



Tone Generation

INTRODUCTION

A general purpose resonator routine is implemented using 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 a 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 :

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

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:

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

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.

$$y(n) = 2 * K_1 * y(n-1) - y(n-2)$$

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

$$y(n-1) = y(n)$$

with the following conditions:

$$K_1 = \cos(wT)$$

$$K_2 = \text{initial } y(n-1) = \sin(wT)$$

$$K_3 = \text{initial } y(n-2) = 0$$

IMPLEMENTATION

The above developed algorithm is implemented as a subroutine using PIC17C42. All computations are performed using double precision arithmetic (16/32-bits). The recursive tone generation algorithm is implemented as a subroutine (labelled 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 `TABLERD` 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 ANDS00540. 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, it is suggested to 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 & 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.

Tone Generation

FIGURE 1 - DUAL TONE WAVE FORM CAPTURED BY PICMASTER

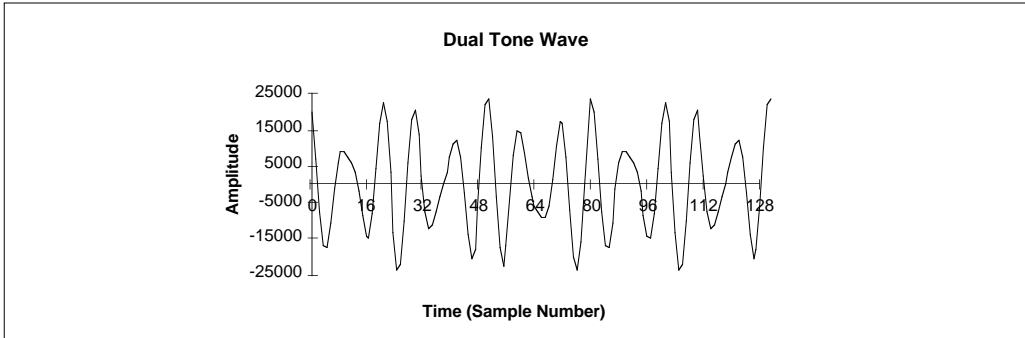
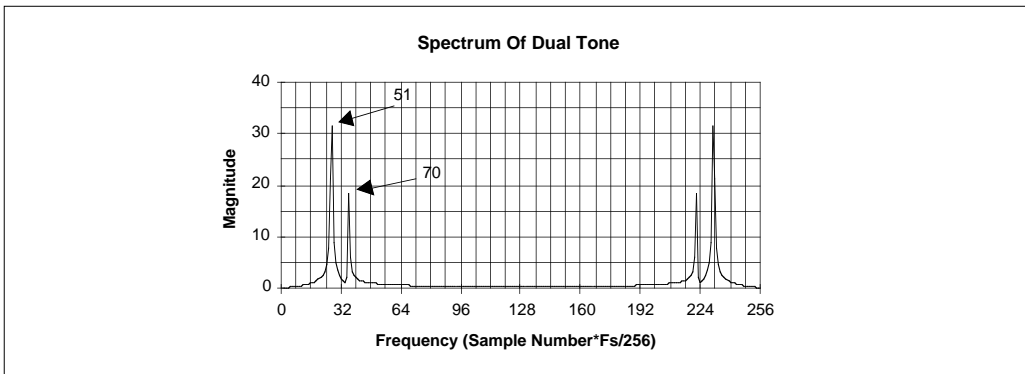


FIGURE 2 - FREQUENCY SPECTRUM OF FIG. 1.1 SHOWING THE DUAL TONE FREQUENCIES



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 1

| | |
|----------------|------------------------------------|
| Cycles | 235 Cycles |
| Time @ 16 MHz | 58.75 μ s |
| Time @ 25 MHz | 47 μ s |
| 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.

*Author: Amar Palacherla
Logic Products Division*

APPENDIX A: TONE.LST

MPASM B0.54

PAGE 1

```

;*****
;
;           Dual Tone Generation
;
;           A generic resonator subroutine is implemented to generate
; tones. Samples Of A Sin/Cos Wave are generated using
; recursive techniques. Table Lookups are thus avoided
; This is especially useful in generating multiple tones
; (e.g. DTMF tone generation, tone signalling, etc.), or programmable
; tone generation which may vary for each application unit.
;
;*****
;
;           LIST    P=17C42, C=80, T=ON, R=DEC, N=0
;           include "17c42.h"
;
;           include "17c42.mac"
;
;*****
;
;           TBLADDR
;
; DESCRIPTION:
;           Load 16 bit table pointer with specified label
;
; TIMING (in cycles):
;           4
;
TBLADDR MACRO    label

           MOVLW    (label)
           MOVWF    tblptrl
           MOVLW    page    (label)
           MOVWF    tblptrh

           ENDM

;*****
;
;           ADDLBL
;
; DESCRIPTION:
;           Add A Label (16 bit constant) To A File Register (1
;
; TIMING (in cycles):
;           4
;
ADDLBL MACRO    label,f

           MOVLW    (label)
           ADDWF    f+B0
           MOVLW    page    (label)
           ADDWFC   f+B1

           ENDM

```

Tone Generation

```
;*****  
  
;  
0000 0004      CBLOCK  0  
                B0,B1,B2,B3      ; RAM offset constants  
                ENDC  
;  
0018 0002      CBLOCK  0x18  
001A 0002      AARG,AARG1      ; 16 bit multiplier A  
001C 0004      BARG,BARG1      ; 16 bit multiplicand B  
                DPX,DPX1,DPX2,DPX3; 32 bit multiplier result = A*B  
  
                ENDC  
  
                ;  
                CBLOCK          ; Generic Resonator  
  
0020 0002      K1,K11          ; K1  
0022 0003      SinCos_2C,SinCos_2,SinCos1_2  
                ; y(n-2), Init Value = K3  
  
0025 0002      SinCos_1,SinCos1_1; y(n-1), Init Value = K2  
  
0027 0002      SinCos,SinCos1  ; y(n)  
0029 0000  
  
                ENDC  
  
                CBLOCK  
0029 0002      tone1K1,tone1K11 ; Tone 1 variables  
002B 0003      tone1_2C,tone1_2,tone11_2  
002E 0002      tone1_1,tone11_1  
0030 0002      tone1,tone11  
0032 0000  
0032 0002      tone2K1,tone2K11 ; Tone 2 variables  
0034 0003      tone2_2C,tone2_2,tone21_2  
0037 0002      tone2_1,tone21_1  
0039 0002      tone2,tone21  
003B 0000  
003B 0002      dualTone,dualTone1; tone1+tone2  
                ENDC  
  
;*****  
  
005A          #define ADD_OFFSET  FALSE  
  
005B          #define coeff_addr  0x0030 ; prog mem addr of  
  
005C          #define DummyAddr   0x0A00  
  
;*****  
  
;                Test Routine For Sin Wave Generation  
;  
;                (a) Call Initialization Subroutine to read desired s  
  
;                freq from prog mem to data mem.  
;                (b) Call the resonator subroutine to generate sampl  
  
;                sine wave.  
;                (c) Perform a Dummy Table Write of The Sine Wave Da  
  
;                PICMASTER emulator can capture the data in  
  
;  
;*****
```

```

                                ORG     0x0000
0000 C040      goto     start

;*****

                                ;Resonator Co-efficients For Desired Tones

                                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/
0031 259E      DATA    9630     ; K2 = SIN(wT) = SIN(360*f/
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/
0034 1855      DATA    6229     ; K2 = SIN(wT) = SIN(360*f/
0035 0000      DATA    0         ; K3 is init value of y(n-2

;*****

start ORG     0x0040
0040 E057      call     Coeff_Read          ; load table pointers with a dummy addr

                                MOVK16 DummyAddr,tblptr1

0041 B000      MOVLW   (0x0A00) & 0xff
0042 010D      MOVWF   tblptr1+B0
0043 B00A      MOVLW   ((0x0A00) >> 8)
0044 010E      MOVWF   tblptr1+B1

NextSample
0045 0000      nop
0046 A43B      tlwt    0,dualTone
capture
0047 AE3C      tablwt  1,0,dualTone+B1    ; for PICMASTER tr
0048 0000      nop
0049 B029      movlw   tone1K1          ; load indirect add
004A 0101      movwf   fsr0          ; for Tone 1
004B E06C      call    Resonator        ; Compute next samp

004C B032      movlw   tone2K1          ; load indirect add
004D 0101      movwf   fsr0          ; for Tone 2
004E E06C      call    Resonator        ; compute next samp

                                ; Compute Tone1 + Tone2 for dual tone

                                ADD16ACC tone1,tone2,dualTone

```

Tone Generation

```
                                ; dualTone = tone1
004F 6030                movfp  tone1+B0,wreg
0050 0E39                addwf  tone2+B0,w
0051 013B                movwf  dualTone+B0
0052 6031                movfp  tone1+B1,wreg
0053 103A                addwfc tone2+B1,w
0054 013C                movwf  dualTone+B1
0055 C045                goto   NextSample

0056 C056                self   goto   self

;*****

; Initialization routine :
; Read Tone 1 & Tone 2 Resonator Frequencies from Program M

; Data Memory
;
;   Program Memory :      3 + 8*(#of tones to be gene
;
;   Timing          :      4 + 11*(#of tones to be gen
;*****
Coeff_Read

                                TBLADDR coeff_addr
0057 B030                MOVLW  (0x0030)
0058 010D                MOVWF  tblptrl
0059 B000                MOVLW  page   (0x0030)
005A 010E                MOVWF  tblptrh
005B A929                tablrd  0,1,tone1K1
005C A029                tlrld  0,tone1K1
005D AB2A                tablrd  1,1,tone1K1+B1 ; read K1
005E A02E                tlrld  0,tone1_1
005F AB2F                tablrd  1,1,tone1_1+B1 ; read K2
0060 A02C                tlrld  0,tone1_2
0061 A22D                tlrld  1,tone1_2+B1 ; read K3
0062 292B                clrf  tone1_2C
0063 A932                tablrd  0,1,tone2K1
0064 A032                tlrld  0,tone2K1
0065 AB33                tablrd  1,1,tone2K1+B1 ; read K1
0066 A037                tlrld  0,tone2_1
0067 AB38                tablrd  1,1,tone2_1+B1 ; read K2
0068 A035                tlrld  0,tone2_2
0069 A236                tlrld  1,tone2_2+B1 ; read K3
006A 2934                clrf  tone2_2C
006B 0002                return

;*****
;                               Resonator Subroutine
;
; Before calling this routine, load the indirect register,
; with the starting RAM address of the desired Tone Variabl
; ( eg. For Tone1 Generation, load FSR0 with "Tone1K1" addr
;
;
;
; 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 gen
;*****
```

```

;
Resonator
;
; Transfer tone variables to resonator's variables using i
; This is necessary for making the "Resonator" a generic
; subroutine, so that the same code can be called for vario
; tones.
; Indirect addressing mode can be used throught the code i
; subroutine, but is less efficient.
;
006C 8D04          bcf     _fs1
006D 8404          bsf     _fs0                ; auto increment FSR0
006E 4020          movpf  indf0,K1+B0
006F 4021          movpf  indf0,K1+B1
0070 4022          movpf  indf0,SinCos_2C
0071 4023          movpf  indf0,SinCos_2+B0
0072 4024          movpf  indf0,SinCos_2+B1
0073 4025          movpf  indf0,SinCos_1+B0
0074 4026          movpf  indf0,SinCos_1+B1
0075 4027          movpf  indf0,SinCos+B0
0076 4028          movpf  indf0,SinCos+B1
;
; Compute 2*K1*y(n-1)
;
MOVFP16 K1,BARG
MOVFP K1+B0,BARG+B0        ; move K1(B0) t
MOVFP K1+B1,BARG+B1        ; move K1(B1) t
MOVFP16 SinCos_1,AARG
MOVFP SinCos_1+B0,AARG+B0; move Si
MOVFP SinCos_1+B1,AARG+B1; move Si
call Db1Mult
007C 8804          BCF     _carry
007D 1B1C          RLCF   DPX+B0
007E 1B1D          RLCF   DPX+B1
007F 1B1E          RLCF   DPX+B2
0080 1B1F          RLCF   DPX+B3                ;
; subtract y(n-2)*(2**15)
0081 2922          clrf   SinCos_2C
RRC24 SinCos_2C
0082 1A24          RLCF   SinCos_2C+B2,W        ; move sign
0083 1924          RRCF   SinCos_2C+B2
0084 1923          RRCF   SinCos_2C+B1
0085 1922          RRCF   SinCos_2C+B0
SUB24 SinCos_2C,DPX1        ; DPX = 2*K1*y(n-1) - y(n-
0086 6022          MOVFP  SinCos_2C+B0,wreg    ; get lowes
0087 051D          SUBWF  DPX1+B0              ; sub lowest byt
0088 6023          MOVFP  SinCos_2C+B1,wreg    ; get 2nd b
0089 031E          SUBWFB DPX1+B1              ; sub 2nd byte o
008A 6024          MOVFP  SinCos_2C+B2,wreg    ; get 3rd b
008B 031F          SUBWFB DPX1+B2              ; sub 3rd byte o
RRC24 DPX1                    ; adjust decimal point
008C 8804          BCF     _carry
008D 1B1D          RLCF   DPX1+B0
008E 1B1E          RLCF   DPX1+B1
008F 1B1F          RLCF   DPX1+B2
;
; update past samples with newly computed values
;
MOVFP16 DPX2,SinCos        ; y(n) = 2*K1*y(n-1) - y(n
0090 5E27          MOVFP  DPX2+B0,SinCos+B0    ; move DPX2
0091 5F28          MOVFP  DPX2+B1,SinCos+B1    ; move DPX2
MOV16 SinCos_1,SinCos_2    ; y(n-2) = y(n-1)
0092 6025          MOVFP  SinCos_1+B0,wreg    ; get byte o
0093 0123          MOVWF  SinCos_2+B0          ; move to Si
0094 6026          MOVFP  SinCos_1+B1,wreg    ; get byte o
0095 0124          MOVWF  SinCos_2+B1          ; move to Si
MOVFP16 DPX2,SinCos_1        ; y(n-1) = y(n)
0096 5E25          MOVFP  DPX2+B0,SinCos_1+B0; move DP
0097 5F26          MOVFP  DPX2+B1,SinCos_1+B1; move DP

```

Tone Generation

```
;
; Generation Of The Next Sample Of The Resonator (sine wave
; The 16 bit result is stored in location "SinCos" (low Byt
; "SinCos+1" (High Byte)
;
; write back all the computed values to respective tone var
;
0098 0701          decf    fsr0
0099 8D04          bcf     _fsl
009A 8C04          bcf     _fs0
009B 6028          movfp  SinCos+B1,indf0
009C 6027          movfp  SinCos+B0,indf0
009D 6026          movfp  SinCos_1+B1,indf0
009E 6025          movfp  SinCos_1+B0,indf0
009F 6024          movfp  SinCos_2C+B2,indf0
00A0 6023          movfp  SinCos_2C+B1,indf0          ;
00A1 0002          return

;
;*****

;          Include Double Precision Multiplication Routine
;*****
0001  SIGNED      equ     TRUE
          include  "17c42mpy.mac"
;          NOLIST
;*****
;          Double Precision Multiplier For PIC17C42
;
;          Dmult
;
; DESCRIPTION:
;
; Multiplication : AARG (16 bits) * BARG (16 bits) -> DPX
;
; (a) Load the 1st operand in locations AARG+B0 & AARG
; (b) Load the 2nd operand in locations BARG+B0 & BARG
; (c) CALL Dmult
; (d) The 32 bit result is in locations ( DPX+B0,DPX+B
;
; In the signed case, a savings of 9 clks can be real
; BARG as the positive factor in the product when pos
;
; TIMING (worst case):
;          unsigned:          17
;
;          signed:          BARG+ 17
;          BARG- 17
;
; NOTE : Define SIGNED/UNSIGNED To 1/0 before including
; this file in your program
;
;*****
;          Multiplication Macro
;*****
; TIMING:          unsigned:    11+7*10+8*11 = 169 clks
;(worst case)    signed:      11+7*10+7*11+5 = 163 clks
;
; MULTMAC  MACRO
;          variable i
;          i = 0
;          if SIGNED
;          while i < 15
;          else
;          while i < 16
;          endif
;          if i < 8          ; test low byte
```



```

        btfsc    BARG+B0,i
        else
            ; test high byte
        btfsc    BARG+B1,i-8
    fi
    goto    add#v(i)
if i < 8
    rlcfc    DPX+B3,W        ; rotate sign into carry bit
    rrcfc    DPX+B3        ; for i < 8, no meaningful bits
    rrcfc    DPX+B2        ; are in DPX+B0
    rrcfc    DPX+B1
else
    rlcfc    DPX+B3,W        ; rotate sign into carry bit
    rrcfc    DPX+B3
    rrcfc    DPX+B2
    rrcfc    DPX+B1
    rrcfc    DPX+B0
fi
    i = i+1
endw
    clrfc    DPX+B0        ; if we get here, BARG = 0
    return
        add0
    movfpc    AARG+B0,WREG
    addwfc    DPX+B2        ;add lsb
    movfpc    AARG+B1,WREG
    addwfc    DPX+B3        ;add msb
    rlcfc    AARG+B1,W        ; rotate sign into carry bit
    rrcfc    DPX+B3        ; for i < 8, no meaningful bits
    rrcfc    DPX+B2        ; are in DPX+B0
    rrcfc    DPX+B1

i = 1
if SIGNED
    while i < 15
else
    while i < 16
endif
    if i < 8
        btfssc    BARG+B0,i    ;test low byte
    else
        btfssc    BARG+B1,i-8    ; test high byte
    fi
    goto    noadd#v(i)
        add#v(i)
    movfpc    AARG+B0,WREG
    addwfc    DPX+B2        ;add lsb
    movfpc    AARG+B1,WREG
    addwfc    DPX+B3        ;add msb
        noadd#v(i)
    if i < 8
        rlcfc    AARG+B1,W        ; rotate sign into carry bit
        rrcfc    DPX+B3        ; for i < 8, no meaningful bits
        rrcfc    DPX+B2        ; are in DPX+B0
        rrcfc    DPX+B1
    else
        rlcfc    AARG+B1,W        ; rotate sign into carry bit
        rrcfc    DPX+B3
        rrcfc    DPX+B2
        rrcfc    DPX+B1
        rrcfc    DPX+B0
        fi
        i = i+1
    endw
if SIGNED
    rlcfc    AARG+B1,W        ; since BARG is always made posit
    rrcfc    DPX+B3        ; the last bit is known to be zer
    rrcfc    DPX+B2
    rrcfc    DPX+B1
    rrcfc    DPX+B0

```

Tone Generation

```
endif

        ENDM

;           Double Precision Multiply ( 16x16 -> 32 )
;           ( AARG*BARG -> : 32 bit output in DPX
;
        DblMult
    if    SIGNED

00A2 971B           btfss    BARG+B1,MSB           ; test sign of BARG

00A3 C0AE           goto     argsok                ; if positive, ok
        NEG16      AARG+B0           ; if negative, then negate

00A4 1318           COMF     AARG+B0+B0
00A5 1319           COMF     AARG+B0+B1
00A6 2900           CLRF    wreg
00A7 1518           INCF    AARG+B0+B0
00A8 1119           ADDWFC   AARG+B0+B1

        NEG16      BARG+B0           ; AARG and BARG

00A9 131A           COMF     BARG+B0+B0
00AA 131B           COMF     BARG+B0+B1
00AB 2900           CLRF    wreg
00AC 151A           INCF    BARG+B0+B0
00AD 111B           ADDWFC   BARG+B0+B1

endif

argsok
        CLR16     DPX+B2           ; clear initial partial pr

00AE 291E           CLRF    DPX+B2+B0
00AF 291F           CLRF    DPX+B2+B1

        MULTMAC           ; use macro for multiplica

0000
        variable i

0000
        i = 0

        if    SIGNED

        while i < 15

    else

        while i < 16

    endif
        if i < 8           ; test low byte

        btfsc     BARG+B0,i

    else
        btfsc     BARG+B1,i-8           ; test high byte
    fi

        goto     add#v(i)
        if i < 8
        rlcfc    DPX+B3,W           ; rotate sign into carry bit
        rrcfc    DPX+B3           ; for i < 8, no meaningful bits
```

```

        rrcf    DPX+B2    ; are in DPX+B0
        rrcf    DPX+B1
    else
        rlcfc   DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
    i = i+1
    endw
    if i < 8    ; test low byte
00B0 981A    btfscc   BARG+B0,i
    else      ; test high byte
        btfscc   BARG+B1,i-8
    fi
00B1 C113    goto     add0
    if i < 8
00B2 1A1F    rlcfc   DPX+B3,W  ; rotate sign into
00B3 191F    rrcf    DPX+B3    ; for i < 8, no mea
00B4 191E    rrcf    DPX+B2    ; are in DPX+B0
00B5 191D    rrcf    DPX+B1
    else
        rlcfc   DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
0001        i = i+1
    if i < 8    ; test low byte
00B6 991A    btfscc   BARG+B0,i
    else      ; test high byte
        btfscc   BARG+B1,i-8
    fi
00B7 C11D    goto     add1
    if i < 8
00B8 1A1F    rlcfc   DPX+B3,W  ; rotate sign into
00B9 191F    rrcf    DPX+B3    ; for i < 8, no mea
00BA 191E    rrcf    DPX+B2    ; are in DPX+B0
00BB 191D    rrcf    DPX+B1
    else
        rlcfc   DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
0002        i = i+1
    if i < 8    ; test low byte
00BC 9A1A    btfscc   BARG+B0,i
    else      ; test high byte
        btfscc   BARG+B1,i-8
    fi
00BD C127    goto     add2
    if i < 8
00BE 1A1F    rlcfc   DPX+B3,W  ; rotate sign into
00BF 191F    rrcf    DPX+B3    ; for i < 8, no mea
00C0 191E    rrcf    DPX+B2    ; are in DPX+B0
00C1 191D    rrcf    DPX+B1
    else
        rlcfc   DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
0003        i = i+1
    if i < 8    ; test low byte

```

Tone Generation

```
00C2 9B1A          btfsc      BARG+B0,i
                   else          ; test high byte
                   btfsc      BARG+B1,i-8

fi
00C3 C131          goto      add3
if i < 8
00C4 1A1F          rlcfc     DPX+B3,W   ; rotate sign into
00C5 191F          rrcfc     DPX+B3     ; for i < 8, no mea
00C6 191E          rrcfc     DPX+B2     ; are in DPX+B0
00C7 191D          rrcfc     DPX+B1
                   else
                   rlcfc     DPX+B3,W   ; rotate sign into carry bit
                   rrcfc     DPX+B3
                   rrcfc     DPX+B2
                   rrcfc     DPX+B1
                   rrcfc     DPX+B0

fi
0004              i = i+1
if i < 8          ; test low byte
00C8 9C1A          btfsc      BARG+B0,i
                   else          ; test high byte
                   btfsc      BARG+B1,i-8

fi
00C9 C13B          goto      add4
if i < 8
00CA 1A1F          rlcfc     DPX+B3,W   ; rotate sign into
00CB 191F          rrcfc     DPX+B3     ; for i < 8, no mea
00CC 191E          rrcfc     DPX+B2     ; are in DPX+B0
00CD 191D          rrcfc     DPX+B1
                   else
                   rlcfc     DPX+B3,W   ; rotate sign into carry bit
                   rrcfc     DPX+B3
                   rrcfc     DPX+B2
                   rrcfc     DPX+B1
                   rrcfc     DPX+B0

fi
0005              i = i+1
if i < 8          ; test low byte
00CE 9D1A          btfsc      BARG+B0,i
                   else          ; test high byte
                   btfsc      BARG+B1,i-8

fi
00CF C145          goto      add5
if i < 8
00D0 1A1F          rlcfc     DPX+B3,W   ; rotate sign into
00D1 191F          rrcfc     DPX+B3     ; for i < 8, no mea
00D2 191E          rrcfc     DPX+B2     ; are in DPX+B0
00D3 191D          rrcfc     DPX+B1
                   else
                   rlcfc     DPX+B3,W   ; rotate sign into carry bit
                   rrcfc     DPX+B3
                   rrcfc     DPX+B2
                   rrcfc     DPX+B1
                   rrcfc     DPX+B0

fi
006              i = i+1
if i < 8          ; test low byte
00D4 9E1A          btfsc      BARG+B0,i
                   else          ; test high byte
                   btfsc      BARG+B1,i-8

fi
00D5 C14F          goto      add6
if i < 8
00D6 1A1F          rlcfc     DPX+B3,W   ; rotate sign into
00D7 191F          rrcfc     DPX+B3     ; for i < 8, no mea
00D8 191E          rrcfc     DPX+B2     ; are in DPX+B0
00D9 191D          rrcfc     DPX+B1
                   else
                   rlcfc     DPX+B3,W   ; rotate sign into carry bit
```

```

        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
0007      i = i+1
        if i < 8      ; test low byte
00DA 9F1A      btfsc  BARG+B0,i
                else  ; test high byte
        btfsc  BARG+B1,i-8
    fi
00DB C159      goto   add7
        if i < 8
00DC 1A1F      rrcf    DPX+B3,W  ; rotate sign into
00DD 191F      rrcf    DPX+B3      ; for i < 8, no mea
00DE 191E      rrcf    DPX+B2      ; are in DPX+B0
00DF 191D      rrcf    DPX+B1
        else
        rrcf    DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
0008      i = i+1
        if i < 8      ; test low byte
        btfsc  BARG+B0,i
                else  ; test high byte
00E0 981B      btfsc  BARG+B1,i-8
    fi
00E1 C163      goto   add8
        if i < 8
        rrcf    DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3      ; for i < 8, no meaningful bits
        rrcf    DPX+B2      ; are in DPX+B0
        rrcf    DPX+B1
    else
        rrcf    DPX+B3,W  ; rotate sign into
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
00E2 1A1F
00E3 191F
00E4 191E
00E5 191D
00E6 191C
    fi
0009      i = i+1
        if i < 8      ; test low byte
        btfsc  BARG+B0,i
                else  ; test high byte
00E7 991B      btfsc  BARG+B1,i-8
    fi
00E8 C16E      goto   add9
        if i < 8
        rrcf    DPX+B3,W  ; rotate sign into carry bit
        rrcf    DPX+B3      ; for i < 8, no meaningful bits
        rrcf    DPX+B2      ; are in DPX+B0
        rrcf    DPX+B1
    else
        rrcf    DPX+B3,W  ; rotate sign into
        rrcf    DPX+B3
        rrcf    DPX+B2
        rrcf    DPX+B1
        rrcf    DPX+B0
    fi
00E9 1A1F
00EA 191F
00EB 191E
00EC 191D
00ED 191C
    fi
000A      i = i+1
        if i < 8      ; test low byte
        btfsc  BARG+B0,i
                else  ; test high byte
00EE 9A1B      btfsc  BARG+B1,i-8
    fi
00EF C179      goto   add10

```

Tone Generation

```

        if i < 8
            rlcfc DPX+B3,W ; rotate sign into carry bit
            rrcfc DPX+B3 ; for i < 8, no meaningful bits
            rrcfc DPX+B2 ; are in DPX+B0
            rrcfc DPX+B1
        else
00F0 1A1F        rlcfc DPX+B3,W ; rotate sign into
00F1 191F        rrcfc DPX+B3
00F2 191E        rrcfc DPX+B2
00F3 191D        rrcfc DPX+B1
00F4 191C        rrcfc DPX+B0
        fi
000B            i = i+1
        if i < 8 ; test low byte
            btfsc BARG+B0,i
            else ; test high byte
00F5 9B1B        btfsc BARG+B1,i-8
        fi
00F6 C184        goto add11
        if i < 8
            rlcfc DPX+B3,W ; rotate sign into carry bit
            rrcfc DPX+B3 ; for i < 8, no meaningful bits
            rrcfc DPX+B2 ; are in DPX+B0
            rrcfc DPX+B1
        else
00F7 1A1F        rlcfc DPX+B3,W ; rotate sign into
00F8 191F        rrcfc DPX+B3
00F9 191E        rrcfc DPX+B2
00FA 191D        rrcfc DPX+B1
00FB 191C        rrcfc DPX+B0
        fi
000C            i = i+1
        if i < 8 ; test low byte
            btfsc BARG+B0,i
            else ; test high byte
00FC 9C1B        btfsc BARG+B1,i-8
        fi
00FD C18F        goto add12

        if i < 8
            rlcfc DPX+B3,W ; rotate sign into carry bit
            rrcfc DPX+B3 ; for i < 8, no meaningful bits
            rrcfc DPX+B2 ; are in DPX+B0
            rrcfc DPX+B1
        else
00FE 1A1F        rlcfc DPX+B3,W ; rotate sign into
00FF 191F        rrcfc DPX+B3
0100 191E        rrcfc DPX+B2
0101 191D        rrcfc DPX+B1
0102 191C        rrcfc DPX+B0
        fi
000D            i = i+1
        if i < 8 ; test low byte
            btfsc BARG+B0,i
            else ; test high byte
0103 9D1B        btfsc BARG+B1,i-8
        fi
0104 C19A        goto add13
        if i < 8
            rlcfc DPX+B3,W ; rotate sign into carry bit
            rrcfc DPX+B3 ; for i < 8, no meaningful bits
            rrcfc DPX+B2 ; are in DPX+B0
            rrcfc DPX+B1
        else
0105 1A1F        rlcfc DPX+B3,W ; rotate sign into
0106 191F        rrcfc DPX+B3
0107 191E        rrcfc DPX+B2
0108 191D        rrcfc DPX+B1
0109 191C        rrcfc DPX+B0
```

```

fi
000E      i = i+1
          if i < 8                ; test low byte
            btfsc BARG+B0,i
          else                    ; test high byte
            btfsc BARG+B1,i-8
          fi
010A 9E1B
          fi
010B C1A5      goto add14
          if i < 8
            rlcfc DPX+B3,W        ; rotate sign into carry bit
            rrcfc DPX+B3          ; for i < 8, no meaningful bits
            rrcfc DPX+B2          ; are in DPX+B0
            rrcfc DPX+B1
          else
            rlcfc DPX+B3,W        ; rotate sign into
            rrcfc DPX+B3
            rrcfc DPX+B2
            rrcfc DPX+B1
            rrcfc DPX+B0
          fi
010C 1A1F      rlcfc DPX+B3,W
010D 191F      rrcfc DPX+B3
010E 191E      rrcfc DPX+B2
010F 191D      rrcfc DPX+B1
0110 191C      rrcfc DPX+B0

000F          fi
          i = i+1
0111 291C      clrfc DPX+B0        ; if we get here, B
0112 0002      return
          add0
0113 6018      movfpc AARG+B0,WREG
0114 0F1E      addwfc DPX+B2        ;add lsb
0115 6019      movfpc AARG+B1,WREG
0116 111F      addwfc DPX+B3        ;add msb
0117 1A19      rlcfc AARG+B1,W    ; rotate sign into
0118 191F      rrcfc DPX+B3        ; for i < 8, no mea
0119 191E      rrcfc DPX+B2        ; are in DPX+B0
011A 191D      rrcfc DPX+B1
0001          i = 1
          if SIGNED
            while i < 15
          else
            while i < 16
          endif
          if i < 8
            btfssc BARG+B0,i        ;test low byte
          else
            btfssc BARG+B1,i-8     ; test high byte
          fi
          goto noadd#v(i)
          add#v(i)
          movfpc AARG+B0,WREG
          addwfc DPX+B2        ;add lsb
          movfpc AARG+B1,WREG
          addwfc DPX+B3        ;add msb
          noadd#v(i)
          if i < 8
            rlcfc AARG+B1,W        ; rotate sign into carry bit
            rrcfc DPX+B3          ; for i < 8, no meaningful bits
            rrcfc DPX+B2          ; are in DPX+B0
            rrcfc DPX+B1
          else
            rlcfc AARG+B1,W        ; rotate sign into carry bit
            rrcfc DPX+B3
            rrcfc DPX+B2
            rrcfc DPX+B1
            rrcfc DPX+B0
          fi
          fi
          i = i+1
          endw
          if i < 8                ;test low byte
            btfssc BARG+B0,i
          else                    ; test high byte
            btfssc BARG+B1,i-8
          fi
          btfssc BARG+B1,i-8

```

Tone Generation

```

                                fi
011C C121                goto    noadd1
                                add1
011D 6018                movfp   AARG+B0,WREG
011E 0F1E                addwf  DPX+B2    ;add lsb
011F 6019                movfp   AARG+B1,WREG
0120 111F                addwfc DPX+B3    ;add msb
                                noadd1
                                if i < 8
0121 1A19                rlcfc  AARG+B1,W  ; rotate sign into
0122 191F                rrcfc  DPX+B3    ; for i < 8, no mea
0123 191E                rrcfc  DPX+B2    ; are in DPX+B0
0124 191D                rrcfc  DPX+B1
                                else
                                rlcfc  AARG+B1,W  ; rotate sign into carry bit
                                rrcfc  DPX+B3
                                rrcfc  DPX+B2
                                rrcfc  DPX+B1
                                rrcfc  DPX+B0
                                fi
0002                    i = i+1
                                if i < 8                ;test low byte
0125 921A                btfss  BARG+B0,i
                                else                    ; test high byte
                                btfss  BARG+B1,i-8
                                fi
0126 C12B                goto    noadd2
                                add2
0127 6018                movfp   AARG+B0,WREG
0128 0F1E                addwf  DPX+B2    ;add lsb
0129 6019                movfp   AARG+B1,WREG
012A 111F                addwfc DPX+B3    ;add msb
                                noadd2
                                if i < 8
012B 1A19                rlcfc  AARG+B1,W  ; rotate sign into
012C 191F                rrcfc  DPX+B3    ; for i < 8, no mea
012D 191E                rrcfc  DPX+B2    ; are in DPX+B0
012E 191D                rrcfc  DPX+B1
                                else
                                rlcfc  AARG+B1,W  ; rotate sign into carry bit
                                rrcfc  DPX+B3
                                rrcfc  DPX+B2
                                rrcfc  DPX+B1
                                rrcfc  DPX+B0
                                fi
0003                    i = i+1
                                if i < 8                ;test low byte
012F 931A                btfss  BARG+B0,i
                                else                    ; test high byte
                                btfss  BARG+B1,i-8
                                fi
0130 C135                goto    noadd3
                                add3
0131 6018                movfp   AARG+B0,WREG
0132 0F1E                addwf  DPX+B2    ;add lsb
0133 6019                movfp   AARG+B1,WREG
0134 111F                addwfc DPX+B3    ;add msb
                                noadd3
                                if i < 8
0135 1A19                rlcfc  AARG+B1,W  ; rotate sign into
0136 191F                rrcfc  DPX+B3    ; for i < 8, no mea
0137 191E                rrcfc  DPX+B2    ; are in DPX+B0
0138 191D                rrcfc  DPX+B1
                                else
                                rlcfc  AARG+B1,W  ; rotate sign into carry bit
                                rrcfc  DPX+B3
                                rrcfc  DPX+B2
                                rrcfc  DPX+B1
```



```

                                rrcf    DPX+B0
                                fi
0004      i = i+1
                                if i < 8      ;test low byte
0139 941A      btfs    BARG+B0,i
                                else
                                                ; test high byte
                                btfs    BARG+B1,i-8
                                fi
013A C13F      goto    noadd4
                                add4
013B 6018      movfp   AARG+B0,WREG
013C 0F1E      addwf   DPX+B2      ;add lsb
013D 6019      movfp   AARG+B1,WREG
013E 111F      addwfc  DPX+B3      ;add msb
                                noadd4
                                if i < 8
013F 1A19      rlcfc   AARG+B1,W    ; rotate sign into
0140 191F      rrcf    DPX+B3      ; for i < 8, no mea
0141 191E      rrcf    DPX+B2      ; are in DPX+B0
0142 191D      rrcf    DPX+B1
                                else
                                rlcfc   AARG+B1,W    ; rotate sign into carry bit
                                rrcf    DPX+B3
                                rrcf    DPX+B2
                                rrcf    DPX+B1
                                rrcf    DPX+B0
                                fi
0005      i = i+1
                                if i < 8      ;test low byte
0143 951A      btfs    BARG+B0,i
                                else
                                                ; test high byte
                                btfs    BARG+B1,i-8
                                fi
0144 C149      goto    noadd5
                                add5
0145 6018      movfp   AARG+B0,WREG
0146 0F1E      addwf   DPX+B2      ;add lsb
0147 6019      movfp   AARG+B1,WREG
0148 111F      addwfc  DPX+B3      ;add msb
                                noadd5
                                if i < 8
0149 1A19      rlcfc   AARG+B1,W    ; rotate sign into
014A 191F      rrcf    DPX+B3      ; for i < 8, no mea
014B 191E      rrcf    DPX+B2      ; are in DPX+B0
014C 191D      rrcf    DPX+B1
                                else
                                rlcfc   AARG+B1,W    ; rotate sign into carry bit
                                rrcf    DPX+B3
                                rrcf    DPX+B2
                                rrcf    DPX+B1
                                rrcf    DPX+B0
                                fi
0006      i = i+1
                                if i < 8      ;test low byte
014D 961A      btfs    BARG+B0,i
                                else
                                                ; test high byte
                                btfs    BARG+B1,i-8
                                fi
014E C153      goto    noadd6
                                add6
014F 6018      movfp   AARG+B0,WREG
0150 0F1E      addwf   DPX+B2      ;add lsb
0151 6019      movfp   AARG+B1,WREG
0152 111F      addwfc  DPX+B3      ;add msb
                                noadd6
                                if i < 8
0153 1A19      rlcfc   AARG+B1,W    ; rotate sign into
0154 191F      rrcf    DPX+B3      ; for i < 8, no mea

```

Tone Generation

```
0155 191E          rrcf    DPX+B2    ; are in DPX+B0
0156 191D          rrcf    DPX+B1
                    else
                    rrcf    AARG+B1,W  ; rotate sign into carry bit
                    rrcf    DPX+B3
                    rrcf    DPX+B2
                    rrcf    DPX+B1
                    rrcf    DPX+B0
                    fi
0007              i = i+1
                    if i < 8          ;test low byte

0157 971A          btfss   BARG+B0,i
                    else              ; test high byte
                    btfss   BARG+B1,i-8
                    fi

0158 C15D          goto    noadd7
                    add7

0159 6018          movfp   AARG+B0,WREG
015A 0F1E          addwf   DPX+B2    ;add lsb
015B 6019          movfp   AARG+B1,WREG
015C 111F          addwfc  DPX+B3    ;add msb
                    noadd7

015D 1A19          rrcf    AARG+B1,W  ; rotate sign into
015E 191F          rrcf    DPX+B3    ; for i < 8, no mea
015F 191E          rrcf    DPX+B2    ; are in DPX+B0
0160 191D          rrcf    DPX+B1
                    else
                    rrcf    AARG+B1,W  ; rotate sign into carry bit
                    rrcf    DPX+B3
                    rrcf    DPX+B2
                    rrcf    DPX+B1
                    rrcf    DPX+B0
                    fi
0008              i = i+1
                    if i < 8          ;test low byte
                    btfss   BARG+B0,i
                    else              ; test high byte
                    btfss   BARG+B1,i-8
                    fi

0161 901B          goto    noadd8
                    add8

0162 C167          movfp   AARG+B0,WREG
0163 6018          addwf   DPX+B2    ;add lsb
0164 0F1E          movfp   AARG+B1,WREG
0165 6019          addwfc  DPX+B3    ;add msb
0166 111F          noadd8

                    if i < 8
                    rrcf    AARG+B1,W  ; rotate sign into carry bit
                    rrcf    DPX+B3    ; for i < 8, no meaningful bits
                    rrcf    DPX+B2    ; are in DPX+B0
                    rrcf    DPX+B1
                    else
                    rrcf    AARG+B1,W  ; rotate sign into
                    rrcf    DPX+B3
                    rrcf    DPX+B2
                    rrcf    DPX+B1
                    rrcf    DPX+B0
                    fi
0167 1A19          i = i+1
0168 191F          if i < 8          ;test low byte
0169 191E          btfss   BARG+B0,i
016A 191D          else              ; test high byte
016B 191C          btfss   BARG+B1,i-8
                    fi

0009              goto    noadd9
                    add9

016C 911B          movfp   AARG+B0,WREG
016D C172          add7
016E 6018          movfp   AARG+B0,WREG
```

```

016F 0F1E          addwfc DPX+B2      ;add lsb
0170 6019          movfpc AARG+B1,WREG
0171 111F          addwfc DPX+B3      ;add msb
                    noadd9
                    if i < 8
                        rlcfc AARG+B1,W ; rotate sign into carry bit
                        rrcfc DPX+B3    ; for i < 8, no meaningful bits
                        rrcfc DPX+B2    ; are in DPX+B0
                        rrcfc DPX+B1
                    else
0172 1A19          rlcfc AARG+B1,W ; rotate sign into
0173 191F          rrcfc DPX+B3
0174 191E          rrcfc DPX+B2
0175 191D          rrcfc DPX+B1
0176 191C          rrcfc DPX+B0
                        fi
000A              i = i+1
                    if i < 8          ;test low byte
                        btfcfc BARG+B0,i
                    else
                        ; test high byte
0177 921B          btfcfc BARG+B1,i-8
                    fi
0178 C17D          goto    noadd10
                    add10
0179 6018          movfpc AARG+B0,WREG
017A 0F1E          addwfc DPX+B2      ;add lsb
017B 6019          movfpc AARG+B1,WREG
017C 111F          addwfc DPX+B3      ;add msb
                    noadd10
                    if i < 8
                        rlcfc AARG+B1,W ; rotate sign into carry bit
                        rrcfc DPX+B3    ; for i < 8, no meaningful bits
                        rrcfc DPX+B2    ; are in DPX+B0
                        rrcfc DPX+B1
                    else
017D 1A19          rlcfc AARG+B1,W ; rotate sign into
017E 191F          rrcfc DPX+B3
017F 191E          rrcfc DPX+B2
0180 191D          rrcfc DPX+B1
0181 191C          rrcfc DPX+B0
                        fi
000B              i = i+1
                    if i < 8          ;test low byte
                        btfcfc BARG+B0,i
                    else
                        ; test high byte
0182 931B          btfcfc BARG+B1,i-8
                    fi
0183 C188          goto    noadd11
                    add11
0184 6018          movfpc AARG+B0,WREG
0185 0F1E          addwfc DPX+B2      ;add lsb
0186 6019          movfpc AARG+B1,WREG
0187 111F          addwfc DPX+B3      ;add msb
                    noadd11
                    if i < 8
                        rlcfc AARG+B1,W ; rotate sign into carry bit
                        rrcfc DPX+B3    ; for i < 8, no meaningful bits
                        rrcfc DPX+B2    ; are in DPX+B0
                        rrcfc DPX+B1
                    else
0188 1A19          rlcfc AARG+B1,W ; rotate sign into
0189 191F          rrcfc DPX+B3
018A 191E          rrcfc DPX+B2
018B 191D          rrcfc DPX+B1
018C 191C          rrcfc DPX+B0
                        fi

```

Tone Generation

```
000C          i = i+1
              if i < 8
                ;test low byte
                btfss  BARG+B0,i
              else
                ; test high byte
018D 941B      btfss  BARG+B1,i-8
              fi
018E C193      goto   noadd12
              add12
018F 6018      movfpl AARG+B0,WREG
0190 0F1E      addwpl DPX+B2 ;add lsb
0191 6019      movfpl AARG+B1,WREG
0192 111F      addwplc DPX+B3 ;add msb
              noadd12
              if i < 8
                rlcfl  AARG+B1,W ; rotate sign into carry bit
                rrcfl  DPX+B3 ; for i < 8, no meaningful bits
                rrcfl  DPX+B2 ; are in DPX+B0
                rrcfl  DPX+B1
              else
0193 1A19      rlcfl  AARG+B1,W ; rotate sign into
0194 191F      rrcfl  DPX+B3
0195 191E      rrcfl  DPX+B2
0196 191D      rrcfl  DPX+B1
0197 191C      rrcfl  DPX+B0
              fi
000D          i = i+1
              if i < 8 ;test low byte
                btfss  BARG+B0,i
              else
                ;
                test high byte
0198 951B      btfss  BARG+B1,i-8
              fi
0199 C19E      goto   noadd13
              add13
019A 6018      movfpl AARG+B0,WREG
019B 0F1E      addwpl DPX+B2 ;add lsb
019C 6019      movfpl AARG+B1,WREG
019D 111F      addwplc DPX+B3 ;add msb
              noadd13
              if i < 8
                rlcfl  AARG+B1,W ; rotate sign into carry bit
                rrcfl  DPX+B3 ; for i < 8, no meaningful bits
                rrcfl  DPX+B2 ; are in DPX+B0
                rrcfl  DPX+B1
              else
019E 1A19      rlcfl  AARG+B1,W ; rotate sign into
019F 191F      rrcfl  DPX+B3
01A0 191E      rrcfl  DPX+B2
01A1 191D      rrcfl  DPX+B1
01A2 191C      rrcfl  DPX+B0
              fi
000E          i = i+1
              if i < 8 ;test low byte
                btfss  BARG+B0,i
              else
                ; test high byte
01A3 961B      btfss  BARG+B1,i-8
              fi
01A4 C1A9      goto   noadd14
              add14
01A5 6018      movfpl AARG+B0,WREG
01A6 0F1E      addwpl DPX+B2 ;add lsb
01A7 6019      movfpl AARG+B1,WREG
01A8 111F      addwplc DPX+B3 ;add msb
              noadd14
              if i < 8
                rlcfl  AARG+B1,W ; rotate sign into carry bit
                rrcfl  DPX+B3 ; for i < 8, no meaningful bits
                rrcfl  DPX+B2 ; are in DPX+B0
```

```

                                rrcf    DPX+B1
                                else
01A9 1A19                        rlcf    AARG+B1,W ; rotate sign into
01AA 191F                        rrcf    DPX+B3
01AB 191E                        rrcf    DPX+B2
01AC 191D                        rrcf    DPX+B1
01AD 191C                        rrcf    DPX+B0
                                fi
000F                              i = i+1
                                if    SIGNED
01AE 1A19                        rlcf    AARG+B1,W ; since BARG is alw
01AF 191F                        rrcf    DPX+B3 ; the last bit is k
01B0 191E                        rrcf    DPX+B2
01B1 191D                        rrcf    DPX+B1
01B2 191C                        rrcf    DPX+B0
                                endif
                                return
                                ;
                                ;*****
                                END

Errors   :    0
Warnings :    0
```

Tone Generation

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

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

Atlanta

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

Boston

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

Chicago

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

Dallas

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

Dayton

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

Los Angeles

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

New York

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

AMERICAS (continued)

San Jose

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

ASIA/PACIFIC

Hong Kong

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

Korea

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

Singapore

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

Taiwan

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

EUROPE

United Kingdom

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

France

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

Germany

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

Italy

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

JAPAN

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

9/22/95

All rights reserved. © 1995, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. 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. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.