
0001                DCARRY  equ     1h
0001                DC      equ     1h
0002                Z_bit   equ     2h              ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2h
0003                P_DOWN  equ     3h
0003                PD      equ     3h
0004                T_OUT   equ     4h
0004                TO      equ     4h
0005                PA0     equ     5h
0006                PA1     equ     6h
```

```
0007                PA2     equ     7h
                    ;
0001                Same    equ     1h
                    ;
0000                LSB     equ     0h
0007                MSB     equ     7h
                    ;
0001                TRUE    equ     1h
0001                YES     equ     1h
0000                FALSE   equ     0h
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************

                    ;
                    ;
0000 0E0F           mpy10b  andlw   0F
0001 01F1                   addwf   L_byte
0002 0603                   btfsc   STATUS,CARRY
0003 02B0                   incf    H_byte
0004 0403           mpy10a  bcf     STATUS,CARRY    ; multiply by 2
0005 0351                   rlf     L_byte,w
0006 0036                   movwf   L_temp
0007 0350                   rlf     H_byte,w        ; (H_temp,L_temp) = 2*N
0008 0035                   movwf   H_temp
                    ;
0009 0403                   bcf     STATUS,CARRY    ; multiply by 2
000A 0371                   rlf     L_byte
000B 0370                   rlf     H_byte
000C 0403                   bcf     STATUS,CARRY    ; multiply by 2
000D 0371                   rlf     L_byte
000E 0370                   rlf     H_byte
000F 0403                   bcf     STATUS,CARRY    ; multiply by 2
0010 0371                   rlf     L_byte
0011 0370                   rlf     H_byte          ; (H_byte,L_byte) = 8*N
                    ;
0012 0216                   movf    L_temp,w
0013 01F1                   addwf   L_byte
0014 0603                   btfsc   STATUS,CARRY
0015 02B0                   incf    H_byte
0016 0215                   movf    H_temp,w
0017 01F0                   addwf   H_byte
0018 0800                   retlw   0               ; (H_byte,L_byte) = 10*N
                    ;
                    ;
0019 0070           BCDtoB  clrf    H_byte
001A 0212                   movf    R0,w
001B 0E0F                   andlw   0F
001C 0031                   movwf   L_byte
001D 0904                   call    mpy10a          ; result = 10a+b
                    ;
001E 0393                   swapf   R1,w
001F 0900                   call    mpy10b          ; result = 10[10a+b]
                    ;
0020 0213                   movf    R1,w
0021 0900                   call    mpy10b          ; result = 10[10[10a+b]+c]
                    ;
0022 0394                   swapf   R2,w
0023 0900                   call    mpy10b          ; result = 10[10[10[10a+b]+c]+d]
                    ;
0024 0214                   movf    R2,w
0025 0E0F                   andlw   0F
0026 01F1                   addwf   L_byte
0027 0603                   btfsc   STATUS,CARRY
0028 02B0                   incf    H_byte          ; result = 10[10[10[10a+b]+c]+d]+e
0029 0800                   retlw   0               ; BCD to binary conversion done
                    ;
                    ;
```

```
                ;*****************************************************************
                ;               Test Program
                ;*****************************************************************
002A 0C06          main    movlw   06
002B 0032                  movwf   R0      ; Set R0 = 06
002C 0C55                  movlw   55
002D 0033                  movwf   R1      ; Set R1 = 55
002E 0C35                  movlw   35
002F 0034                  movwf   R2      ; Set R2 = 35      ( R0, R1, R2 = 6,55,35 )
                ;
0030 0919                  call    BCDtoB  ; After conversion H_Byte = FF & L_Byte = FF
                ;
0031 0A31          self    goto    self
                ;
                         org     1FF
01FF 0A2A                  goto    main
                ;
                         END


Errors   :    0
Warnings :    0
```

5

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

```
MPASM B0.54                                                    PAGE  1



                        LIST    p=16c54,n=0

                    ;
                    ;*****************************************************************
                    ;                  Binary To BCD Conversion Routine
                    ;
                    ;       This routine converts the 8 bit binary number in the W Register
                    ; to a 2 digit BCD number.
                    ;       The least significant digit is returned in location LSD and
                    ; the most significant digit is returned in location MSD.
                    ;
                    ;   Performance :
                    ;               Program Memory  :       10
                    ;               Clock Cycles    :       81 (worst case when W = 63 Hex )
                    ;                                          ( i.e max Decimal number 99 )
                    ;*****************************************************************
                    ;
0010                LSD     equ     10
0011                MSD     equ     11
                    ;
       INCLUDE        "mpreg.h"
                    ;************************  PIC16C5X Header    ************************
01FF                PIC54   equ     1FFH    ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1
0002                PC      equ     2
0003                STATUS  equ     3       ; F3 Reg is STATUS Reg.
0004                FSR     equ     4
                    ;
0005                Port_A  equ     5
0006                Port_B  equ     6       ; I/O Port Assignments
0007                Port_C  equ     7
                    ;
                    ;*****************************************************************************
                    ;
                    ;                       ; STATUS REG. Bits
0000                CARRY   equ     0       ; Carry Bit is Bit.0 of F3
0000                C       equ     0
0001                DCARRY  equ     1
0001                DC      equ     1
0002                Z_bit   equ     2       ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2
0003                P_DOWN  equ     3
0003                PD      equ     3
0004                T_OUT   equ     4
0004                TO      equ     4
0005                PA0     equ     5
0006                PA1     equ     6
0007                PA2     equ     7
                    ;
0001                Same    equ     1
0000                W       equ     0
                    ;
0000                LSB     equ     0
0007                MSB     equ     7
                    ;
0001                TRUE    equ     1
```

```
0001                YES     equ     1
0000                FALSE   equ     0
0000                NO      equ     0
                    ;
                    ;*****************************************************************************


                    ;
0000 0071           BinBCD  clrf    MSD
0001 0030                   movwf   LSD
0002 0C0A           gtenth  movlw   .10
0003 0090                   subwf   LSD,W
0004 0703                   BTFSS   STATUS,CARRY
0005 0A09                   goto    over
0006 0030                   movwf   LSD
0007 02B1                   incf    MSD
0008 0A02                   goto    gtenth
0009 0800           over    retlw   0
                    ;*****************************************************************
                    ;
000A 0C63           main    movlw   63              ; W reg = 63 Hex
000B 0900                   call    BinBCD          ; after conversion, MSD = 9 & LSD = 9
000C 0A0C           self    goto    self            ; ( 63 Hex = 99 Decimal )
                    ;
                            org     1FF
01FF 0A0A                   goto    main
                    ;
                            END

Errors   :    0
Warnings :    0
```

5

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

```
MPASM B0.54                                                    PAGE  1

                    ;
                    ;*****************************************************************
                    ;                  Binary To BCD Conversion Routine
                    ;      This routine converts a 16 Bit binary Number to a 5 Digit
                    ; BCD Number. This routine is useful since PIC16C55 & PIC16C57
                    ; have  two 8 bit ports and one 4 bit port ( total of 5 BCD digits)
                    ;
                    ;       The 16 bit binary number is input in locations H_byte and
                    ; L_byte with the high byte in H_byte.
                    ;       The 5 digit BCD number is returned in R0, R1 and R2 with R0
                    ; containing the MSD in its right most nibble.
                    ;
                    ;   Performance :
                    ;               Program Memory  :       35
                    ;               Clock Cycles    :       885
                    ;
                    ;*****************************************************************;
                    ;
                          LIST    P=16C54
0016                count  equ      16
0017                temp   equ      17
                    ;
0010                H_byte  equ     10
0011                L_byte  equ     11
0012                R0      equ     12              ; RAM Assignments
0013                R1      equ     13
0014                R2      equ     14
                    ;
                      include         “mpreg.h”
                    ;************************        PIC16C5X Header ************************
01FF                PIC54   equ     1FFH            ; Define Reset Vectors
01FF                PIC55   equ     1FFH
03FF                PIC56   equ     3FFH
07FF                PIC57   equ     7FFH
                    ;
0001                RTCC    equ     1h
0002                PC      equ     2h
0003                STATUS  equ     3h              ; F3 Reg is STATUS Reg.
0004                FSR     equ     4h
                    ;
0005                Port_A  equ     5h
0006                Port_B  equ     6h              ; I/O Port Assignments
0007                Port_C  equ     7h
                    ;
                    ;*****************************************************************************
                    ;
                    ;                               ; STATUS REG. Bits
0000                CARRY   equ     0h              ; Carry Bit is Bit.0 of F3
0000                C       equ     0h
0001                DCARRY  equ     1h
0001                DC      equ     1h
0002                Z_bit   equ     2h              ; Bit 2 of F3 is Zero Bit
0002                Z       equ     2h
0003                P_DOWN  equ     3h
0003                PD      equ     3h
0004                T_OUT   equ     4h
0004                TO      equ     4h
0005                PA0     equ     5h
0006                PA1     equ     6h
0007                PA2     equ     7h
                    ;
0001                Same    equ     1h
                    ;
0000                LSB     equ     0h
```

```
0007                MSB     equ     7h
                    ;
0001                TRUE    equ     1h
0001                YES     equ     1h
0000                FALSE   equ     0h
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************


                    ;
0000 0403           B2_BCD  bcf     STATUS,0                ; clear the carry bit
0001 0C10                   movlw   .16
0002 0036                   movwf   count
0003 0072                   clrf    R0
0004 0073                   clrf    R1
0005 0074                   clrf    R2
0006 0371           loop16  rlf     L_byte
0007 0370                   rlf     H_byte
0008 0374                   rlf     R2
0009 0373                   rlf     R1
000A 0372                   rlf     R0
                    ;
000B 02F6                   decfsz  count
000C 0A0E                   goto    adjDEC
000D 0800                   RETLW   0
                    ;
000E 0C14           adjDEC  movlw   R2
000F 0024                   movwf   FSR
0010 0918                   call    adjBCD
                    ;
0011 0C13                   movlw   R1
0012 0024                   movwf   FSR
0013 0918                   call    adjBCD
                    ;
0014 0C12                   movlw   R0
0015 0024                   movwf   FSR
0016 0918                   call    adjBCD
                    ;
0017 0A06                   goto    loop16
                    ;
0018 0C03           adjBCD  movlw   3
0019 01C0                   addwf   0,W
001A 0037                   movwf   temp
001B 0677                   btfsc   temp,3          ; test if result > 7
001C 0020                   movwf   0
001D 0C30                   movlw   30
001E 01C0                   addwf   0,W
001F 0037                   movwf   temp
0020 06F7                   btfsc   temp,7          ; test if result > 7
0021 0020                   movwf   0               ; save as MSD
0022 0800                   RETLW   0
                    ;
                    ;*****************************************************************
                    ;                   Test Program
                    ;*****************************************************************
0023 0CFF           main    movlw   0FF
0024 0030                   movwf   H_byte
0025 0031                   movwf   L_byte          ; The 16 bit binary number = FFFF
0026 0900                   call    B2_BCD          ; After conversion the Decimal Number
                    ;                               ; in R0,R1,R2 = 06,55,35
                    ;
0027 0A27           self    goto    self
                    ;
                          org     1FF
01FF 0A23                   goto    main
                    ;
                          END
```

APPENDIX I: UNSIGNED BCD ADDITION LISTING

```
MPASM B0.54                                                          PAGE  1

                ;******************  Unsigned BCD Addition   ***************
                ;
                ;       This routine performs a 2 Digit Unsigned BCD Addition
                ; It is assumed that the two BCD numbers to be added are in
                ; locations Num_1 & Num_2. The result is the sum of Num_1+Num_2
                ; and is stored in location Num_2 and the overflow carry is returned
                ; in location Num_1
                ;
                ;   Performance :
                ;               Program Memory  :       25
                ;               Clock Cycles    :       23   ( worst case )
                ;
                ;       Rev 2.0 changed on 7/30/92.
                ;
                ;*****************************************************************;
                ;
                       LIST    P=16C54
0008            Num_1   equ     8       ; Overflow flow carry overwrites Num_1
0008            result  equ     8
                ;
0009            Num_2   equ     9       ; Num_2 + Num_1 overwrites Num_2
0009            O_flow  equ     9
                ;
                ;
                  include         "mpreg.h"
                ;*************************       PIC16C5X Header ************************
01FF            PIC54   equ     1FFH    ; Define Reset Vectors
01FF            PIC55   equ     1FFH
03FF            PIC56   equ     3FFH
07FF            PIC57   equ     7FFH
                ;
0001            RTCC    equ     1h
0002            PC      equ     2h
0003            STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004            FSR     equ     4h
                ;
0005            Port_A  equ     5h
0006            Port_B  equ     6h      ; I/O Port Assignments
0007            Port_C  equ     7h
                ;
                ;*****************************************************************************
                ;
                ;                       ; STATUS REG. Bits
0000            CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000            C       equ     0h
0001            DCARRY  equ     1h
0001            DC      equ     1h
0002            Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002            Z       equ     2h
0003            P_DOWN  equ     3h
0003            PD      equ     3h
0004            T_OUT   equ     4h
0004            TO      equ     4h
0005            PA0     equ     5h
0006            PA1     equ     6h
0007            PA2     equ     7h
                ;
0001             Same    equ     1h
                ;
0000            LSB     equ     0h
0007            MSB     equ     7h
                ;
0001            TRUE    equ     1h
0001            YES     equ     1h
0000            FALSE   equ     0h
```

```
0000                NO      equ     0h
                    ;
                    ;*****************************************************************************


                    ;
0000 0208           BCDAdd  movf    Num_1,w
0001 01E9                   addwf   Num_2           ; do binary addition
0002 0068                   clrf    Num_1
0003 0368                   rlf     Num_1
0004 0623                   btfsc   STATUS,DC       ; Is DC = 0 ?
0005 0A0D                   goto    adjust          ; adjust LSD
0006 0C06                   movlw   6
0007 01E9                   addwf   Num_2           ; Test for LSD > 9 ( by adding 6
0008 0603                   btfsc   STATUS,CARRY
0009 02A8                   incf    Num_1
000A 0723                   btfss   STATUS,DC       ; & checking Digit Carry
000B 00A9                   subwf   Num_2           ; LSD < 9 , so get back original value.
000C 0A0F                   goto    over1
000D 0C06           adjust  movlw   6
000E 01E9                    addwf   Num_2
000F 0C60           over1   movlw   60              ; add 6 to MSD
0010 01E9                   addwf   Num_2
0011 0603                   btfsc   STATUS,CARRY
0012 0A16                   goto    over3
0013 0708                   btfss   Num_1,0
0014 00A9                   subwf   Num_2
0015 0800                   RETLW   0
0016 0C01           over3   movlw   1
0017 0028                   movwf   Num_1
0018 0800                   RETLW   0
                    ;
                    ;*****************************************************************
                    ;               Test Program
                    ;*****************************************************************
0019 0C99           main    movlw   99
001A 0028                   movwf   Num_1           ; Set Num_1 = 99 ( max BCD digit )
001B 0C99                   movlw   99
001C 0029                   movwf   Num_2           ; Set Num_2 = 99
                    ;
001D 0900                   call    BCDAdd          ; After addition, Num_2 = 98
                    ;                               ;  and Num_1 = 01 ( 99+99 = 198 -> max number
)
                    ;
001E 0A1E           self    goto    self
                    ;
                          org     1FF
01FF 0A19                  goto    main
                    ;
                          END




Errors   :    0
Warnings :    0
```


APPENDIX J: UNSIGNED BCD SUBTRACTION LISTING

```
MPASM B0.54                                                          PAGE  1

                ;******************  Unsigned BCD Subtraction   **************
                ;
                ;       This routine performs a 2 Digit Unsigned BCD Subtraction.
                ; It is assumed that the two BCD numbers to be subtracted are in
                ; locations Num_1 & Num_2. The result is the difference of Num_1 & Num_2
                ; ( Num_2 - Num_1) and is stored in location Num_2 and the overflow carry
                ; is returned in location Num_1.
                ;
                ;   Performance :
                ;               Program Memory  :       31
                ;               Clock Cycles    :       21  ( worst case )
                ;
                ;*****************************************************************
                ;
                       LIST    P=16C54
0008            Num_1   equ     8       ; Overflow flow carry overwrites Num_1
0008            result  equ     8
                ;
0009            Num_2   equ     9       ; Num_2 - Num_1 overwrites Num_2
0009            O_flow  equ     9
                ;
                  include         "mpreg.h"
                ;*************************       PIC16C5X Header ************************
01FF            PIC54   equ     1FFH    ; Define Reset Vectors
01FF            PIC55   equ     1FFH
03FF            PIC56   equ     3FFH
07FF            PIC57   equ     7FFH
                ;
0001            RTCC    equ     1h
0002            PC      equ     2h
0003            STATUS  equ     3h      ; F3 Reg is STATUS Reg.
0004            FSR     equ     4h
                ;
0005            Port_A  equ     5h
0006            Port_B  equ     6h      ; I/O Port Assignments
0007            Port_C  equ     7h
                ;
                ;*****************************************************************
                ;
                ;                       ; STATUS REG. Bits
0000            CARRY   equ     0h      ; Carry Bit is Bit.0 of F3
0000            C       equ     0h
0001            DCARRY  equ     1h
0001            DC      equ     1h
0002            Z_bit   equ     2h      ; Bit 2 of F3 is Zero Bit
0002            Z       equ     2h
0003            P_DOWN  equ     3h
0003            PD      equ     3h
0004            T_OUT   equ     4h
0004            TO      equ     4h
0005            PA0     equ     5h
0006            PA1     equ     6h
0007            PA2     equ     7h
                ;
0001            Same    equ     1h
                ;
0000            LSB     equ     0h
0007            MSB     equ     7h
                ;
0001            TRUE    equ     1h
0001            YES     equ     1h
0000            FALSE   equ     0h
0000            NO      equ     0h
                ;
                ;*****************************************************************
```

```
                        ;
0000 0208               BCDSub  movf    Num_1,w
0001 00A9                       subwf   Num_2
0002 0068                       clrf    Num_1
0003 0368                       rlf     Num_1
0004 0723                       btfss   STATUS,DC
0005 0A0C                       goto    adjst1
0006 0769                       btfss   Num_2,3         ; Adjust LSD of Result
0007 0A0E                       goto    Over_1
0008 0649                       btfsc   Num_2,2
0009 0A0C                       goto    adjst1          ; Adjust LSD of Result
000A 0729                       btfss   Num_2,1
000B 0A0E                       goto    Over_1          ; No : Go for MSD
000C 0C06               adjst1  movlw   6
000D 00A9                       subwf   Num_2
000E 0708               Over_1  btfss   Num_1,0         ; CY = 0 ?
000F 0A17                       goto    adjst2          ; Yes, adjust MSD of result
0010 0068                       clrf    Num_1
0011 07E9                       btfss   Num_2,7         ; No, test for MSD >9
0012 0800                       RETLW   0
0013 06C9                       btfsc   Num_2,6
0014 0A17                       goto    adjst2
0015 07A9                       btfss   Num_2,5
0016 0800                       RETLW   0
0017 0C60               adjst2  movlw   60              ; add 6 to MSD
0018 00A9                       subwf   Num_2
0019 0068                       clrf    Num_1
001A 0703                       btfss   STATUS,CARRY    ; test if underflow
001B 0800                       RETLW   0
001C 0C01                       movlw   1
001D 0028                       movwf   Num_1
001E 0800               Over    RETLW   0
                        ;
                        ;*****************************************************************
                        ;               Test Program
                        ;*****************************************************************
001F 0C23               main    movlw   23
0020 0028                       movwf   Num_1           ; Set Num_1 = 23
0021 0C99                       movlw   99
0022 0029                       movwf   Num_2           ; Set Num_2 = 99
0023 0900                       call    BCDSub          ; After subtraction, Num_2 = 76 ( 99-23 )
                        ;                               ;  and Num_1 = 0 ( indicates positive result
                        ;
0024 0C99                       movlw   99
0025 0028                       movwf   Num_1           ; Set Num_1 = 99
0026 0C00                       movlw   0
0027 0029                       movwf   Num_2           ; Set Num_2 = 0
                        ;
0028 0900                       call    BCDSub          ; After subtraction, Num_2 = 1
                        ;                               ;  and Num_1 = 1 ( indicates negative result
                        ;                               ;  -1  <-  ( -99 )
                        ;
0029 0A29               self    goto    self
                        ;
                               org     1FF
01FF 0A1F                       goto    main
                        ;
                               END


Errors   :    0
Warnings :    0
```

APPENDIX K: SQUARE ROOT BY NEWTON-RAPHSON METHOD

```
MPASM 01.01 Released   SQRT_94.ASM   8-4-1994  10:38:49                PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001
                     0002         list    p=16C54, n=0
                     0003 ;*****************************************************************
                     0004 ;
                     0005 ;       Program:    SQRT.ASM
                     0006 ;       Revision:   08-03-94
                     0007 ;
                     0008 ;
                     0009 ;            Square Root By Newton Raphson Method
                     0010 ;
                     0011 ;    This routine computes the square root of a 16 bit number(with
                     0012 ;  low byte in NumLo & high byte in NumHi ). After loading NumLo &
                     0013 ;  NumHi with the desired number whose square root is to be computed,
                     0014 ;  branch to location Sqrt ( by “GOTO  Sqrt” ). “ CALL  Sqrt” cannot
                     0015 ;  be issued because the Sqrt function makes calls to Math routines
                     0016 ;  and the stack is completely used up.
                     0017 ;       The result = sqrt(NumHi,NumLo) is returned in location SqrtLo.
                     0018 ;  The total number of iterations is set to ten. If more iterations
                     0019 ;  are desired, change “LupCnt equ .10” to the desired value. Also,
                     0020 ;  the initial guess value of the square root is given set as
                     0021 ;  input/2 ( in subroutine “init” ). The user may modify this scheme
                     0022 ;  if a better initial approximation value is known. A good initial
                     0023 ;  guess will help the algorithm converge at a faster rate and thus
                     0024 ;  less number of iterations required.
                     0025 ;       Two utility math routines are used by this program : D_divS
                     0026 ;  and D_add. These two routines are listed as seperate routines
                     0027 ;  under double precision Division and double precision addtion
                     0028 ;  respectively.
                     0029 ;
                     0030 ;  Note : If square root of an 8 bit number is desired, it is probably
                     0031 ;         better to have a table look scheme rather than using numerical
                     0032 ;         methods.
                     0033 ;
                     0034 ;
                     0035 ;
                     0036 ;   Performance :
                     0037 ;               Program Memory  :       27  (excluding Math Routines
                     0038 ;                                        D_divS & D_add )
                     0039 ;               Clock Cycles    :       3600 ( approximately )
                     0040 ;
                     0041 ;       To assemble this program, one routine, namely “D_add”
                     0042 ;       must be included into this program. These two routines
                     0043 ;       are listed as a separate program, in files “DBL_ADD.ASM”
                     0044 ;       The D_divS routine was previously found in “DBL_DIVS.ASM”.
                     0045 ;
                     0046 ;*****************************************************************
                     0047     include “picreg.h”
                     0048
                     0049         org     0
                     0050 ;
000A                 0051 LupCnt  equ     .10             ; Number of iterations
                     0052 ;
001D                 0053 NumLo   equ     1D
001E                 0054 NumHi   equ     1E
001F                 0055 count   equ     1F
                     0056 ;
                     0057 ; These are used by the divide routine
                     0058 ;
0010                 0059 ACCaLO  equ     10
0011                 0060 ACCaHI  equ     11
0012                 0061 EXPa    equ     12
```

```
0013                0061 ACCbLO  equ     13
0014                0062 ACCbHI  equ     14
0015                0063 EXPb    equ     15
0016                0064 ACCcLO  equ     16
0017                0065 ACCcHI  equ     17
0018                0066 ACCdLO  equ     18
0019                0067 ACCdHI  equ     19
001A                0068 temp    equ     1A
001B                0069 sign    equ     1B
                    0070 ;
                    0071 ;
0010                0072 SqrtLo  equ     ACCaLO
0011                0073 SqrtHi  equ     ACCaHI
                    0074 ;
                    0075 init
0000 0C0A           0076         movlw   LupCnt
0001 003F           0077         movwf   count
0002 021E           0078         movf    NumHi,W
0003 0031           0079         movwf   SqrtHi
0004 021D           0080         movf    NumLo,W         ; set initial guess root = NUM/2
0005 0030           0081         movwf   SqrtLo
0006 0403           0082         bcf     STATUS,CARRY
0007 0331           0083         rrf     SqrtHi
0008 0330           0084         rrf     SqrtLo
0009 0800           0085         retlw   0
                    0086 ;
000A 0403           0087 div2    bcf     STATUS,CARRY
000B 0314           0088         rrf     ACCbHI,W
000C 0031           0089         movwf   SqrtHi
000D 0313           0090         rrf     ACCbLO,W
000E 0030           0091         movwf   SqrtLo
000F 0800           0092         retlw   0
                    0093 ;
0010 0900           0094 Sqrt    call    init
0011 021D           0095 sloop   movf    NumLo,W
0012 0033           0096         movwf   ACCbLO
0013 021E           0097         movf    NumHi,W
0014 0034           0098         movwf   ACCbHI
                    0099 ;
0015 091B           0100         call    D_divS          ; double precision division
0016 0946           0101         call    D_add           ; double precision addition
                    0102 ;
0017 090A           0103         call    div2
0018 02FF           0104         decfsz  count
0019 0A11           0105         goto    sloop
001A 0A52           0106         goto    over            ; all iterations done
                    0107 ;                               ; branch back to desired location
                    0108 ;
                    0109 ; Double precision routines placed here
                    0110 ;
                    0111 ;*****************************************************************
0000                0112 SIGNED  equ     FALSE           ; Set This To 'TRUE' if the routines
                    0113 ;                               ; for Multiplication & Division needs
                    0114 ;                               ; to be assembled as Signed Integer
                    0115 ;                               ; Routines. If 'FALSE' the above two
                    0116 ;                               ; routines ( D_mpy & D_div ) use
                    0117 ;                               ; unsigned arithmetic.
                    0118 ;*****************************************************************
                    0119 ;                       Double Precision Division
                    0120 ;
                    0121 ;               ( Optimized for Code Size : Looped Code )
                    0122 ;
                    0123 ;*****************************************************************;
                    0124 ;   Division : ACCb(16 bits) / ACCa(16 bits) -> ACCb(16 bits) with
                    0125 ;                                              Remainder in ACCc (16 bits)
                    0126 ;      (a) Load the Denominator in location ACCaHI & ACCaLO ( 16 bits )
                    0127 ;      (b) Load the Numerator in location ACCbHI & ACCbLO ( 16 bits )
                    0128 ;      (c) CALL D_div
                    0129 ;      (d) The 16 bit result is in location ACCbHI & ACCbLO
```

```
                     0130 ;      (e) The 16 bit Remainder is in locations ACCcHI & ACCcLO
                     0131 ;
                     0132 ;   Performance :
                     0133 ;               Program Memory  :       037
                     0134 ;               Clock Cycles    :       310
                     0135 ;
                     0136 ;        NOTE :
                     0137 ;               The performance specs are for Unsigned arithmetic ( i.e,
                     0138 ;               with “SIGNED equ  FALSE “).
                     0139 ;
                     0140 ;*****************************************************************;
                     0141 ;
                     0142 ;       Double Precision Divide ( 16/16 -> 16 )
                     0143 ;
                     0144 ;         ( ACCb/ACCa -> ACCb with remainder in ACCc ) : 16 bit output
                     0145 ; with Quotiont in ACCb (ACCbHI,ACCbLO) and Remainder in ACCc (ACCcHI,ACCcLO).
                     0146 ;
                     0147 ;   NOTE  :  Before calling this routine, the user should make sure that
                     0148 ;            the Numerator(ACCb) is greater than Denominator(ACCa). If
                     0149 ;            the case is not true, the user should scale either Numerator
                     0150 ;            or Denominator or both such that Numerator is greater than
                     0151 ;            the Denominator.
                     0152 ;
                     0153 ;
                     0154 D_divS
                     0155 ;
                     0156      IF   SIGNED
                     0157         CALL    S_SIGN
                     0158      ENDIF
                     0159 ;
001B 0937            0160         call    setup
001C 0077            0161         clrf    ACCcHI
001D 0076            0162         clrf    ACCcLO
001E 0403            0163 dloop   bcf     STATUS,CARRY
001F 0378            0164         rlf     ACCdLO
0020 0379            0165         rlf     ACCdHI
0021 0376            0166         rlf     ACCcLO
0022 0377            0167         rlf     ACCcHI
0023 0211            0168         movf    ACCaHI,w
0024 0097            0169         subwf   ACCcHI,w          ;check if a>c
0025 0743            0170         btfss   STATUS,Z_bit
0026 0A29            0171         goto    nochk
0027 0210            0172         movf    ACCaLO,w
0028 0096            0173         subwf   ACCcLO,w        ;if msb equal then check lsb
0029 0703            0174 nochk   btfss   STATUS,CARRY    ;carry set if c>a
002A 0A32            0175         goto    nogo
002B 0210            0176         movf    ACCaLO,w        ;c-a into c
002C 00B6            0177         subwf   ACCcLO
002D 0703            0178         btfss   STATUS,CARRY
002E 00F7            0179         decf    ACCcHI
002F 0211            0180         movf    ACCaHI,w
0030 00B7            0181         subwf   ACCcHI
0031 0503            0182         bsf     STATUS,CARRY    ;shift a 1 into b (result)
0032 0373            0183 nogo    rlf     ACCbLO
0033 0374            0184         rlf     ACCbHI
0034 02FA            0185         decfsz  temp            ;loop untill all bits checked
0035 0A1E            0186         goto    dloop
                     0187 ;
                     0188     IF    SIGNED
                     0189         btfss   sign,MSB        ; check sign if negative
                     0190         retlw   0
                     0191         goto    neg_B          ; negate ACCa ( -ACCa -> ACCa )
                     0192     ELSE
0036 0800            0193         retlw   0
                     0194     ENDIF
                     0195 ;
                     0196 ;*****************************************************************
                     0197 ;
0037 0C10            0198 setup   movlw   .16             ; for 16 shifts
```

```
0038 003A           0199         movwf   temp
0039 0214           0200         movf    ACCbHI,w          ;move ACCb to ACCd
003A 0039           0201         movwf   ACCdHI
003B 0213           0202         movf    ACCbLO,w
003C 0038           0203         movwf   ACCdLO
003D 0074           0204         clrf    ACCbHI
003E 0073           0205         clrf    ACCbLO
003F 0800           0206         retlw   0
                    0207 ;
                    0208 ;*****************************************************************
                    0209 ;
0040 0270           0210 neg_A   comf    ACCaLO          ; negate ACCa ( -ACCa -> ACCa )
0041 02B0           0211         incf    ACCaLO
0042 0643           0212         btfsc   STATUS,Z_bit
0043 00F1           0213         decf    ACCaHI
0044 0271           0214         comf    ACCaHI
0045 0800           0215         retlw   0
                    0216 ;
                    0217 ;*****************************************************************
                    0218 ;  Assemble this section only if Signed Arithmetic Needed
                    0219 ;
                    0220      IF    SIGNED
                    0221 ;
                    0222 S_SIGN  movf    ACCaHI,W
                    0223         xorwf   ACCbHI,W
                    0224         movwf   sign
                    0225         btfss   ACCbHI,MSB        ; if MSB set go & negate ACCb
                    0226         goto    chek_A
                    0227 ;
                    0228         comf    ACCbLO          ; negate ACCb
                    0229         incf    ACCbLO
                    0230         btfsc   STATUS,Z_bit
                    0231         decf    ACCbHI
                    0232         comf    ACCbHI
                    0233 ;
                    0234 chek_A  btfss   ACCaHI,MSB        ; if MSB set go & negate ACCa
                    0235         retlw   0
                    0236         goto    neg_A
                    0237 ;
                    0238      ENDIF
                    0239 ;
                    0240     include  <D_add.asm>
                    0001 ;*****************************************************************
                    0002 ;                 Double Precision Addition & Subtraction
                    0003 ;
                    0004 ;         Program:   D_ASM.ASM
                    0005 ;         Revision:  08-03-94
                    0006 ;
                    0007 ;*****************************************************************;
                    0008 ;   Addition :  ACCb(16 bits) + ACCa(16 bits) -> ACCb(16 bits)
                    0009 ;      (a) Load the 1st operand in location ACCaLO & ACCaHI ( 16 bits )
                    0010 ;      (b) Load the 2nd operand in location ACCbLO & ACCbHI ( 16 bits )
                    0011 ;      (c) CALL D_add
                    0012 ;      (d) The result is in location ACCbLO & ACCbHI ( 16 bits )
                    0013 ;
                    0014 ;   Performance :
                    0015 ;               Program Memory  :       07
                    0016 ;               Clock Cycles    :       08
                    0017 ;*****************************************************************
                    0018 ;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
                    0019 ;
0046 0210           0020 D_add   movf    ACCaLO,w
0047 01F3           0021         addwf   ACCbLO          ;add lsb
0048 0603           0022         btfsc   STATUS,CARRY    ;add in carry
0049 02B4           0023         incf    ACCbHI
004A 0211           0024         movf    ACCaHI,w
004B 01F4           0025         addwf   ACCbHI            ;add msb
004C 0800           0026         retlw   0
                    0027 ;
```

```
                    0028
                    0029
                    0240
                    0241 ;
                    0242 ;*************************************************************
                    0243 ;               Test Program
                    0244 ;*************************************************************
                    0245 ;
004D 0CF3           0246 main    movlw   0F3
004E 003E           0247         movwf   NumHi
004F 0CF6           0248         movlw   0F6       ; Set input test number = 62454
0050 003D           0249         movwf   NumLo     ;  = F3F6h
                    0250 ;
0051 0A10           0251         goto    Sqrt      ; cannot use CALL : Math routines
                    0252 ;                         ; use up all the stack.
0052 0000           0253 over    nop               ; all iterations done
                    0254 ;
0053 0A53           0255 self    goto    self      ; result = 00F9h = 249
                    0256 ;                         ; exact sqrt(62454) = 249.9
                    0257 ;
                    0258         org     PIC54
01FF 0A4D           0259         goto    main
                    0260 ;
                    0261         END
                    0262


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

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXX------- --------- ---------

0180 : --------- --------- --------- ---------
01C0 : --------- --------- --------- --------X

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

APPENDIX L: INCLUDE FILE FOR FIXED POINT ROUTINE

```
               processor      16C71

;     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 _RP0          STATUS,5
               #define _PA0          STATUS,5
               #define _RP1          STATUS,6
               #define _PA1          STATUS,6
               #define _IRP          STATUS,7
               #define _PA2          STATUS,7
```

5

APPENDIX M: 16/8 FIXED POINT DIVIDE ROUTINES

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

APPENDIX N: 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

U0K55LL         RLF             ACC+B1
                RLF             ACC+B0

                DECFSZ          LOOPCOUNT
                GOTO            LOOPU1515

                BTFSC           ACC+B1,LSB
                GOTO            U0K55L
                MOVF            BARG+B1,W
                ADDWF           REM+B1
                MOVF            BARG+B0,W
                BTFSC           _C
                INCF            BARG+B0,W
                ADDWF           REM+B0
U0K55L

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

APPENDIX O: 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

U0K26L8         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            U0K26LB

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

U0K26LB         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            U0K26L16
```

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

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

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

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

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

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

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

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95