AN542

Implementation of Fast Fourier Transforms

Author: *Amar Palacherla*
Microchip Technology Inc.

INTRODUCTION

Fourier transforms are one of the fundamental operations in signal processing. In digital computations, Discrete Fourier Transforms (DFT) are used to describe, represent, and analyze discrete-time signals. However, direct implementation of DFT is computationally very inefficient. Of the various available high speed algorithms to compute DFT, the Cooley-Tukey algorithm is the simplest and most commonly used. These efficient algorithms, used to compute DFTs, are called Fast Fourier Transforms (FFTs).

This application note provides the source code to compute FFTs using a PIC17C42. The theory behind the FFT algorithms is well established and described in literature and hence not described in this application note. A Radix-2 Cooley-Tukey FFT is implemented with no limits on the length of the FFT. The length is only limited by the amount of available program memory space. All computations are performed using double precision arithmetic.

IMPLEMENTATION

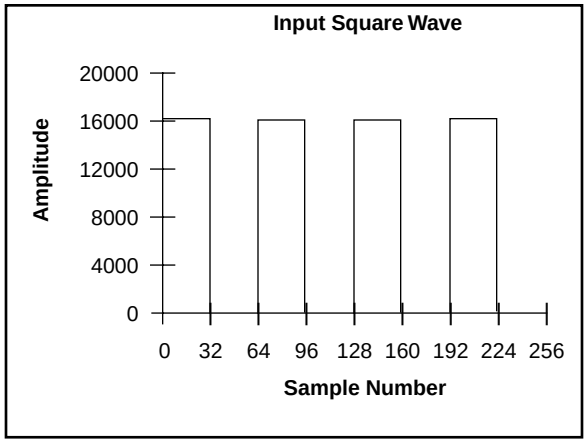
Since the PIC17C42 has only 232 x 8 general purpose RAM (equivalent of 116 x 16), at most a 32-point FFT (16-bit REAL & IMAGINARY data) can be implemented using on-chip RAM. To compute higher point FFTs, the data can be stored in the program memory space of the PIC17C42. The PIC17C42 has instructions (TABLRD & TABLWT) to transfer data between program memory space and on-chip file registers. In extended microcontroller mode, the PIC17C42 has 2K x 16 (0000h:07FFh) on-chip program memory space and is capable of addressing 62K x 16 (0800h:0FFFFh) of external program memory space. In this mode, the code (in this case, the FFT code) may reside on the on-chip EPROM and the data to be analyzed may be stored in external RAM (up to 62K). A suggested method of connecting external RAM (appropriate EEPROMs may also be used) is shown in Figure 3.

If the PIC17C42 is used in extended microcontroller mode and if all the code resides on-chip, then the cost may further be reduced by using only one external SRAM instead of two. The block diagram is shown in Figure 4. The 16-bit data stored in the external RAM is organized as low byte followed by high byte. To achieve this, the code presented in this application note needs minor modifications, especially where TABLRD and TABLWT instructions are used. Address indexing must be incremented by two since two reads/writes must be performed to access a 16-bit data.

The FFT is implemented with Decimation In Frequency. Thus the input data, before calling the FFT routine (R2FFT), should be in normal order and the transformed data should be in scrambled order. The original data is overwritten by the transformed data to conserve memory. This is achieved by use of in-place calculations. These in-place calculations cause the order of the DFT terms to be permuted. So at the end of the transform, all of the data needs to be unscrambled to get the right order of the DFT terms. In some applications the order of terms is not necessary. Keeping this in mind, the unscrambling code is written as a separate subroutine (Unscramble) and may be called if necessary.

Before implementing the FFT using a PIC17C42, a C program was written and tested. This high level programming helps in writing the assembly code and the results of both programs can be compared while debugging the assembly code. The C source code for the Radix-2 FFT is shown in Appendix A. The assembly code source file of the FFT program is shown in Appendix B. For a listing of the header file 17C42.h and the macro definition file 17C42.mac please refer to Appendices C and D respectively of application note AN540.

FIGURE 1: TEST WAVE FORM



© 1997 Microchip Technology Inc.

TESTING

The assembly code was developed and debugged using Microchip's PICMASTER In-Circuit Emulator System. A main program generates a test pattern (like a square wave) and calls FFT routines `R2FFT` and `Unscramble`. After the DFT terms are computed, the results are captured into PICMASTER's real-time trace buffer by putting a trace point on a dummy `TABLRD` instruction and capturing only the 2nd cycle (data cycle of `TABLRD`) of the instruction. The data from the trace buffer was hot linked to a Microsoft Excel spread sheet using DDE and then the graphs were plotted and analyzed.

The code was tested on various waveforms (a rectangular pulse, a triangular wave, square wave and a sine wave) using FFT lengths of 64, 256 and 1024. The results of a 256 point FFT on a square wave is shown below. The test waveform is shown in Figure 1 and it's frequency spectrum computed by the PIC17C42 is shown in Figure 2. As expected, the spectra appears at the odd harmonics of the input waveform's fundamental frequency (At N*256/64, N = 0,1,3,5,..).

PERFORMANCE

The performance of FFTs using a PIC17C42 is quite impressive for an 8-bit machine with no hardware multiplier. Also note that all computations are performed using double precision arithmetic (16- and 32-bit) which is the case for most of the low end DSPs. Table 1 provides the real-time performance in total number of Instruction cycles for both the `R2FFT` and `Unscramble` routines using 64, 256 and 1024 point FFTs. Note that the timings are in a worst case situation and in general will be a lot better than shown in the table. The worst case situation arises because the 16 x 16 software multiplier (`DblMult`) does not have uniform timing and depends on the input data. The worst case timing of the multiplier is used in computing it's performance.

FIGURE 2: FFT (MAGNITUDE SPECTRUM) OF FIGURE 1 COMPUTED BY A PIC17C42

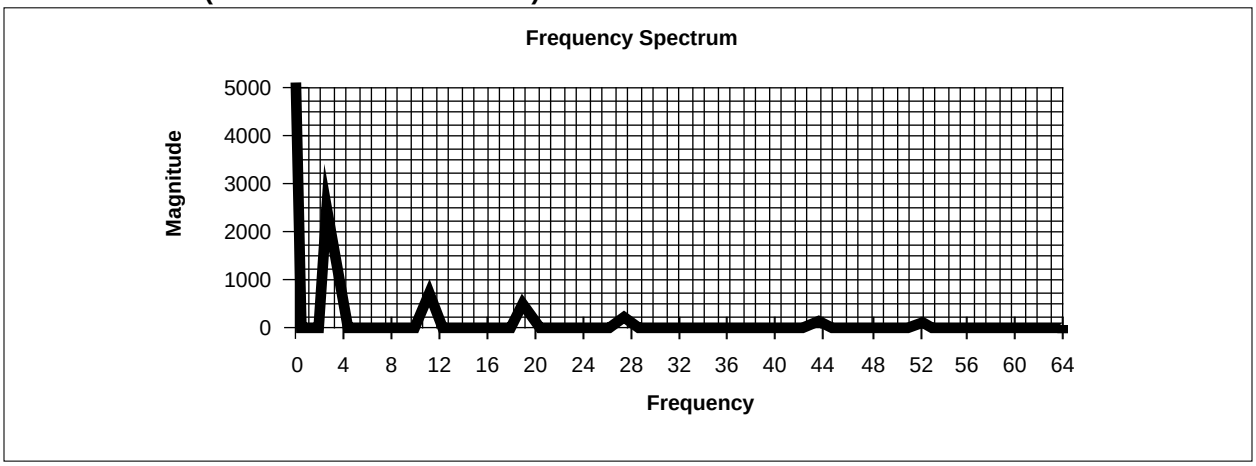


TABLE 1: WORST CASE PERFORMANCE OF FFT IN INSTRUCTION CYCLES AND REAL-TIME @ 25 MHz

N (FFT Length)	64 Point	256 Point	1024 Point
`R2FFT`	34116 + 768* Mult = 171588	178024 + 4096* Mult = 911208	878752 + 20480* Mult = 4544672
`Unscramble`	5143	22495	93525
`Total`	176731 (28.28 ms)	933703 (149.39 ms)	4638197 (742.11 ms)

© 1997 Microchip Technology Inc.

Table 2 shows the Program Memory and Data RAM requirements for an N Point FFT. The multiplier routine and other general purpose macro requirements are included in the memory requirements. The speed performance for the square wave test differs from Table 1, since “worst case timings” is not used and reflects a more reasonable data.

FFT APPLICATIONS

Although the FFT does not find a place in many microcontroller applications, it is very useful in providing a benchmark of the processor. As can be seen from Table 2, the performance is very satisfactory, considering the fact that the PIC17C42 is a Microcontroller and not a DSP. Also it should be borne in mind that all computations are performed in 16/32 bit arithmetic and that the PIC17C42 is a low-cost 8-bit device unlike DSPs which are relatively expensive.

In applications such as Instrumentation, where real-time FFT computation is not required, a PIC17C42 can be used as a single chip solution instead of a microcontroller and a Digital Signal Processor.

Suggested Reading:

[1] Rabiner. L.R., and Gold, B., Theory and Application Of Digital Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[2] Burrs, C.S., and Parks, T.W., DFT/FFT and Convolution Algorithms, New York: Wiley, 1985.

[3] Rodriguez, Jeffrey J., “An Improved FFT Digit-Reversal Algorithm,” IEEE Transactions On Acoustics, Speech, And Signal Processing, Vol. 37, No. 8, Aug 1989.

TABLE 2: REQUIREMENTS FOR RADIX-2 FFT @25 MHz

N (FFT Length)	64 Point	256 Point	1024 Point
Code Space (locations)	603 + 0.75*N = 651	603 + 0.75*N = 795	603 + 0.75*N = 1371
Data Storage in Program Memory Space	2*N = 128	2*N = 512	2*N = 2048
Scratch RAM	49	49	49
Performance (Square Wave Input Data)	122384 (19.58 ms)	644416 (103.11 ms)	3192176 (510.75 ms)

FIGURE 3: 2-SRAM EXTERNAL MEMORY CONNECTION

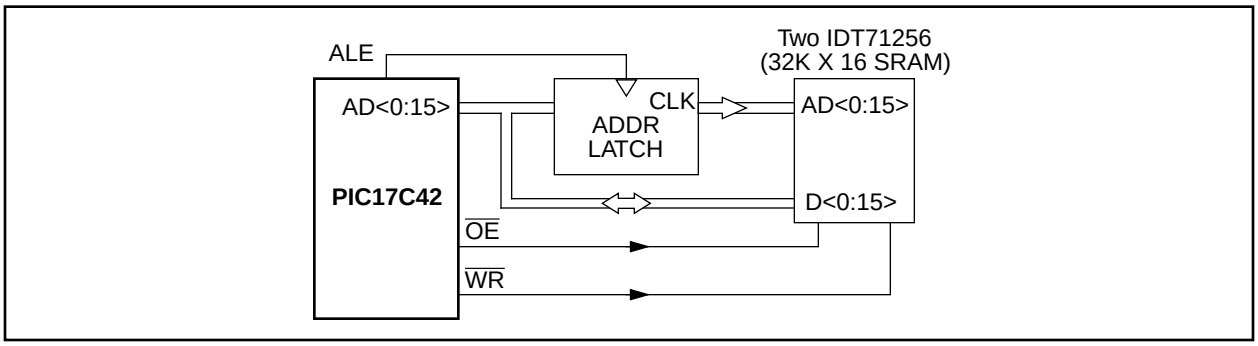
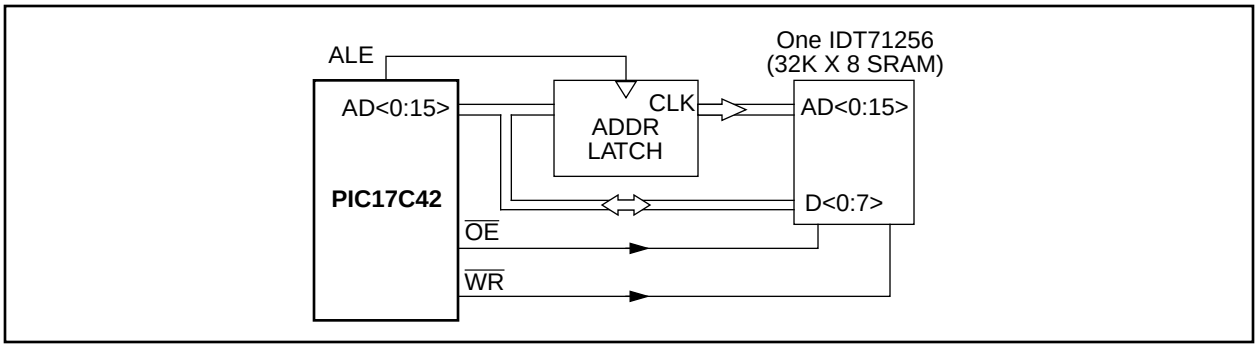


FIGURE 4: 1-SRAM ALTERNATIVE EXTERNAL MEMORY CONNECTION



© 1997 Microchip Technology Inc.

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

APPENDIX A: FFT ALGORITHM

```
MPASM 01.40 Released             FFT.ASM   1-16-1997  14:54:45         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001         LIST    P=17C42, columns=120,WRAP, R=DEC
                     00002 ;
                     00003 ;*****************************************************************
                     00004 ;               A Cooley-Tukey Radix-2 DIF FFT
                     00005 ;
                     00006 ;       Radix-2 implementation
                     00007 ;       Decimation In Frequency
                     00008 ;       Single Butterfly
                     00009 ;       Table Lookup of Twiddle Factors
                     00010 ;       Complex Input & Complex Output
                     00011 ;
                     00012 ;       All data is assumed to be 16 bits and the intermediate
                     00013 ;       results are stored in 32 bits
                     00014 ;
                     00015 ;       Length Of FFT must be a Power Of 2
                     00016 ;       Max Length Possible is 2**15
                     00017 ;
                     00018 ;       The input/output complex data is organized as a single array
                     00019 ;       of real data followed by imaginary data
                     00020 ;               Data is stored in External Memory and is accessed by
                     00021 ;       TABLRD & TABLWT Instructions
                     00022 ;
                     00023 ;       Program:          FFT.ASM
                     00024 ;       Revision Date:
                     00025 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                     00026 ;
                     00027 ;*****************************************************************
                     00028 ;
                     00029         include <p17c42.inc>
                     00001         LIST
                     00002 ;P17C42.INC Standard Header File, Ver. 1.03 Microchip Technology, Inc.
                     00264         LIST
                     00030
                     00031         #define TRUE    1
                     00032         #define FALSE   0
                     00033
                     00034         #define LSB     0
                     00035         #define MSB     7
                     00036
                     00037         include <17c42.mac>
                     00945         LIST
                     00038
                     00039 ;*****************************************************************
                     00040 ;  RLC16AB
                     00041 ;
                     00042 ; DESCRIPTION:
                     00043 ;       16 bit rotate left A into B
                     00044 ;
                     00045 ; ARGUMENTS:
                     00046 ;       2*a => b
                     00047 ;
                     00048 ; TIMING (in cycles):
                     00049 ;       3
                     00050 ;
```


© 1997 Microchip Technology Inc.

```
                  00051 ;
                  00052
                  00053 RLC16AB MACRO   a,b
                  00054
                  00055    BCF     ALUSTA,C
                  00056    RLCF    a+BB0,W
                  00057    MOVWF   b+BB0
                  00058    RLCF    a+BB1,W
                  00059    MOVWF   b+BB1
                  00060
                  00061         ENDM
                  00062
                  00063 ;*****************************************************************
                  00064 ;       TBLADDR
                  00065 ;
                  00066 ; DESCRIPTION:
                  00067 ;       Load 16 bit table pointer with specified label
                  00068 ;
                  00069 ; TIMING (in cycles):
                  00070 ;       4
                  00071 ;
                  00072
                  00073 TBLADDR MACRO   label
                  00074
                  00075    MOVLW   LOW   label
                  00076    MOVWF   TBLPTRL
                  00077    MOVLW   HIGH   label
                  00078    MOVWF   TBLPTRH
                  00079
                  00080         ENDM
                  00081
                  00082 ;*****************************************************************
                  00083 ;  ADDLBL
                  00084 ;
                  00085 ; DESCRIPTION:
                  00086 ;  ADd A Label (16 bit constant) To A File Register (16 bit)
                  00087 ;
                  00088 ; TIMING (in cycles):
                  00089 ;       4
                  00090 ;
                  00091
                  00092 ADDLBL  MACRO   label,f
                  00093
                  00094    MOVLW   LOW   label
                  00095    ADDWF   f+BB0, F
                  00096    MOVLW   HIGH   label
                  00097    ADDWFC  f+BB1, F
                  00098
                  00099         ENDM
                  00100
                  00101 ;*****************************************************************
                  00102
                  00103 ;
  00000100        00104 FftLen  .set    256             ; FFT Length
  00000008        00105 Power   .set    8               ; (2**Power = FftLen)
  000000EF        00106 DigitRevCount   .set    239     ; (FftLen-1) - (2**((Power+1)/2))
                  00107
  00000001        00108 SCALE_BUTTERFLY .set    TRUE    ; intermediate scaling performed
                  00109
  00000800        00110 EXT_RAM_START_ADDR      .set    0x0800  ; External Memory Data Storage
                  00111                                         ; Start Addr
                  00112 ;*****************************************************************
                  00113 ;
                  00114         CBLOCK  0
  00000000        00115           BB0,BB1,BB2,BB3 ; RAM offset constants
                  00116         ENDC
```

```
                  00117 ;
                  00118         CBLOCK  0x18
  00000018        00119           AARG,AARG1              ; 16 bit multiplier A
  0000001A        00120           BARG,BARG1              ; 16 bit multiplicand B
  0000001C        00121           DPX,DPX1,DPX2,DPX3      ; 32 bit multiplier result = A*B
                  00122         ENDC
                  00123 ;
                  00124         CBLOCK
  00000020        00125           ACC, ACC1, ACC2, ACC3  ; 32 bit accumulator for computations
                  00126         ENDC
                  00127 ;
                  00128         CBLOCK
  00000024        00129           count1,count11          ; N1
  00000026        00130           count2,count22          ; N2
  00000028        00131           QuartLen,QuartLen1      ; FftLen/4
                  00132         ENDC
                  00133 ;
                  00134         CBLOCK
  0000002A        00135           TF_Offset,TF_Offset1
  0000002C        00136           TF_Addr,TF_Addr1        ; twiddle factor address computations
  0000002E        00137           Cos,Cos1
  00000030        00138           Sin,Sin1
                  00139         ENDC
                  00140 ;
                  00141         CBLOCK
  00000032        00142           VarIloop,VarIloop1
  00000034        00143           VarJloop,VarJloop1
  00000036        00144           VarKloop
  00000037        00145           VarL,VarL1
                  00146         ENDC
                  00147 ;
                  00148         CBLOCK
  00000039        00149           Xi,Xi1
  0000003B        00150           Yi,Yi1
  0000003D        00151           Xl,Xl1
  0000003F        00152           Yl,Yl1
                  00153         ENDC
                  00154 ;
                  00155         CBLOCK
  00000041        00156           Xt,Xt1
  00000043        00157           Yt,Yt1
                  00158         ENDC
                  00159
                  00160         CBLOCK
  00000045        00161           temp,temp1,temp2,temp3
  00000049        00162           testCount,testCount1
  0000004B        00163           PulseCount, PulseCount1
                  00164         ENDC
                  00165 ;
                  00166 ;
                  00167 ;*****************************************************************
                  00168 ;               Test Program For FFT Subroutine
                  00169 ;*****************************************************************
                  00170 ;
0000              00171         ORG     0x0000
                  00172 ;
                  00173         include <square.asm>            ; Generate Test Vector Data
                  00001 ;*****************************************************************
                  00002 ;               Test Routine For FFT
                  00003 ; FFT Of Square Wave Pulse
                  00004 ;*****************************************************************
  00000008        00005 PulseWidthFactor        .set    8
                  00006 ;
0000              00007 testFft
                  00008   MOVK16  2*PulseWidthFactor,testCount
                      M
```


© 1997 Microchip Technology Inc.

```
0000 B010              M   MOVLW   (2*PulseWidthFactor) & 0xff
0001 0149              M   MOVWF   testCount+B0
0002 B000              M   MOVLW   ((2*PulseWidthFactor) >> 8)
0003 014A              M   MOVWF   testCount+B1
                       M
                   00009   CLR16   Yi
                       M
0004 293B              M   CLRF    Yi+B0, F
0005 293C              M   CLRF    Yi+B1, F
                       M
                   00010   TBLADDR ExtRamAddr      ; load table pointers with data start addr
                       M
0006 B000              M   MOVLW   LOW   ExtRamAddr
0007 010D              M   MOVWF   TBLPTRL
0008 B008              M   MOVLW   HIGH   ExtRamAddr
0009 010E              M   MOVWF   TBLPTRH
                       M
                   00011 ;
000A               00012 nextPulse
                   00013   MOVK    FftLen/PulseWidthFactor,PulseCount
                       M
000A B020              M   MOVLW   FftLen/PulseWidthFactor
000B 014B              M   MOVWF   PulseCount
                       M
                   00014   MOVK    FftLen/PulseWidthFactor,PulseCount1
                       M
000C B020              M   MOVLW   FftLen/PulseWidthFactor
000D 014C              M   MOVWF   PulseCount1
                       M
                   00015 ;
                   00016   MOVK16  0x3FFF,Xi
                       M
000E B0FF              M   MOVLW   (0x3FFF) & 0xff
000F 0139              M   MOVWF   Xi+B0
0010 B03F              M   MOVLW   ((0x3FFF) >> 8)
0011 013A              M   MOVWF   Xi+B1
                       M
0012               00017 LX1
0012 E034          00018   call    write
0013 174B          00019   decfsz  PulseCount, F
0014 C012          00020   goto    LX1
                   00021 ;
                   00022   CLR16   Xi
                       M
0015 2939              M   CLRF    Xi+B0, F
0016 293A              M   CLRF    Xi+B1, F
                       M
0017               00023 LX2
0017 E034          00024   call    write
0018 174C          00025   decfsz  PulseCount1, F
0019 C017          00026   goto    LX2
                   00027 ;
                   00028   DEC16   testCount
                       M
001A 290A              M   CLRF    WREG, F
001B 0749              M   DECF    testCount+B0, F
001C 034A              M   SUBWFB  testCount+B1, F
                       M
                   00029   TFSZ16  testCount
                       M
001D 6A49              M   MOVFP   testCount+B0,WREG
001E 084A              M   IORWF   testCount+B1,W
001F 330A              M   TSTFSZ  WREG
                       M
0020 C00A          00030   goto    nextPulse
                   00031
```

```
                  00174 ;
0021 E039         00175    call    R2FFT           ; Compute Fourier Transform
0022 E116         00176    call    Unscramble      ; Digit Reverse the scrambled data
                  00177 ;
                  00178 ; Fourier Transform Completed
                  00179 ;
                  00180 ; capture data to PIC-MASTER Trace Buffer
                  00181   MOVK16  FftLen*2,temp
                      M
0023 B000             M   MOVLW   (FftLen*2) & 0xff
0024 0145             M   MOVWF   temp+B0
0025 B002             M   MOVLW   ((FftLen*2) >> 8)
0026 0146             M   MOVWF   temp+B1
                      M
                  00182   TBLADDR ExtRamAddr      ; load table pointers with data start addr
                      M
0027 B000             M   MOVLW   LOW   ExtRamAddr
0028 010D             M   MOVWF   TBLPTRL
0029 B008             M   MOVLW   HIGH   ExtRamAddr
002A 010E             M   MOVWF   TBLPTRH
                      M
002B              00183 capture
002B A930         00184   tablrd  0,1,Sin         ; table latch = mem(tblptr)
                  00185   DEC16   temp
                      M
002C 290A             M   CLRF    WREG, F
002D 0745             M   DECF    temp+B0, F
002E 0346             M   SUBWFB  temp+B1, F
                      M
                  00186   TFSZ16  temp
                      M
002F 6A45             M   MOVFP   temp+B0,WREG
0030 0846             M   IORWF   temp+B1,W
0031 330A             M   TSTFSZ  WREG
                      M
0032 C02B         00187   goto    capture
                  00188 ;
0033 C033         00189 self  goto    self
                  00190 ;
0034              00191 write
0034 A439         00192    tlwt    0,Xi
0035 AF3A         00193    tablwt  1,1,Xi+BB1     ; auto increment for Imag Data
0036 A43B         00194    tlwt    0,Yi
0037 AF3C         00195    tablwt  1,1,Yi+BB1
0038 0002         00196    return
                  00197 ;
                  00198 ;*****************************************************************
                  00199 ;                       RADIX-2 FFT
                  00200 ;
                  00201 ;       Decimation In Frequency
                  00202 ;
                  00203 ; Input Data should be unscrambled
                  00204 ; Output Data at the end is in scrambled form
                  00205 ; To obtain the unscrambled form, the digit reverse counter
                  00206 ; subroutine, “Unscramble” should be called (see the example)
                  00207 ;
                  00208 ;*****************************************************************
                  00209
0039              00210 R2FFT
                  00211    MOVK16  FftLen,count2       ; count2 = N
                      M
0039 B000             M    MOVLW   (FftLen) & 0xff
003A 0126             M    MOVWF   count2+B0
003B B001             M    MOVLW   ((FftLen) >> 8)
003C 0127             M    MOVWF   count2+B1
                      M
```


© 1997 Microchip Technology Inc.

```
                    00212   MOVK16  FftLen/4,QuartLen    ; QuartLen = FftLen/4
                        M
003D B040               M   MOVLW   (FftLen/4) & 0xff
003E 0128               M   MOVWF   QuartLen+B0
003F B000               M   MOVLW   ((FftLen/4) >> 8)
0040 0129               M   MOVWF   QuartLen+B1
                        M
0041 292B           00213   clrf    TF_Offset+BB1, F
                    00214   MOVK    1,TF_Offset          ; Init TF_Offset = 1
                        M
0042 B001               M   MOVLW   1
0043 012A               M   MOVWF   TF_Offset
                        M
                    00215   MOVK    Power,VarKloop       ; Kloop
                        M
0044 B008               M   MOVLW   Power
0045 0136               M   MOVWF   VarKloop
                        M
0046                00216 Kloop                          ; for K = 1 to Power-1
                    00217   MOV16   count2,count1        ; count1 = count2
                        M
0046 6A26               M   MOVFP   count2+B0,WREG       ; get byte of a into w
0047 0124               M   MOVWF   count1+B0            ; move to b(B0)
0048 6A27               M   MOVFP   count2+B1,WREG       ; get byte of a into w
0049 0125               M   MOVWF   count1+B1            ; move to b(B1)
                        M
                    00218   RRC16   count2               ; count2 = count2/2
                        M
004A 1A27               M   RLCF    count2+B1,W          ; move sign into carry bit
004B 1927               M   RRCF    count2+B1, F
004C 1926               M   RRCF    count2+B0, F
                        M
                    00219   CLR16   VarJloop             ; J = 0
                        M
004D 2934               M   CLRF    VarJloop+B0, F
004E 2935               M   CLRF    VarJloop+B1, F
                        M
                    00220   CLR16   TF_Addr              ; TF_Addr = 0
                        M
004F 292C               M   CLRF    TF_Addr+B0, F
0050 292D               M   CLRF    TF_Addr+B1, F
                        M
0051                00221 Jloop
                    00222 ;
                    00223 ; Read Twiddle factors from Sine/Cosine Table from Prog Mem
                    00224 ;
                    00225    MOVFP16 TF_Addr,TBLPTRL ; load sine table address to table pointers
                        M
0051 6D2C               M    MOVFP   TF_Addr+B0,TBLPTRL+B0   ; move A(B0) to B(B0)
0052 6E2D               M    MOVFP   TF_Addr+B1,TBLPTRL+B1   ; move A(B1) to B(B1)
                        M
                    00226    ADDLBL  SineTable,TBLPTRL       ; add table offset
                        M
0053 B094               M    MOVLW   LOW   SineTable
0054 0F0D               M    ADDWF   TBLPTRL+BB0, F
0055 B002               M    MOVLW   HIGH   SineTable
0056 110E               M    ADDWFC  TBLPTRL+BB1, F
                        M
                    00227
0057 A830           00228    tablrd  0,0,Sin              ; Read Sine Value from lookup table
0058 A030           00229    tlrd    0,Sin
0059 A231           00230    tlrd    1,Sin+BB1
                    00231    ADD16   QuartLen,TBLPTRL
                        M
005A 6A28               M    MOVFP   QuartLen+B0,WREG     ; get lowest byte of a into w
005B 0F0D               M    ADDWF   TBLPTRL+B0, F        ; add lowest byte of b, save in b(B0)
```

© 1997 Microchip Technology Inc.

```
005C 6A29              M    MOVFP   QuartLen+B1,WREG      ; get 2nd byte of a into w
005D 110E              M    ADDWFC  TBLPTRL+B1, F         ; add 2nd byte of b, save in b(B1)
                       M
005E A82E           00232   tablrd  0,0,Cos               ; Read Cosine Value from table
005F A02E           00233   tlrd    0,Cos
0060 A22F           00234   tlrd    1,Cos+BB1
                    00235 ;
                    00236   ADD16   TF_Offset,TF_Addr     ; TF_Addr = TF_Addr + TF_Offset
                       M
0061 6A2A              M    MOVFP   TF_Offset+B0,WREG     ; get lowest byte of a into w
0062 0F2C              M    ADDWF   TF_Addr+B0, F         ; add lowest byte of b, save in b(B0)
0063 6A2B              M    MOVFP   TF_Offset+B1,WREG     ; get 2nd byte of a into w
0064 112D              M    ADDWFC  TF_Addr+B1, F         ; add 2nd byte of b, save in b(B1)
                       M
                    00237 ;
                    00238   RLC16AB VarJloop,VarIloop  ; I=J*2 since Real followed by Imag Data
                       M
0065 8804              M    BCF     ALUSTA,C
0066 1A34              M    RLCF    VarJloop+BB0,W
0067 0132              M    MOVWF   VarIloop+BB0
0068 1A35              M    RLCF    VarJloop+BB1,W
0069 0133              M    MOVWF   VarIloop+BB1
                       M
                    00239 ;
006A                00240 Iloop
                    00241
                    00242   RLC16AB count2,VarL           ; VarL = count2*2
                       M
006A 8804              M    BCF     ALUSTA,C
006B 1A26              M    RLCF    count2+BB0,W
006C 0137              M    MOVWF   VarL+BB0
006D 1A27              M    RLCF    count2+BB1,W
006E 0138              M    MOVWF   VarL+BB1
                       M
                    00243   ADD16   VarIloop,VarL         ; VarL = (I+count2)*2
                       M
006F 6A32              M    MOVFP   VarIloop+B0,WREG      ; get lowest byte of a into w
0070 0F37              M    ADDWF   VarL+B0, F            ; add lowest byte of b, save in b(B0)
0071 6A33              M    MOVFP   VarIloop+B1,WREG      ; get 2nd byte of a into w
0072 1138              M    ADDWFC  VarL+B1, F            ; add 2nd byte of b, save in b(B1)
                       M
                    00244 ;
                    00245 ; Get Real & Imag Data from external RAMs (Program Memory)
                    00246 ; load table pointers with data start addr
                    00247 ;
                    00248   MOVFP16 VarL,TBLPTRL          ; read data(L)
                       M
0073 6D37              M    MOVFP   VarL+B0,TBLPTRL+B0    ; move A(B0) to B(B0)
0074 6E38              M    MOVFP   VarL+B1,TBLPTRL+B1    ; move A(B1) to B(B1)
                       M
                    00249   ADDLBL  ExtRamAddr,TBLPTRL    ; add data addr offset
                       M
0075 B000              M    MOVLW   LOW   ExtRamAddr
0076 0F0D              M    ADDWF   TBLPTRL+BB0, F
0077 B008              M    MOVLW   HIGH   ExtRamAddr
0078 110E              M    ADDWFC  TBLPTRL+BB1, F
                       M
                    00250
0079 A93D           00251   tablrd  0,1,Xl               ; auto increment for Imag Data
007A A03D           00252   tlrd    0,Xl
007B A23E           00253   tlrd    1,Xl+BB1             ; real data XL
007C A83F           00254   tablrd  0,0,Yl
007D A03F           00255   tlrd    0,Yl
007E A240           00256   tlrd    1,Yl+BB1             ; imag data YL
                    00257
                    00258   MOVFP16 VarIloop,TBLPTRL      ; read data(I)
```


© 1997 Microchip Technology Inc.

```
                        M
007F 6D32               M    MOVFP   VarIloop+B0,TBLPTRL+B0 ; move A(B0) to B(B0)
0080 6E33               M    MOVFP   VarIloop+B1,TBLPTRL+B1 ; move A(B1) to B(B1)
                        M
                    00259    ADDLBL  ExtRamAddr,TBLPTRL    ; add data addr offset
                        M
0081 B000               M    MOVLW   LOW   ExtRamAddr
0082 0F0D               M    ADDWF   TBLPTRL+BB0, F
0083 B008               M    MOVLW   HIGH   ExtRamAddr
0084 110E               M    ADDWFC  TBLPTRL+BB1, F
                        M
                    00260
0085 A939           00261    tablrd  0,1,Xi                ; auto increment for Imag Data
0086 A039           00262    tlrd    0,Xi
0087 A23A           00263    tlrd    1,Xi+BB1              ; real data XI
0088 A83B           00264    tablrd  0,0,Yi
0089 A03B           00265    tlrd    0,Yi
008A A23C           00266    tlrd    1,Yi+BB1              ; imag data YI
                    00267 ;
                    00268 ; Real & Imag Data is fetched
                    00269 ; Compute Butterfly
                    00270 ;
                    00271    SUB16ACC       Xl,Xi,Xt       ; Xt = Xi - Xl
                        M
008B 6A3D               M    MOVFP   Xl+B0,WREG            ; get lowest byte of a into w
008C 0439               M    SUBWF   Xi+B0,W               ; sub lowest byte of b, save in b(B0)
008D 0141               M    MOVWF   Xt+B0
008E 6A3E               M    MOVFP   Xl+B1,WREG            ; get 2nd byte of a into w
008F 023A               M    SUBWFB  Xi+B1,W               ; sub 2nd byte of b, save in b(B1)
0090 0142               M    MOVWF   Xt+B1
                        M
                    00272    ADD16           Xl,Xi         ; Xi = Xi + Xl
                        M
0091 6A3D               M    MOVFP   Xl+B0,WREG            ; get lowest byte of a into w
0092 0F39               M    ADDWF   Xi+B0, F              ; add lowest byte of b, save in b(B0)
0093 6A3E               M    MOVFP   Xl+B1,WREG            ; get 2nd byte of a into w
0094 113A               M    ADDWFC  Xi+B1, F              ; add 2nd byte of b, save in b(B1)
                        M
                    00273    SUB16ACC        Yl,Yi,Yt      ; Yt = Yi - Yl
                        M
0095 6A3F               M    MOVFP   Yl+B0,WREG            ; get lowest byte of a into w
0096 043B               M    SUBWF   Yi+B0,W               ; sub lowest byte of b, save in b(B0)
0097 0143               M    MOVWF   Yt+B0
0098 6A40               M    MOVFP   Yl+B1,WREG            ; get 2nd byte of a into w
0099 023C               M    SUBWFB  Yi+B1,W               ; sub 2nd byte of b, save in b(B1)
009A 0144               M    MOVWF   Yt+B1
                        M
                    00274    ADD16           Yl,Yi         ; Yi = Yi + Yl
                        M
009B 6A3F               M    MOVFP   Yl+B0,WREG            ; get lowest byte of a into w
009C 0F3B               M    ADDWF   Yi+B0, F              ; add lowest byte of b, save in b(B0)
009D 6A40               M    MOVFP   Yl+B1,WREG            ; get 2nd byte of a into w
009E 113C               M    ADDWFC  Yi+B1, F              ; add 2nd byte of b, save in b(B1)
                        M
                    00275 ;
                    00276    #if SCALE_BUTTERFLY
                    00277    RRC16   Xi
                        M
009F 1A3A               M    RLCF    Xi+B1,W               ; move sign into carry bit
00A0 193A               M    RRCF    Xi+B1, F
00A1 1939               M    RRCF    Xi+B0, F
                        M
                    00278    RRC16   Yi
                        M
00A2 1A3C               M    RLCF    Yi+B1,W               ; move sign into carry bit
00A3 193C               M    RRCF    Yi+B1, F
```

© 1997 Microchip Technology Inc.

```
00A4 193B              M    RRCF    Yi+B0, F
                       M
                    00279   RRC16   Xt
                       M
00A5 1A42              M    RLCF    Xt+B1,W              ; move sign into carry bit
00A6 1942              M    RRCF    Xt+B1, F
00A7 1941              M    RRCF    Xt+B0, F
                       M
                    00280   RRC16   Yt
                       M
00A8 1A44              M    RLCF    Yt+B1,W              ; move sign into carry bit
00A9 1944              M    RRCF    Yt+B1, F
00AA 1943              M    RRCF    Yt+B0, F
                       M
                    00281   #endif
                    00282 ;
                    00283
                    00284   MOVFP16 Cos,AARG
                       M
00AB 782E              M    MOVFP   Cos+B0,AARG+B0       ; move A(B0) to B(B0)
00AC 792F              M    MOVFP   Cos+B1,AARG+B1       ; move A(B1) to B(B1)
                       M
                    00285   MOVFP16 Yt,BARG
                       M
00AD 7A43              M    MOVFP   Yt+B0,BARG+B0        ; move A(B0) to B(B0)
00AE 7B44              M    MOVFP   Yt+B1,BARG+B1        ; move A(B1) to B(B1)
                       M
00AF E182           00286   call    DblMult              ; COS*Yt
                    00287   MOVPF32 DPX,ACC
                       M
00B0 5C20              M    MOVPF   DPX+B0,ACC+B0        ; move A(B0) to B(B0)
00B1 5D21              M    MOVPF   DPX+B1,ACC+B1        ; move A(B1) to B(B1)
00B2 5E22              M    MOVPF   DPX+B2,ACC+B2        ; move A(B2) to B(B2)
00B3 5F23              M    MOVPF   DPX+B3,ACC+B3        ; move A(B3) to B(B3)
                       M
                    00288
                    00289   MOVFP16 Sin,AARG
                       M
00B4 7830              M    MOVFP   Sin+B0,AARG+B0       ; move A(B0) to B(B0)
00B5 7931              M    MOVFP   Sin+B1,AARG+B1       ; move A(B1) to B(B1)
                       M
                    00290   MOVFP16 Xt,BARG
                       M
00B6 7A41              M    MOVFP   Xt+B0,BARG+B0        ; move A(B0) to B(B0)
00B7 7B42              M    MOVFP   Xt+B1,BARG+B1        ; move A(B1) to B(B1)
                       M
00B8 E182           00291   Call    DblMult              ; SIN*Xt, Scale if necessary
                    00292
                    00293   ADD32   ACC,DPX
                       M
00B9 6A20              M    MOVFP   ACC+B0,WREG          ; get lowest byte of a into w
00BA 0F1C              M    ADDWF   DPX+B0, F            ; add lowest byte of b, save in b(B0)
00BB 6A21              M    MOVFP   ACC+B1,WREG          ; get 2nd byte of a into w
00BC 111D              M    ADDWFC  DPX+B1, F            ; add 2nd byte of b, save in b(B1)
00BD 6A22              M    MOVFP   ACC+B2,WREG          ; get 3rd byte of a into w
00BE 111E              M    ADDWFC  DPX+B2, F            ; add 3rd byte of b, save in b(B2)
00BF 6A23              M    MOVFP   ACC+B3,WREG          ; get 4th byte of a into w
00C0 111F              M    ADDWFC  DPX+B3, F            ; add 4th byte of b, save in b(B3)
                       M
                    00294   MOVPF16 DPX+BB2,Yl     ; Yl = COS*Yt + SIN*Xt, Scale if necessary
                       M
00C1 5E3F              M    MOVPF   DPX+BB2+B0,Yl+B0     ; move A(B0) to B(B0)
00C2 5F40              M    MOVPF   DPX+BB2+B1,Yl+B1     ; move A(B1) to B(B1)
                       M
                    00295 ;
                    00296   MOVFP16 Yt,BARG              ; AARG = SIN, BARG = Yt
```


© 1997 Microchip Technology Inc.

```
                        M
00C3 7A43               M    MOVFP   Yt+B0,BARG+B0          ; move A(B0) to B(B0)
00C4 7B44               M    MOVFP   Yt+B1,BARG+B1          ; move A(B1) to B(B1)
                        M
00C5 E182             00297  Call    DblMult                ; SIN*Yt
                      00298  MOVPF32 DPX,ACC
                        M
00C6 5C20               M    MOVPF   DPX+B0,ACC+B0          ; move A(B0) to B(B0)
00C7 5D21               M    MOVPF   DPX+B1,ACC+B1          ; move A(B1) to B(B1)
00C8 5E22               M    MOVPF   DPX+B2,ACC+B2          ; move A(B2) to B(B2)
00C9 5F23               M    MOVPF   DPX+B3,ACC+B3          ; move A(B3) to B(B3)
                        M
                      00299
                      00300  MOVFP16 Cos,AARG
                        M
00CA 782E               M    MOVFP   Cos+B0,AARG+B0         ; move A(B0) to B(B0)
00CB 792F               M    MOVFP   Cos+B1,AARG+B1         ; move A(B1) to B(B1)
                        M
                      00301  MOVFP16 Xt,BARG
                        M
00CC 7A41               M    MOVFP   Xt+B0,BARG+B0          ; move A(B0) to B(B0)
00CD 7B42               M    MOVFP   Xt+B1,BARG+B1          ; move A(B1) to B(B1)
                        M
00CE E182             00302  Call    DblMult                ; COS*Xt, Scale if necessary
                      00303
                      00304  SUB32   ACC,DPX                ; DPX = COS*Xt - SIN*Yt
                        M
00CF 6A20               M    MOVFP   ACC+B0,WREG            ; get lowest byte of a into w
00D0 051C               M    SUBWF   DPX+B0, F              ; sub lowest byte of b, save in b(B0)
00D1 6A21               M    MOVFP   ACC+B1,WREG            ; get 2nd byte of a into w
00D2 031D               M    SUBWFB  DPX+B1, F              ; sub 2nd byte of b, save in b(B1)
00D3 6A22               M    MOVFP   ACC+B2,WREG            ; get 3rd byte of a into w
00D4 031E               M    SUBWFB  DPX+B2, F              ; sub 3rd byte of b, save in b(B2)
00D5 6A23               M    MOVFP   ACC+B3,WREG            ; get 4th byte of a into w
00D6 031F               M    SUBWFB  DPX+B3, F              ; sub 4th byte of b, save in b(B3)
                        M
                      00305  MOVPF16 DPX+BB2,Xl       ; Xl = COS*Xt - SIN*Yt, Scale if necessary
                        M
00D7 5E3D               M    MOVPF   DPX+BB2+B0,Xl+B0       ; move A(B0) to B(B0)
00D8 5F3E               M    MOVPF   DPX+BB2+B1,Xl+B1       ; move A(B1) to B(B1)
                        M
                      00306 ;
                      00307 ;
                      00308 ; Store results of butterfly
                      00309 ;
                      00310    DEC16   TBLPTRL         ; table pointer already loaded with I addr
                        M
00D9 290A               M    CLRF    WREG, F
00DA 070D               M    DECF    TBLPTRL+B0, F
00DB 030E               M    SUBWFB  TBLPTRL+B1, F
                        M
                      00311
00DC A439             00312  tlwt    0,Xi
00DD AF3A             00313  tablwt  1,1,Xi+BB1            ; auto increment for Imag Data
00DE A43B             00314  tlwt    0,Yi
00DF AE3C             00315  tablwt  1,0,Yi+BB1            ; Xi & Yi stored
                      00316
                      00317  MOVFP16 VarL,TBLPTRL          ; read data(L)
                        M
00E0 6D37               M    MOVFP   VarL+B0,TBLPTRL+B0     ; move A(B0) to B(B0)
00E1 6E38               M    MOVFP   VarL+B1,TBLPTRL+B1     ; move A(B1) to B(B1)
                        M
                      00318  ADDLBL  ExtRamAddr,TBLPTRL     ; add data addr offset
                        M
00E2 B000               M    MOVLW   LOW   ExtRamAddr
00E3 0F0D               M    ADDWF   TBLPTRL+BB0, F
```

```
00E4 B008              M    MOVLW   HIGH   ExtRamAddr
00E5 110E              M    ADDWFC  TBLPTRL+BB1, F
                       M
                    00319
00E6 A43D           00320    tlwt    0,Xl
00E7 AF3E           00321    tablwt  1,1,Xl+BB1           ; auto increment for Imag Data
00E8 A43F           00322    tlwt    0,Yl
00E9 AE40           00323    tablwt  1,0,Yl+BB1           ; X(L) & Y(L) stored
                    00324 ;
                    00325 ; Increment for next Iloop
                    00326 ;
                    00327    RLC16AB count1,temp          ; temp = count1*2
                       M
00EA 8804              M    BCF     ALUSTA,C
00EB 1A24              M    RLCF    count1+BB0,W
00EC 0145              M    MOVWF   temp+BB0
00ED 1A25              M    RLCF    count1+BB1,W
00EE 0146              M    MOVWF   temp+BB1
                       M
                    00328    ADD16   temp,VarIloop        ; I = I + temp
                       M
00EF 6A45              M    MOVFP   temp+B0,WREG         ; get lowest byte of a into w
00F0 0F32              M    ADDWF   VarIloop+B0, F       ; add lowest byte of b, save in b(B0)
00F1 6A46              M    MOVFP   temp+B1,WREG         ; get 2nd byte of a into w
00F2 1133              M    ADDWFC  VarIloop+B1, F       ; add 2nd byte of b, save in b(B1)
                       M
                    00329 ;
                    00330
                    00331    MOVK16  (FftLen*2),temp
                       M
00F3 B000              M    MOVLW   ((FftLen*2)) & 0xff
00F4 0145              M    MOVWF   temp+B0
00F5 B002              M    MOVLW   (((FftLen*2)) >> 8)
00F6 0146              M    MOVWF   temp+B1
                       M
                    00332    SUB16   VarIloop,temp        ; temp = 2*FftLen - I
                       M
00F7 6A32              M    MOVFP   VarIloop+B0,WREG     ; get lowest byte of a into w
00F8 0545              M    SUBWF   temp+B0, F           ; sub lowest byte of b, save in b(B0)
00F9 6A33              M    MOVFP   VarIloop+B1,WREG     ; get 2nd byte of a into w
00FA 0346              M    SUBWFB  temp+B1, F           ; sub 2nd byte of b, save in b(B1)
                       M
                    00333    DEC16   temp
                       M
00FB 290A              M    CLRF    WREG, F
00FC 0745              M    DECF    temp+B0, F
00FD 0346              M    SUBWFB  temp+B1, F
                       M
00FE 9746           00334    Btfss   temp+BB1,MSB
00FF C06A           00335    Goto    Iloop                ; while I < 2*FftLen
                    00336 ;
                    00337 ; I Loop end
                    00338 ;
                    00339 ; increment for next J Loop
                    00340 ;
                    00341
                    00342    INC16   VarJloop             ; J = J + 1
                       M
0100 290A              M    CLRF    WREG, F
0101 1534              M         INCF    (VarJloop)+B0, F
0102 1135              M    ADDWFC  (VarJloop)+B1, F
                       M
                    00343
                    00344    MOV16   count2,temp
                       M
0103 6A26              M    MOVFP   count2+B0,WREG       ; get byte of a into w
```


© 1997 Microchip Technology Inc.

```
0104 0145              M    MOVWF   temp+B0              ; move to b(B0)
0105 6A27              M    MOVFP   count2+B1,WREG       ; get byte of a into w
0106 0146              M    MOVWF   temp+B1              ; move to b(B1)
                       M
                   00345    SUB16   VarJloop,temp        ; temp = count2 - J
                       M
0107 6A34              M    MOVFP   VarJloop+B0,WREG     ; get lowest byte of a into w
0108 0545              M    SUBWF   temp+B0, F           ; sub lowest byte of b, save in b(B0)
0109 6A35              M    MOVFP   VarJloop+B1,WREG     ; get 2nd byte of a into w
010A 0346              M    SUBWFB  temp+B1, F           ; sub 2nd byte of b, save in b(B1)
                       M
                   00346    DEC16   temp
                       M
010B 290A              M    CLRF    WREG, F
010C 0745              M    DECF    temp+B0, F
010D 0346              M    SUBWFB  temp+B1, F
                       M
010E 9746          00347    Btfss   temp+BB1,MSB
010F C051          00348    Goto    Jloop                ; while J < count2
                   00349 ;
                   00350 ; J Loop end
                   00351 ;
                   00352 ; increment for next K Loop
                   00353 ;
                   00354    RLC16   TF_Offset            ; TF_Offset = 2 * TF_Offset
0110 8804              M    BCF     ALUSTA,C
0111 1B2A              M    RLCF    TF_Offset+B0, F
0112 1B2B              M    RLCF    TF_Offset+B1, F
                       M
0113 1736          00355    decfsz  VarKloop, F
0114 C046          00356    Goto    Kloop                ; while K < Power
                   00357 ;
0115 0002          00358    return                       ; FFT complete
                   00359 ;
                   00360 ; K Loop End
                   00361 ; FFT Computation Over with data scrambled
                   00362 ; Descramble the data using “Unscramble” Routine
                   00363 ;
                   00364 ;*****************************************************************
                   00365 ;       Unscramble Data Order Sequence
                   00366 ;       A digit reverse counter
                   00367 ;*****************************************************************
                   00368
                   00369         include <reverse.asm>
                   00001 ;*****************************************************************
                   00002 ;                       A digit reverse counter
                   00003 ;
                   00004 ;       Unscramble Data Order Sequence Of Radix-2 FFT
                   00005 ;       Length (must be a power of 2) is limited only by
                   00006 ;       the amount of External RAM available and must be
                   00007 ;       a number less than 2**15
                   00008 ;
                   00009 ;*****************************************************************
                   00010
0116               00011 Unscramble
                   00012
                   00013    CLR16   VarJloop        ; J = 0
                       M
0116 2934              M    CLRF    VarJloop+B0, F
0117 2935              M    CLRF    VarJloop+B1, F
                       M
0118 2933          00014    ClRF    VarIloop+B1, F
                   00015    MOVK    1,VarIloop      ; I = 1
                       M
0119 B001              M    MOVLW   1
011A 0132              M    MOVWF   VarIloop
```

```
                    M
011B                00016 nextI
                    00017    MOVK16  FftLen/2,VarKloop
                        M
011B B080               M    MOVLW   (FftLen/2) & 0xff
011C 0136               M    MOVWF   VarKloop+B0
011D B000               M    MOVLW   ((FftLen/2) >> 8)
011E 0137               M    MOVWF   VarKloop+B1
                        M
011F C127           00018    Goto    testK
0120                00019 KlessJ
                    00020    SUB16   VarKloop,VarJloop     ; J = J - K
                        M
0120 6A36               M    MOVFP   VarKloop+B0,WREG      ; get lowest byte of a into w
0121 0534               M    SUBWF   VarJloop+B0, F        ; sub lowest byte of b, save in b(B0)
0122 6A37               M    MOVFP   VarKloop+B1,WREG      ; get 2nd byte of a into w
0123 0335               M    SUBWFB  VarJloop+B1, F        ; sub 2nd byte of b, save in b(B1)
                        M
                    00021    RRC16   VarKloop              ; K = K/2
                        M
0124 1A37               M    RLCF    VarKloop+B1,W         ; move sign into carry bit
0125 1937               M    RRCF    VarKloop+B1, F
0126 1936               M    RRCF    VarKloop+B0, F
                        M
0127                00022 testK
                    00023    MOV16   VarJloop,temp
                        M
0127 6A34               M    MOVFP   VarJloop+B0,WREG      ; get byte of a into w
0128 0145               M    MOVWF   temp+B0               ; move to b(B0)
0129 6A35               M    MOVFP   VarJloop+B1,WREG      ; get byte of a into w
012A 0146               M    MOVWF   temp+B1               ; move to b(B1)
                        M
                    00024    SUB16   VarKloop,temp         ; temp = J - K
                        M
012B 6A36               M    MOVFP   VarKloop+B0,WREG      ; get lowest byte of a into w
012C 0545               M    SUBWF   temp+B0, F            ; sub lowest byte of b, save in b(B0)
012D 6A37               M    MOVFP   VarKloop+B1,WREG      ; get 2nd byte of a into w
012E 0346               M    SUBWFB  temp+B1, F            ; sub 2nd byte of b, save in b(B1)
                        M
012F 9746           00025    Btfss   temp+B1,MSB
0130 C120           00026    goto    KlessJ                ; while K < J
                    00027
                    00028    ADD16   VarKloop,VarJloop     ; J = J + K
                        M
0131 6A36               M    MOVFP   VarKloop+B0,WREG      ; get lowest byte of a into w
0132 0F34               M    ADDWF   VarJloop+B0, F        ; add lowest byte of b, save in b(B0)
0133 6A37               M    MOVFP   VarKloop+B1,WREG      ; get 2nd byte of a into w
0134 1135               M    ADDWFC  VarJloop+B1, F        ; add 2nd byte of b, save in b(B1)
                        M
                    00029 ;
                    00030 ; if (i < j) then swap data(i) & data(j)
                    00031 ;
                    00032    MOV16   VarJloop,temp
                        M
0135 6A34               M    MOVFP   VarJloop+B0,WREG      ; get byte of a into w
0136 0145               M    MOVWF   temp+B0               ; move to b(B0)
0137 6A35               M    MOVFP   VarJloop+B1,WREG      ; get byte of a into w
0138 0146               M    MOVWF   temp+B1               ; move to b(B1)
                        M
                    00033    SUB16   VarIloop,temp         ; temp = J - I
                        M
0139 6A32               M    MOVFP   VarIloop+B0,WREG      ; get lowest byte of a into w
013A 0545               M    SUBWF   temp+B0, F            ; sub lowest byte of b, save in b(B0)
013B 6A33               M    MOVFP   VarIloop+B1,WREG      ; get 2nd byte of a into w
013C 0346               M    SUBWFB  temp+B1, F            ; sub 2nd byte of b, save in b(B1)
                        M
```


© 1997 Microchip Technology Inc.

```
                  00034    DEC16   temp
                      M
013D 290A             M    CLRF    WREG, F
013E 0745             M    DECF    temp+B0, F
013F 0346             M    SUBWFB  temp+B1, F
                      M
0140 9F46         00035    Btfsc   temp+B1,MSB
0141 C174         00036    Goto    incI
                  00037 ;
                  00038 ; swap data
                  00039 ; read data(i)
                  00040
                  00041    RLC16AB VarIloop,TBLPTRL      ; add twice the addr, since Real Data
                      M
0142 8804             M    BCF     ALUSTA,C
0143 1A32             M    RLCF    VarIloop+BB0,W
0144 010D             M    MOVWF   TBLPTRL+BB0
0145 1A33             M    RLCF    VarIloop+BB1,W
0146 010E             M    MOVWF   TBLPTRL+BB1
                      M
                  00042    ADDLBL  ExtRamAddr,TBLPTRL    ; is followed by Imag Data
                      M
0147 B000             M    MOVLW   LOW   ExtRamAddr
0148 0F0D             M    ADDWF   TBLPTRL+BB0, F
0149 B008             M    MOVLW   HIGH   ExtRamAddr
014A 110E             M    ADDWFC  TBLPTRL+BB1, F
                      M
                  00043
014B A939         00044    tablrd  0,1,Xi                ; auto increment for Imag Data
014C A039         00045    tlrd    0,Xi
014D A23A         00046    tlrd    1,Xi+B1               ; real data XI
014E A83B         00047    tablrd  0,0,Yi
014F A03B         00048    tlrd    0,Yi
0150 A23C         00049    tlrd    1,Yi+B1               ; imag data YI
                  00050 ;
                  00051 ; read data(j)
                  00052 ;
                  00053    RLC16AB VarJloop,TBLPTRL      ; add twice the addr, since Real Data
                      M
0151 8804             M    BCF     ALUSTA,C
0152 1A34             M    RLCF    VarJloop+BB0,W
0153 010D             M    MOVWF   TBLPTRL+BB0
0154 1A35             M    RLCF    VarJloop+BB1,W
0155 010E             M    MOVWF   TBLPTRL+BB1
                      M
                  00054    ADDLBL  ExtRamAddr,TBLPTRL    ; is followed by Imag Data
                      M
0156 B000             M    MOVLW   LOW   ExtRamAddr
0157 0F0D             M    ADDWF   TBLPTRL+BB0, F
0158 B008             M    MOVLW   HIGH   ExtRamAddr
0159 110E             M    ADDWFC  TBLPTRL+BB1, F
                      M
                  00055
015A A93D         00056    tablrd  0,1,Xl               ; auto increment for Imag Data
015B A03D         00057    tlrd    0,Xl
015C A23E         00058    tlrd    1,Xl+B1              ; real data XL
015D A83F         00059    tablrd  0,0,Yl
015E A03F         00060    tlrd    0,Yl
015F A240         00061    tlrd    1,Yl+B1              ; imag data YL
                  00062 ;
                  00063 ; Interchange data(I) & data(J)
                  00064 ;
                  00065 ; J addr already loaded into table pointers, by auto incremented
                  00066 ;
                  00067    DEC16   TBLPTRL
                      M
```

```
0160 290A               M    CLRF    WREG, F
0161 070D               M    DECF    TBLPTRL+B0, F
0162 030E               M    SUBWFB  TBLPTRL+B1, F
                        M
                    00068
0163 A439           00069    tlwt    0,Xi
0164 AF3A           00070    tablwt  1,1,Xi+B1            ; auto increment for Imag Data
0165 A43B           00071    tlwt    0,Yi
0166 AE3C           00072    tablwt  1,0,Yi+B1            ; X(I) & Y(I) stored
                    00073
                    00074    RLC16AB VarIloop,TBLPTRL     ; add twice the addr, since Real Data
                        M
0167 8804               M    BCF     ALUSTA,C
0168 1A32               M    RLCF    VarIloop+BB0,W
0169 010D               M    MOVWF   TBLPTRL+BB0
016A 1A33               M    RLCF    VarIloop+BB1,W
016B 010E               M    MOVWF   TBLPTRL+BB1
                        M
                    00075    ADDLBL  ExtRamAddr,TBLPTRL   ; is followed by Imag Data
                        M
016C B000               M    MOVLW   LOW   ExtRamAddr
016D 0F0D               M    ADDWF   TBLPTRL+BB0, F
016E B008               M    MOVLW   HIGH   ExtRamAddr
016F 110E               M    ADDWFC  TBLPTRL+BB1, F
                        M
                    00076
0170 A43D           00077    tlwt    0,Xl
0171 AF3E           00078    tablwt  1,1,Xl+B1            ; auto increment for Imag Data
0172 A43F           00079    tlwt    0,Yl
0173 AE40           00080    tablwt  1,0,Yl+B1            ; X(L) & Y(L) stored
                    00081 ;
                    00082 ; increment I
                    00083 ;
0174                00084 incI
                    00085   INC16   VarIloop
                        M
0174 290A               M    CLRF    WREG, F
0175 1532               M    INCF    (VarIloop)+B0, F
0176 1133               M    ADDWFC  (VarIloop)+B1, F
                        M
                    00086
                    00087    MOVK16  DigitRevCount,temp
                        M
0177 B0EF               M    MOVLW   (DigitRevCount) & 0xff
0178 0145               M    MOVWF   temp+B0
0179 B000               M    MOVLW   ((DigitRevCount) >> 8)
017A 0146               M    MOVWF   temp+B1
                        M
                    00088    SUB16   VarIloop,temp        ; temp = DigitRevCount - I
                        M
017B 6A32               M    MOVFP   VarIloop+B0,WREG     ; get lowest byte of a into w
017C 0545               M    SUBWF   temp+B0, F           ; sub lowest byte of b, save in b(B0)
017D 6A33               M    MOVFP   VarIloop+B1,WREG     ; get 2nd byte of a into w
017E 0346               M    SUBWFB  temp+B1, F           ; sub 2nd byte of b, save in b(B1)
                        M
017F 9746           00089    Btfss   temp+B1,MSB
0180 C11B           00090    Goto    nextI                ; while i < DigitRevCount
0181 0002           00091    return
                    00092 ;
                    00093 ; End digit reverse counter
                    00094 ;
                    00095 ;*****************************************************************
                    00096
                    00370
                    00371 ;*****************************************************************
                    00372 ;       Include Double Precision Multiplication Routine
```


© 1997 Microchip Technology Inc.

```
                 00373 ;*****************************************************************
  00000001       00374 SIGNED  equ     TRUE
                 00375
                 00376         include <17c42mpy.mac>
                 00178         LIST
                 00377
                 00378 ;
                 00379 ;*****************************************************************
                 00380 ;               Sine-Cosine Tables
                 00381 ;*****************************************************************
                 00382 ;
                 00383         include <fft256.tbl>
                 00001 ;
                 00002 ;               256 Point FFT Sine Table
                 00003 ; coefficient table (size of table is 3n/4).
                 00004 ;
0294             00005 SineTable
                 00006 ;
0294 0000        00007         data          0
0295 0324        00008         data          804
0296 0648        00009         data          1608
0297 096A        00010         data          2410
0298 0C8C        00011         data          3212
0299 0FAB        00012         data          4011
029A 12C8        00013         data          4808
029B 15E2        00014         data          5602
029C 18F9        00015         data          6393
029D 1C0B        00016         data          7179
029E 1F1A        00017         data          7962
029F 2223        00018         data          8739
02A0 2528        00019         data          9512
02A1 2826        00020         data          10278
02A2 2B1F        00021         data          11039
02A3 2E11        00022         data          11793
02A4 30FB        00023         data          12539
02A5 33DF        00024         data          13279
02A6 36BA        00025         data          14010
02A7 398C        00026         data          14732
02A8 3C56        00027         data          15446
02A9 3F17        00028         data          16151
02AA 41CE        00029         data          16846
02AB 447A        00030         data          17530
02AC 471C        00031         data          18204
02AD 49B4        00032         data          18868
02AE 4C3F        00033         data          19519
02AF 4EBF        00034         data          20159
02B0 5133        00035         data          20787
02B1 539B        00036         data          21403
02B2 55F5        00037         data          22005
02B3 5842        00038         data          22594
02B4 5A82        00039         data          23170
02B5 5CB3        00040         data          23731
02B6 5ED7        00041         data          24279
02B7 60EB        00042         data          24811
02B8 62F1        00043         data          25329
02B9 64E8        00044         data          25832
02BA 66CF        00045         data          26319
02BB 68A6        00046         data          26790
02BC 6A6D        00047         data          27245
02BD 6C23        00048         data          27683
02BE 6DC9        00049         data          28105
02BF 6F5E        00050         data          28510
02C0 70E2        00051         data          28898
02C1 7254        00052         data          29268
02C2 73B5        00053         data          29621
02C3 7504        00054         data          29956
```

© 1997 Microchip Technology Inc.

```
02C4 7641          00055        data          30273
02C5 776B          00056        data          30571
02C6 7884          00057        data          30852
02C7 7989          00058        data          31113
02C8 7A7C          00059        data          31356
02C9 7B5C          00060        data          31580
02CA 7C29          00061        data          31785
02CB 7CE3          00062        data          31971
02CC 7D89          00063        data          32137
02CD 7E1D          00064        data          32285
02CE 7E9C          00065        data          32412
02CF 7F09          00066        data          32521
02D0 7F61          00067        data          32609
02D1 7FA6          00068        data          32678
02D2 7FD8          00069        data          32728
02D3 7FF5          00070        data          32757
                   00071 ;
02D4               00072 CosTable
                   00073 ;
02D4 7FFF          00074        data          32767
02D5 7FF5          00075        data          32757
02D6 7FD8          00076        data          32728
02D7 7FA6          00077        data          32678
02D8 7F61          00078        data          32609
02D9 7F09          00079        data          32521
02DA 7E9C          00080        data          32412
02DB 7E1D          00081        data          32285
02DC 7D89          00082        data          32137
02DD 7CE3          00083        data          31971
02DE 7C29          00084        data          31785
02DF 7B5C          00085        data          31580
02E0 7A7C          00086        data          31356
02E1 7989          00087        data          31113
02E2 7884          00088        data          30852
02E3 776B          00089        data          30571
02E4 7641          00090        data          30273
02E5 7504          00091        data          29956
02E6 73B5          00092        data          29621
02E7 7254          00093        data          29268
02E8 70E2          00094        data          28898
02E9 6F5E          00095        data          28510
02EA 6DC9          00096        data          28105
02EB 6C23          00097        data          27683
02EC 6A6D          00098        data          27245
02ED 68A6          00099        data          26790
02EE 66CF          00100        data          26319
02EF 64E8          00101        data          25832
02F0 62F1          00102        data          25329
02F1 60EB          00103        data          24811
02F2 5ED7          00104        data          24279
02F3 5CB3          00105        data          23731
02F4 5A82          00106        data          23170
02F5 5842          00107        data          22594
02F6 55F5          00108        data          22005
02F7 539B          00109        data          21403
02F8 5133          00110        data          20787
02F9 4EBF          00111        data          20159
02FA 4C3F          00112        data          19519
02FB 49B4          00113        data          18868
02FC 471C          00114        data          18204
02FD 447A          00115        data          17530
02FE 41CE          00116        data          16846
02FF 3F17          00117        data          16151
0300 3C56          00118        data          15446
0301 398C          00119        data          14732
0302 36BA          00120        data          14010
```


© 1997 Microchip Technology Inc.

```
0303 33DF          00121       data          13279
0304 30FB          00122       data          12539
0305 2E11          00123       data          11793
0306 2B1F          00124       data          11039
0307 2826          00125       data          10278
0308 2528          00126       data          9512
0309 2223          00127       data          8739
030A 1F1A          00128       data          7962
030B 1C0B          00129       data          7179
030C 18F9          00130       data          6393
030D 15E2          00131       data          5602
030E 12C8          00132       data          4808
030F 0FAB          00133       data          4011
0310 0C8C          00134       data          3212
0311 096A          00135       data          2410
0312 0648          00136       data          1608
0313 0324          00137       data          804
0314 0000          00138       data          0
0315 FCDC          00139       data         -804
0316 F9B8          00140       data         -1608
0317 F696          00141       data         -2410
0318 F374          00142       data         -3212
0319 F055          00143       data         -4011
031A ED38          00144       data         -4808
031B EA1E          00145       data         -5602
031C E707          00146       data         -6393
031D E3F5          00147       data         -7179
031E E0E6          00148       data         -7962
031F DDDD          00149       data         -8739
0320 DAD8          00150       data         -9512
0321 D7DA          00151       data         -10278
0322 D4E1          00152       data         -11039
0323 D1EF          00153       data         -11793
0324 CF05          00154       data         -12539
0325 CC21          00155       data         -13279
0326 C946          00156       data         -14010
0327 C674          00157       data         -14732
0328 C3AA          00158       data         -15446
0329 C0E9          00159       data         -16151
032A BE32          00160       data         -16846
032B BB86          00161       data         -17530
032C B8E4          00162       data         -18204
032D B64C          00163       data         -18868
032E B3C1          00164       data         -19519
032F B141          00165       data         -20159
0330 AECD          00166       data         -20787
0331 AC65          00167       data         -21403
0332 AA0B          00168       data         -22005
0333 A7BE          00169       data         -22594
0334 A57E          00170       data         -23170
0335 A34D          00171       data         -23731
0336 A129          00172       data         -24279
0337 9F15          00173       data         -24811
0338 9D0F          00174       data         -25329
0339 9B18          00175       data         -25832
033A 9931          00176       data         -26319
033B 975A          00177       data         -26790
033C 9593          00178       data         -27245
033D 93DD          00179       data         -27683
033E 9237          00180       data         -28105
033F 90A2          00181       data         -28510
0340 8F1E          00182       data         -28898
0341 8DAC          00183       data         -29268
0342 8C4B          00184       data         -29621
0343 8AFC          00185       data         -29956
0344 89BF          00186       data         -30273
```

© 1997 Microchip Technology Inc.

```
0345 8895           00187       data          -30571
0346 877C           00188       data          -30852
0347 8677           00189       data          -31113
0348 8584           00190       data          -31356
0349 84A4           00191       data          -31580
034A 83D7           00192       data          -31785
034B 831D           00193       data          -31971
034C 8277           00194       data          -32137
034D 81E3           00195       data          -32285
034E 8164           00196       data          -32412
034F 80F7           00197       data          -32521
0350 809F           00198       data          -32609
0351 805A           00199       data          -32678
0352 8028           00200       data          -32728
0353 800B           00201       data          -32757
                    00202 ;
                    00203 ;***********************************************
                    00204
                    00384 ;
                    00385 ;*****************************************************************
                    00386 ;       FFT Input/Output Data Stored In External RAM
                    00387 ; Operate Processor In Extended Microcontroller Mode
                    00388 ; External Data Starts at Address 0x0800, with 2 bytes of
                    00389 ; Real Data followed by 2 bytes of Imaginary Data.
                    00390 ;*****************************************************************
0800                00391         ORG     EXT_RAM_START_ADDR
                    00392 ;
0800                00393 ExtRamAddr
                    00394 ;
                    00395         END
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0300 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0340 : XXXXXXXXXXXXXXXX XXXX------------ ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:   852


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


© 1997 Microchip Technology Inc.

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

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

Trademarks

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

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

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

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

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

Printed on recycled paper.

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

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

WORLDWIDE SALES AND SERVICE

AMERICAS

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

EUROPE

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

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

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

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

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

03/01/02

© 2002 Microchip Technology Inc.