AN543

Tone Generation

Author: Amar Palacherla
Microchip Technology Inc.

INTRODUCTION

A general purpose resonator routine is implemented using a PIC17C42. This routine is used to generate multiple tones. A tone signal is normally generated using extensive table lookup schemes. When a multiple tone signal is desired, each tone must have its own lookup table, thus requiring a large amount of storage space, especially when various frequencies are to be generated. This application note implements tone generation using recursive techniques. The algorithm for a resonator is developed and implemented using PIC17C42.

THEORY

Generation of a single tone basically implies generating samples of a sine/cosine wave. The Z-Transform of a sine wave is given as follows:

$$Z\{sin(wt)\} = \frac{Y(z)}{X(z)} = \frac{z^* sin(wT)}{z^2 - 2^* z^* cos(wT) + 1}$$

The impulse response of the above transform (i.e., for X(z) = 1) will generate a sine wave of frequency w sampled at a rate of T (= 1/fs). Thus the above equation is translated to:

$$Y(z) = \frac{z^{-1*} sin(wT)}{1 - 2^* z^{-1*} cos(wT) + z^2}$$

The above equation can be rewritten in a difference equation form as follows:

$y(n) - 2y(n-1)\cos(wT) + y(n-2) = x(n-1)sin(wT)$

Rearranging the above equation and setting, x(n) as an impulse sequence, the following recursive equations are obtained.

*y(n) = 2*K1*y(n-1)-y(n-2)*

y(n-2) = y(n-1)

y(n-1) = y(n)

with the following conditions:

K1 = cos(wT)

K2 = initial y(n-1) = sin(wT)

K3 = initial y(n-2) = 0

IMPLEMENTATION

The above developed algorithm is implemented as a subroutine using a PIC17C42. All computations are performed using double precision arithmetic (16/32-bits). The recursive tone generation algorithm is implemented as a subroutine (labeled as `Resonator`). This subroutine generates samples of a single tone. To generate multiple frequencies, simply call this resonator routine for the desired frequencies and sum the outputs. The three tone co-efficients are stored in program memory and are read into the data memory using `TABLRD` instructions.

The fully commented code is listed in Appendix A. The timing and memory requirements are included in the comment sections of the code. 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 the application note DS00540. This code can be easily modified and used in various applications like DTMF generation, sound generation, etc. The tones generated can easily be output to an on-chip PWM channel which in turn can drive a speaker for producing various sounds. If using a PWM channel, set the PWM frequency much higher than the sampling frequency used (in the example code, for 8 kHz sampling frequency, use at least 20 kHz PWM frequency).

As an example, a dual tone is generated and the resulting digital wave form is analyzed. The main program calls the function `Resonator` twice for generating the two desired tones and the two outputs are summed. A sampling frequency of 8 kHz was used to generate a dual tone of 800 Hz and 1.10 kHz. The resulting wave form is shown in Figure 1. The spectrum of the signal shown in Figure 2 shows two peaks corresponding to the two desired tones (800 Hz and 1.10 kHz). The assembly code was tested using PICMASTER® (Microchip's Universal In-Circuit Emulator System).

The generated tones were captured into the PICMASTER's trace buffer and then transferred to Microsoft® Excel® using Dynamic Data Exchange (DDE). Once the data is in Excel it is analyzed using Excel's FFT utility.

FIGURE 1: DUAL TONE WAVE FORM CAPTURED BY PICMASTER

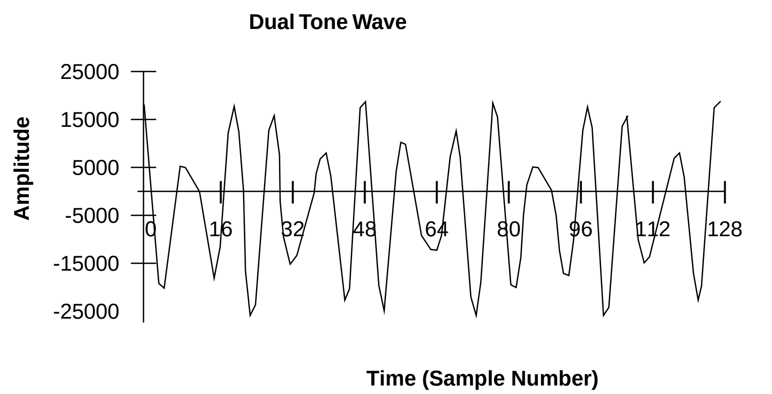
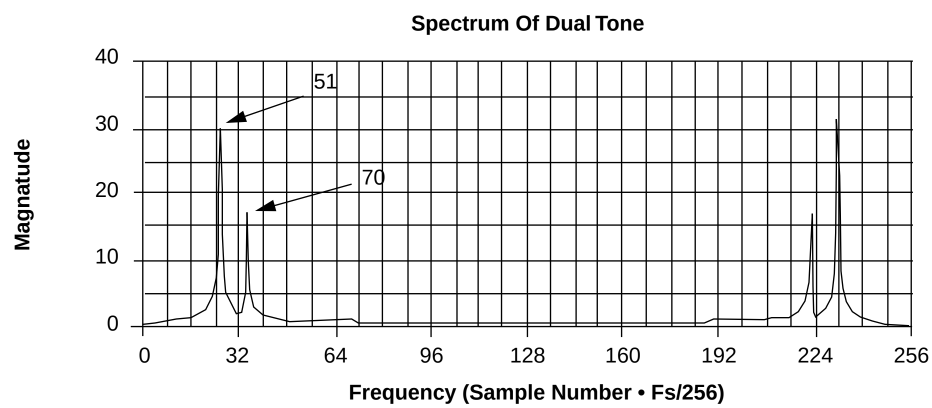


FIGURE 2: FREQUENCY SPECTRUM SHOWING THE DUAL TONE FREQUENCIES



PERFORMANCE

Table 1 below provides the performance of the resonator (labeled in the source code as `Resonator`) in terms of both timing and memory requirements. Since a double precision multiplier is used (software implementation), the multiplier timing is not always constant. Therefore the timings are given for the worst case. Note that irrespective of the frequency and the sampling (resolution) of the tone, program memory requirements is only 54 locations which in case of table lookup could be very large.

TABLE 0-1:

Cycles	235 Cycles
Time @ 16 MHz	58.75 µs
Time @ 25 MHz	47 µs
Data Memory	9 + 9* (# of tones to be generated)
Program memory	54 locations

APPLICATIONS

Tone generation is required in many applications. The code provided in this application note may be used as a general purpose routine to generate desired tones. It can be used in applications involving secure off-site control, where commands/data in the format of tones are transmitted over a telephone line. The tone generation finds applications involving signal modulations as well. The routine can be used to generate audible tones and output to a speaker connected to an I/O Port or a PWM channel.

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: TONE.ASM

```
MPASM 01.40 Released            TONE.ASM   1-16-1997  15:04:23         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001         LIST    P=17C42, columns=120,WRAP, R=DEC
                    00002 ;
                    00003 ;***************************************************************
                    00004 ;                       Dual Tone Generation
                    00005 ;
                    00006 ;       A generic resonator subroutine is implemented to generate
                    00007 ; tones. Samples Of A Sin/Cos Wave are generated using
                    00008 ; recursive techniques. Table Lookups are thus avoided
                    00009 ; This is especially useful in generating multiple tones
                    00010 ; (e.g. DTMF tone generation, tone signalling, etc), or programmable
                    00011 ; tone generation which may vary for each application unit.
                    00012 ;
                    00013 ;
                    00014 ;       Program:          TONE.ASM
                    00015 ;       Revision Date:
                    00016 ;                         1-13-97      Compatibility with MPASMWIN 1.40
                    00017 ;
                    00018 ;*************************************************************
                    00019 ;
                    00020         include "p17c42.inc"
                    00001         LIST
                    00002 ;P17C42.INC Standard Header File, Ver. 1.03 Microchip Technology, Inc.
                    00264         LIST
                    00021
                    00022         #define TRUE    1
                    00023         #define FALSE   0
                    00024         #define LSB     0
                    00025         #define MSB     7
                    00026
                    00027         include "17c42.mac"
                    00945         LIST
                    00028 ;
                    00029 ;*******************************************************************
                    00030 ;       TBLADDR
                    00031 ;
                    00032 ; DESCRIPTION:
                    00033 ;       Load 16 bit table pointer with specified label
                    00034 ;
                    00035 ; TIMING (in cycles):
                    00036 ;       4
                    00037 ;
                    00038
                    00039 TBLADDR MACRO   label
                    00040
                    00041         MOVLW   (label)
                    00042         MOVWF   TBLPTRL
                    00043         MOVLW   page    (label)
                    00044         MOVWF   TBLPTRH
                    00045
                    00046         ENDM
                    00047
                    00048 ;********************************************************************
                    00049 ;       ADDLBL
                    00050 ;
```

```
                 ; DESCRIPTION:
                 ;       Add A Label (16 bit constant) To A File Register (16 bit)
                 ;
                 ; TIMING (in cycles):
                 ;       4
                 ;

                 ADDLBL  MACRO   label,f

                         MOVLW   (label)
                         ADDWF   f+BB0, F
                         MOVLW   page    (label)
                         ADDWFC  f+BB1, F

                         ENDM

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

                         LIST
                 ;
                         CBLOCK  0
00000000                         BB0,BB1,BB2,BB3 ; RAM offset constants
                         ENDC
                 ;
                         CBLOCK  0x18
00000018                  AARG,AARG1                    ; 16 bit multiplier A
0000001A                  BARG,BARG1                    ; 16 bit multiplicand B
0000001C                  DPX,DPX1,DPX2,DPX3            ; 32 bit multiplier result = A*B
                          ENDC
                 ;
                         CBLOCK                         ; Generic Resonator Variables
00000020                  K1,K11                        ; K1
00000022                  SinCos_2C,SinCos_2,SinCos1_2  ; y(n-2), Init Value = K3
00000025                  SinCos_1,SinCos1_1            ; y(n-1), Init Value = K2
00000027                  SinCos,SinCos1                ; y(n)

                         ENDC

                         CBLOCK
00000029                  tone1K1,tone1K11              ; Tone 1 variables
0000002B                  tone1_2C,tone1_2,tone11_2
0000002E                  tone1_1,tone11_1
00000030                         tone1,tone11

00000032                  tone2K1,tone2K11             ; Tone 2 variables
00000034                  tone2_2C,tone2_2,tone21_2
00000037                  tone2_1,tone21_1
00000039                  tone2,tone21

0000003B                   dualTone,dualTone1          ; tone1+tone2
                         ENDC

                 ;
                 ;************************************************************
                 ;
                 #define ADD_OFFSET  FALSE
                 ;
                 #define coeff_addr  0x0030  ; prog mem addr of sine wave frequency,etc
                 #define DummyAddr   0x0A00
                 ;
                 ;******************************************************************
                 ;               Test Routine For Sin Wave Generation
                 ;
                 ;       (a) Call Initialization Subroutine to read desired sine wave
                 ;               freq from prog mem to data mem.
                 ; (b) Call the resonator subroutine to generate samples of desired
```

```
                   ; sine wave.
                   ; (c) Perform a Dummy Table Write of The Sine Wave Data, so that
                   ; PIC-MASTER emulator can capture the data into trace buffer
                   ;
                   ;*****************************************************************
                   ;
0000                       ORG     0x0000
0000 C040                  goto    start
                   ;
                   ;*****************************************************************
                   ;               Resonator Co-efficients For Desired Tones

0030                  ORG     coeff_addr

                   ; Tone 1 Resonator Constants
                   ; Sample Rate = fs = 8 khz, Tone Freq = f = 0.800 Khz
                   ;
0030 678E            DATA    26510  ; K1 = COS(wT) = COS(360*f/fs)
0031 259E            DATA    9630   ; K2 = SIN(wT) = SIN(360*f/fs) = init value of y(n-1)
0032 0000            DATA    0      : K3 is init value of y(n-2)
                   ;
                   ; Tone 2 Resonator Constants
                   ; Sample Rate = fs = 8 khz, Tone Freq = f = 1.10 Khz
                   ;
0033 5321            DATA    21281  ; K1 = COS(wT) = COS(360*f/fs)
0034 1855            DATA   6229    ; K2 = SIN(wT) = SIN(360*f/fs) = init value of y(n-1)
0035 0000            DATA    0      ; K3 is init value of y(n-2)
                   ;
                   ;*****************************************************************
                   ;
0040                 ORG     0x0040
0040               start
0040 E057             call    Coeff_Read
                   ; load table pointers with a dummy  addr for PIC-MASTER data capture
                           MOVK16  DummyAddr,TBLPTRL
                 M
0041 B000        M   MOVLW   (0x0A00) & 0xff
0042 010D        M   MOVWF   TBLPTRL+B0
0043 B00A        M   MOVLW   ((0x0A00) >> 8)
0044 010E        M   MOVWF   TBLPTRL+B1
                 M
0045               NextSample
0045 0000            nop
0046 A43B            tlwt    0,dualTone
0047               capture
0047 AE3C            tablwt  1,0,dualTone+BB1   ; for PIC-MASTER trace capture
0048 0000            nop
                   ;
0049 B029            movlw   tone1K1            ; load indirect address pointer
004A 0101            movwf   FSR0               ; for Tone 1
004B E06C            call    Resonator          ; Compute next sample of tone 1

004C B032            movlw   tone2K1            ; load indirect address pointer
004D 0101            movwf   FSR0               ; for Tone 2
004E E06C            call    Resonator          ; compute next sample of tone 2
                   ;
                   ; Compute Tone1 + Tone2 for dual tone generation
                   ;
                     ADD16ACC tone1,tone2,dualTone  ; dualTone = tone1 + tone2
                 M
004F 6A30        M   movfp   tone1+B0,WREG
0050 0E39        M   addwf   tone2+B0,W
0051 013B        M   movwf   dualTone+B0
0052 6A31        M   movfp   tone1+B1,WREG
0053 103A        M   addwfc  tone2+B1,W
0054 013C        M   movwf   dualTone+B1
```

```
                     M
                       ;
0055 C045                 goto    NextSample
                       ;
0056 C056              self  goto self
                       ;
                       ;*****************************************************************
                       ; Initialization routine :
                       ; Read Tone 1 & Tone 2 Resonator Frequencies from Program Memory To
                       ; Data Memory
                       ;
                       ;       Program Memory  :       3 + 8*(#of tones to be generated)
                       ;       Timing          :       4 + 11*(#of tones to be generated)
                       ;**************************************************************

0057                   Coeff_Read

                          TBLADDR coeff_addr
                     M
0057 B030            M    MOVLW   (0x0030)
0058 010D            M    MOVWF   TBLPTRL
0059 B000            M    MOVLW   page    (0x0030)
005A 010E            M    MOVWF   TBLPTRH
                     M
005B A929                 tablrd  0,1,tone1K1
005C A029                 tlrd    0,tone1K1
005D AB2A                 tablrd  1,1,tone1K1+BB1 ; read K1
005E A02E                 tlrd    0,tone1_1
005F AB2F                 tablrd  1,1,tone1_1+BB1 ; read K2
0060 A02C                 tlrd    0,tone1_2
0061 A22D                 tlrd    1,tone1_2+BB1   ; read K3
0062 292B                 clrf    tone1_2C, F

0063 A932                 tablrd  0,1,tone2K1
0064 A032                 tlrd    0,tone2K1
0065 AB33                 tablrd  1,1,tone2K1+BB1 ; read K1
0066 A037                 tlrd    0,tone2_1
0067 AB38                 tablrd  1,1,tone2_1+BB1 ; read K2
0068 A035                 tlrd    0,tone2_2
0069 A236                 tlrd    1,tone2_2+BB1   ; read K3
006A 2934                 clrf    tone2_2C, F

006B 0002                 return
                       ;
                       ;**********************************************************
                       ;                       Resonator Subroutine
                       ;
                       ; Before calling this routine, load the indirect register, FSR0
                       ; with the starting RAM address of the desired Tone Variables,
                       ; ( eg. For Tone1 Generation, load FSR0 with “Tone1K1” address.
                       ;
                       ;
                       ;
                       ; Timing (worst case) :
                       ;               20 + 36 + (worst case multiplier time)
                       ;               = 56 + 179 = 235 cycles
                       ;               = 58.75 uS @ 16Mhz
                       ;               = 47.00 uS @ 20Mhz
                       ;               = 37.60 uS @ 25 Mhz
                       ;
                       ; Memory Requirements :
                       ;               Program Memory  : 54 locations
                       ;               Data Memory     : 9 + 9*(#of tones to be generated)
                       ;
                       ;***********************************************************
                       ;
```

```
006C                      Resonator

                          ;
                          ; Transfer tone variables to resonator's variables using indirect
                          ; addressing This is necessary for making the "Resonator" a generic
                          ; subroutine, so that the same code can be called for various
                          ; tones.
                          ; Indirect addressing mode can be used throught the code in this
                          ; subroutine, but is less efficient.
                          ;
006C 8D04                   bcf     ALUSTA,FS1
006D 8404                   bsf     ALUSTA,FS0             ; auto increment FSR0
006E 4020                   movpf   INDF0,K1+BB0
006F 4021                   movpf   INDF0,K1+BB1
0070 4022                   movpf   INDF0,SinCos_2C
0071 4023                   movpf   INDF0,SinCos_2+BB0
0072 4024                   movpf   INDF0,SinCos_2+BB1
0073 4025                   movpf   INDF0,SinCos_1+BB0
0074 4026                   movpf   INDF0,SinCos_1+BB1
0075 4027                   movpf   INDF0,SinCos+BB0
0076 4028                   movpf   INDF0,SinCos+BB1

                          ;
                          ; Compute 2*K1*y(n-1)
                          ;
                            MOVFP16 K1,BARG
                        M
0077 7A20               M   MOVFP   K1+B0,BARG+B0          ; move A(B0) to B(B0)
0078 7B21               M   MOVFP   K1+B1,BARG+B1          ; move A(B1) to B(B1)
                        M
                            MOVFP16 SinCos_1,AARG
                        M
0079 7825               M   MOVFP   SinCos_1+B0,AARG+B0    ; move A(B0) to B(B0)
007A 7926               M   MOVFP   SinCos_1+B1,AARG+B1    ; move A(B1) to B(B1)
                        M
007B E0A2                   call    DblMult
                            RLC32   DPX                    ; DPX = 2*K1*y(n-1)
                        M
007C 8804               M   BCF     ALUSTA,C
007D 1B1C               M   RLCF    DPX+B0, F
007E 1B1D               M   RLCF    DPX+B1, F
007F 1B1E               M   RLCF    DPX+B2, F
0080 1B1F               M   RLCF    DPX+B3, F
                        M
                          ;
                          ; subtract y(n-2)*(2**15)
                          ;
0081 2922                   clrf    SinCos_2C, F
                            RRC24   SinCos_2C
                        M
0082 1A24               M   RLCF    SinCos_2C+B2,W         ; move sign into carry bit
0083 1924               M   RRCF    SinCos_2C+B2, F
0084 1923               M   RRCF    SinCos_2C+B1, F
0085 1922               M   RRCF    SinCos_2C+B0, F
                        M
                            SUB24   SinCos_2C,DPX1         ; DPX = 2*K1*y(n-1) - y(n-2)
                        M
0086 6A22               M   MOVFP   SinCos_2C+B0,WREG      ; get lowest byte of a into w
0087 051D               M   SUBWF   DPX1+B0, F             ; sub lowest byte of b, save in b(B0)
0088 6A23               M   MOVFP   SinCos_2C+B1,WREG      ; get 2nd byte of a into w
0089 031E               M   SUBWFB  DPX1+B1, F             ; sub 2nd byte of b, save in b(B1)
008A 6A24               M   MOVFP   SinCos_2C+B2,WREG      ; get 3rd byte of a into w
008B 031F               M   SUBWFB  DPX1+B2, F             ; sub 3rd byte of b, save in b(B2)
                        M
```

```
                  00266   RLC24   DPX1                    ; adjust decimal point
                      M
008C 8804             M   BCF     ALUSTA,C
008D 1B1D             M   RLCF    DPX1+B0, F
008E 1B1E             M   RLCF    DPX1+B1, F
008F 1B1F             M   RLCF    DPX1+B2, F
                      M
                  00267 ;
                  00268 ; update past samples with newly computed values
                  00269 ;
                  00270   MOVPF16 DPX2,SinCos             ; y(n) = 2*K1*y(n-1) - y(n-2)
                      M
0090 5E27             M   MOVPF   DPX2+B0,SinCos+B0       ; move A(B0) to B(B0)
0091 5F28             M   MOVPF   DPX2+B1,SinCos+B1       ; move A(B1) to B(B1)
                      M
                  00271   MOV16   SinCos_1,SinCos_2       ; y(n-2) = y(n-1)
                      M
0092 6A25             M   MOVFP   SinCos_1+B0,WREG        ; get byte of a into w
0093 0123             M   MOVWF   SinCos_2+B0             ; move to b(B0)
0094 6A26             M   MOVFP   SinCos_1+B1,WREG        ; get byte of a into w
0095 0124             M   MOVWF   SinCos_2+B1             ; move to b(B1)
                      M
                  00272   MOVPF16 DPX2,SinCos_1           ; y(n-1) = y(n)
                      M
0096 5E25             M   MOVPF   DPX2+B0,SinCos_1+B0     ; move A(B0) to B(B0)
0097 5F26             M   MOVPF   DPX2+B1,SinCos_1+B1     ; move A(B1) to B(B1)
                      M
                  00273 ;
                  00274 ; Generation Of The Next Sample Of The Resonator (sine wave) is done.
                  00275 ; The 16 bit result is stored in location “SinCos” (low Byte) &
                  00276 ; “SinCos+1” (High Byte)
                  00277 ;
                  00278 ; write back all the computed values to respective tone variables
                  00279 ;
0098 0701         00280   decf    FSR0, F
0099 8D04         00281   bcf     ALUSTA,FS1
009A 8C04         00282   bcf     ALUSTA,FS0
009B 6028         00283   movfp   SinCos+BB1,INDF0
009C 6027         00284   movfp   SinCos+BB0,INDF0
009D 6026         00285   movfp   SinCos_1+BB1,INDF0
009E 6025         00286   movfp   SinCos_1+BB0,INDF0
009F 6024         00287   movfp   SinCos_2C+BB2,INDF0
00A0 6023         00288   movfp   SinCos_2C+BB1,INDF0
                  00289 ;
00A1 0002         00290   return
                  00291 ;
                  00292 ;*****************************************************************
                  00293 ;       Include Double Precision Multiplication Routine
                  00294 ;*****************************************************************
  00000001        00295 SIGNED  equ     TRUE
                  00296
                  00297    include “17c42mpy.mac”
                  00178    LIST
                  00298
                  00299 ;
                  00300    END
```

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

0000 : X--------------- ---------------- ---------------- XXXXXX----------
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 XXXX------------

All other memory blocks unused.

Program Memory Words Used:   379


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

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.

Printed on recycled paper.

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