

```

00001 ;SXNCOEX.asm      8/9/99          Richard Ottosen
00002 ;
00003 ;Scenix Numerically Controlled Oscillator Example
00004 ;
00005 ;
00006 ;Numerically controlled oscillator using direct digital synthesis.
00007 ;Based on an article by Tom Napier in Circuit Cellar INK #99, October
1998.

00008 ;
00009
00010 ;This code is assembled by the Microchip Inc. cross-assembler MPASM.
00011
00012          RADIX          DEC
00013          ERRORLEVEL    -305
00014          INCLUDE      SXDefs.inc
000140         LIST
000141         PROCESSOR     16C57   ; "SX28AC"
000333        LIST
1010 06FA    000334        DATA    _FUSE          ;configuration bits (TURBO, SYNC,
OPTIONX, etc.)
1011 040E    000335        DATA    _FUSEX       ; (PINS, CARRYX, BOR40, BANKS, PAGES)
000339        LIST
00015
00016          EXPAND          ;Must expand for SXSim to work
00017 ;***      NOEXPAND      ;Should not expand if small printed listing
is desired

00018          LIST          ST=OFF, MM=OFF   ;Keep listing small
00019
02FAF080    00020 MIPS      EQU      50000000      ;The SX instruction rate
00021
0047E903    00022 DEVICE   EQU      PINS28+PAGES4+BANKS8+OSCHS+BOR40+TURBO+STACKX+CARRYX
00023
0000000A    00024 Ver      EQU      10          ;Version # for outputing
00025 ID      'V','e','r',' ',' ',Ver/10,'.',Ver-(Ver/10*10),' ' ;Version # in
ID location

00026
00027
00028 ;
00029 ;Where things are
00030 ;
00000008    00031 RAMBase EQU      08h          ;Start of RAM
00000020    00032 RAMTop  EQU      020h        ;Last RAM location +1
00000800    00033 MemSize EQU      2048         ;Program memory size in words
00034
00035
00036
00037 ;For instruction destination argument
00038 ;W      EQU      0
00039 ;F      EQU      1
00040
00000000    00041 Indir   EQU      00h          ;Used for indirects thru FSR
00000001    00042 RTCC    EQU      01h          ;Real Time Clock/Counter
00000002    00043 PC      EQU      02h          ;Program Counter Low
00000004    00044 FSR     EQU      04h          ;File Select Register (index
register)
00000005    00045 PortA   EQU      05h          ;I/O Port A
00000006    00046 PortB   EQU      06h          ;I/O Port B
00000007    00047 PortC   EQU      07h          ;I/O Port C
00048
00000003    00049 Status  EQU      03h          ;Status register:
00050 #DEFINE CF      Status,0      ; Bit 0 = Carry Flag
00051 #DEFINE ZF      Status,2      ; Bit 2 = Zero Flag
00052

```

```

00053
00054 ;----- Constants -----
-----
00002580 00055
00056 Speed EQU 9600 ;Set speed: 1200 to 19200 Baud
00057
00058
00059 ;-----
-----
00060 ;
00061 ;Start of RAM
00062 ;
0008 00063 ORG RAMBase
00064
0008 00065 Zero RES 1 ;Holds the value 0 for two's-
complementing W
00066
0009 00067 Count RES 1 ;General purpose counter
000A 00068 DlyCnt RES 1 ;General purpose delay counter
000B 00069 DelMCnt RES 1 ;Cycle counter for fast delays
000C 00070 BitCnt RES 1 ;Bit counter for serial output
000D 00071 SerReg RES 1 ;Data register for serial output
00072
000E 00073 Freq RES 3 ;Holds the increment value for the
desired
0011 00074 Phase RES 3 ;Phase of the generated waveform
00075
0014 00076 Digit RES 1 ;Holds digit for decimal input
0015 00077 RegA RES 3 ;24 bit accumulator
00078
0018 00079 Temp RES 3 ;Very temporary storage
00080
00081 ;-----
-----
00082
07FF 00083 ORG MemSize-1 ;Reset vector
Message[306]: Crossing page boundary -- ensure page bits are set.
07FF 0A01 00084 GOTO Reset ;Start of program
00085
00086
0000 00087 ORG 0000h ;Start of ROM
00088 ;
00089 ;Interrupt handler
00090 ;
0000 00091 IntHan
00092 RETI ;End of "IntHan"
0000 000E M DATA 0Eh ;RETI ;return from interrupt (pull W,
STATUS, FSR, PC)
00093
00094
0001 0010 M FGOTO Start
0002 0AAB M DATA 10h|((Start)>>d'9') ;PAGE--write bits A12:A10 into
PA2:PA0
00095 M GOTO (Start)&1FFh
00096
00097
00098 ;
00099 ;Initialize the hardware of the SX
00100 ;
0003 0004 00101 InitHW CLRWDT ;Reset watchdog timer and prescale
0004 0064 00102 CLRF FSR
0005 0CFF 00103 MOVLW 11111111b
0006 0002 00104 OPTION ; bit 0,1,2 = Prescaler divider.
/256 00105 ; WDT: /1 to /128, Timer0: /2 to
00106 ; bit 3 = Prescaler assign: 1 = WDT

```

```

00107                                     ; bit 4 = RTCC edge: 1 = high to low
00108                                     ; bit 5 = RTCC source: 1 = external
00109                                     ; bit 6 = RTCC interrupt: 1 =
disabled
00110                                     ; bit 7 = Remap W-reg: 1 = Read RTCC
00111
00112         MODE      0Fh                ;Point the mode register to TRIS
registers
0007 005F          M          DATA  50h|(0Fh) ;MODE      ;write N into MODE register (N = 0-
F)
00113
00114 ;PortA  EQU      5                ;I/O PortA
00115 #DEFINE nupa0  PortA,0          ; Bit 0 = Output:
00116 #DEFINE nupa1  PortA,1          ; Bit 1 = Output:
00117 #DEFINE nupa2  PortA,2          ; Bit 2 = Output:
00118 #DEFINE RxD    PortA,3          ; Bit 3 = Input:
00119
00000008         00120 PADir   EQU      00001000b
0008 0C08         00121         MOVLW   PADir                ;Initialize Port A directions:
0009 0005         00122         TRIS    PortA
00123
00000000         00124 InitPA  EQU      00000000b          ;Initialize Port A outputs:
000A 0C00         00125         MOVLW   InitPA
000B 0025         00126         MOVWF   PortA
00127
00128 ;PortB  EQU      6                ;I/O PortB
00129 #DEFINE nupb0  PortB,0          ; Bit 0 = Output:
00130 #DEFINE nupb1  PortB,1          ; Bit 1 = Output:
00131 #DEFINE nupb2  PortB,2          ; Bit 2 = Output:
00132 #DEFINE nupb3  PortB,3          ; Bit 3 = Output:
00133 #DEFINE nupb4  PortB,4          ; Bit 4 = Output:
00134 #DEFINE nupb5  PortB,5          ; Bit 5 = Output:
00135 #DEFINE nupb6  PortB,6          ; Bit 6 = Output:
00136 #DEFINE nupb7  PortB,7          ; Bit 7 = Output:
00137
00000000         00138 PBDir   EQU      00000000b
000C 0C00         00139         MOVLW   PBDir                ;Initialize Port B directions:
000D 0006         00140         TRIS    PortB
00141
00000000         00142 InitPB  EQU      00000000b          ;Initialize Port B outputs:
000E 0C00         00143         MOVLW   InitPB
000F 0026         00144         MOVWF   PortB
00145
00000000         00146 PCDir   EQU      00000000b
0010 0C00         00147         MOVLW   PCDir                ;Initialize Port C directions:
0011 0007         00148         TRIS    PortC
00149
00000000         00150 InitPC  EQU      00000000b          ;Initialize Port C outputs:
0012 0C00         00151         MOVLW   InitPC
0013 0027         00152         MOVWF   PortC
00153
00154         RETP                    ;End of "InitHW"
0014 000D         M          DATA  0Dh          ;RETP      ;RET & write return addr bits 10:9
into PA1:PA0
00155
00156
00157 ;
00158 ;Receive a character as 8 bits, no parity, 1 stop and put in
"SerReg".
00159 ; This routine has the start, stop and data bits inverted because an
inverting
00160 ; RS-232 buffer is not used.
00161 ;
0015 0C08         00162 Rcv     MOVLW   8                ;Put # of data bits
0016 002C         00163         MOVWF   BitCnt          ; into counter
0017 0765         00164 _Rc05  BTFSS   RxD                ;Wait for the start bit

```

```

0018 0A17          00165          GOTO    _Rc05
                   00166
0019 0C1A          00167 _RDly1  MOVLW  249600/Speed    ;Delay 1/2 bit time
001A 002A          00168          MOVWF  DlyCnt
001B 0403          00169          BCF    CF              ; by dividing "Speed" by 2
001C 032A          00170          RRF    DlyCnt
001D 0C31          00171 _RD1     MOVLW  ((MIPS/250000)-4)/4      ;
001E 002B          00172          MOVWF  DelMCnt
001F 02EB          00173 _RD1a   DECFSZ  DelMCnt
0020 0A1F          00174          GOTO   _RD1a
0021 02EA          00175          DECFSZ  DlyCnt          ;
0022 0A1D          00176          GOTO   _RD1          ;
Cycles
                   00177
0023 0765          00178          BTFSS  RxD            ;Is start still active?
0024 0A15          00179          GOTO   Rcv            ; try again if not
                   00180
0025 0C1A          00181 _Rc10   MOVLW  249600/Speed    ;Delay 1 bit time
0026 002A          00182          MOVWF  DlyCnt
0027 0C31          00183 _RD2     MOVLW  ((MIPS/250000)-4)/4      ;Delay in multiples of 4us
0028 002B          00184          MOVWF  DelMCnt        ; allowing for loop overhead
0029 02EB          00185 _RD2a   DECFSZ  DelMCnt
002A 0A29          00186          GOTO   _RD2a
002B 02EA          00187          DECFSZ  DlyCnt          ;
002C 0A27          00188          GOTO   _RD2          ;
Cycles
                   00189
002D 0765          00190          BTFSS  RxD            ;If data is a zero
002E 0503          00191          BSF    CF              ; then set the carry
002F 0665          00192          BTFSC  RxD            ;If data is a one
0030 0403          00193          BCF    CF              ; then clear the carry
0031 032D          00194          RRF    SerReg         ;Get carry into data bit
                   00195
0032 02EC          00196          DECFSZ  BitCnt        ;Count the data bits
0033 0A25          00197          GOTO   _Rc10         ; and exit when done
                   00198
                   00199
0034 0C1A          00200 _RDly3  MOVLW  249600/Speed    ;Delay 1/2 bit time
0035 002A          00201          MOVWF  DlyCnt
0036 0403          00202          BCF    CF              ; by dividing "Speed" by 2
0037 032A          00203          RRF    DlyCnt
0038 0C31          00204 _RD3     MOVLW  ((MIPS/250000)-4)/4      ;
0039 002B          00205          MOVWF  DelMCnt
003A 02EB          00206 _RD3a   DECFSZ  DelMCnt
003B 0A3A          00207          GOTO   _RD3a
003C 02EA          00208          DECFSZ  DlyCnt          ;
003D 0A38          00209          GOTO   _RD3          ;
Cycles
                   00210
003E 020D          00210          MOVF   SerReg,W
00211          RETP                ;End of "Rcv"
003F 000D          M              DATA  0Dh      ;RETP ;RET & write return addr bits 10:9
into PA1:PA0
                   00212
                   00213
00214 ;
00215 ;Subroutine to input a 24-bit integer in decimal ASCII FORMAT.
00216 ; The result is in RegA.
00217 ;
0040 0075          00218 DecIn   CLRF   RegA          ;Initialize
0041 0076          00219          CLRF   RegA+1
0042 0077          00220          CLRF   RegA+2
                   00221
0043 0915          00222 _DI10   CALL   Rcv            ;Read in an ASCII digit
0044 0034          00223          MOVWF  Digit
0045 0503          00224          BSF    CF
0046 0C30          00225          MOVLW  '0'           ;Convert it to binary

```

```

0047 00B4          00226      SUBWF    Digit
0048 0703          00227      BTFSS   CF
0049 0A67          00228      GOTO    _NotDig          ;Branch if it wasn't a digit
004A 0503          00229      BSF     CF
004B 0C0A          00230      MOVLW  10
004C 0094          00231      SUBWF   Digit,W
004D 0603          00232      BTFSC  CF
004E 0A67          00233      GOTO    _NotDig
          00234
          00235 ;RegA:= RegA *10.
004F 0215          00236      MOVF   RegA,W          ;Save starting value
0050 0038          00237      MOVWF  Temp
0051 0216          00238      MOVF   RegA+1,W
0052 0039          00239      MOVWF  Temp+1
0053 0217          00240      MOVF   RegA+2,W
0054 003A          00241      MOVWF  Temp+2
          00242
0055 0C09          00243      MOVLW  9              ;10 times
0056 0029          00244      MOVWF  Count
0057 0403          00245      _DI20  BCF     CF          ;RegA:= RegA +Temp
0058 0218          00246      MOVF   Temp,W
0059 01F5          00247      ADDWF  RegA
005A 0219          00248      MOVF   Temp+1,W
005B 01F6          00249      ADDWF  RegA+1
005C 021A          00250      MOVF   Temp+2,W
005D 01F7          00251      ADDWF  RegA+2
005E 02E9          00252      DECFSZ Count
005F 0A57          00253      GOTO   _DI20
          00254
0060 0403          00255      BCF     CF          ;RegA:= RegA +Digit
0061 0214          00256      MOVF   Digit,W
0062 01F5          00257      ADDWF  RegA
0063 0040          00258      CLRW
0064 01F6          00259      ADDWF  RegA+1
0065 01F7          00260      ADDWF  RegA+2
0066 0A43          00261      GOTO   _DI10          ;Loop until not a digit
          00262
          00263 _NotDig RETP          ;End of "DecIn"
0067 000D          M          DATA  0Dh          ;RETP ;RET & write return addr bits 10:9
into PA1:PA0
          00264
          00265
0068 0403          00266      SineTbl BCF     CF
0069 01E2          00267      ADDWF  PC
006A 0880 0883 0886 00268      DT     080h,083h,086h,089h,08Ch,090h,093h,096h
          0889 088C 0890
          0893 0896
0072 0899 089C 089F 00269      DT     099h,09Ch,09Fh,0A2h,0A5h,0A8h,0ABh,0AEh
          08A2 08A5 08A8
          08AB 08AE
007A 08B1 08B3 08B6 00270      DT     0B1h,0B3h,0B6h,0B9h,0BCh,0BFh,0C1h,0C4h
          08B9 08BC 08BF
          08C1 08C4
0082 08C7 08C9 08CC 00271      DT     0C7h,0C9h,0CCh,0CEh,0D1h,0D3h,0D5h,0D8h
          08CE 08D1 08D3
          08D5 08D8
008A 08DA 08DC 08DE 00272      DT     0DAh,0DCh,0DEh,0E0h,0E2h,0E4h,0E6h,0E8h
          08E0 08E2 08E4
          08E6 08E8
0092 08EA 08EB 08ED 00273      DT     0EAh,0EBh,0EDh,0EFh,0F0h,0F1h,0F3h,0F4h
          08EF 08F0 08F1
          08F3 08F4
009A 08F5 08F6 08F8 00274      DT     0F5h,0F6h,0F8h,0F9h,0FAh,0FAh,0FBh,0FCh
          08F9 08FA 08FA
          08FB 08FC
00A2 08FD 08FD 08FE 00275      DT     0FDh,0FDh,0FEh,0FEh,0FEh,0FFh,0FFh,0FFh

```

```

08FE 08FE 08FF
08FF 08FF
00AA 08FF          00276          DT          0FFh
                   00277
                   00278
                   00279 ;
                   00280 ;===== Start of Program
=====
                   00281 ;
                   00282 ;
                   00283
00AB 0010          00284 Start  FCALL  InitHW
PA2:PA0           M          DATA  10h|((InitHW)>>d'9') ;PAGE--write bits A12:A10 into
00AC 0903          M          CALL   (InitHW)&1FFh
00AD 0068          00285          CLRF   Zero          ; initialize the value of zero so we
can
                   00286          ; calculate the two's complement of W
using
                   00287          ; "SUBWF ZERO,W"
                   00288
                   00289
00290 ;          OscFreq/LoopLength          AccumulatorLength
00291 ;          ----- = -----
00292 ;          FrequencyOut          Increment
00293
00294 ;
00295 ;FreqOut = (MIPS /LoopCycles *N) / 2^^24
00296 ;N = FreqOut * 2^^24 /OscFreq *LoopCycles
00297 ;
00002710          00298 N          EQU    16777216 *30 /MIPS *1000 ;This gives about 10,000
for 1KHZ:
                   00299          ; 10,066,3296
00AE 0C10          00300          MOVLW  LOW(N)          ;Initialize to 1 KHz
00AF 002E          00301          MOVWF  Freq
00B0 0C27          00302          MOVLW  HIGH(N)
00B1 002F          00303          MOVWF  Freq+1
00B2 0C00          00304          MOVLW  00h
00B3 0030          00305          MOVWF  Freq+2
                   00306
                   00307
00308 ;
00309 ;===== MAIN LOOP =====
00310 ;
00311
00B4          00312 Main
                   00313
00B4 0403          00314 Loop   BCF    CF          ;
1 Cycle
00B5 020E          00315          MOVF   Freq,W          ;Get the current frequency
1
00B6 01F1          00316          ADDWF  Phase          ; and increment the phase
1
00B7 020F          00317          MOVF   Freq+1,W          ;
1
00B8 01F2          00318          ADDWF  Phase+1          ;
1
00B9 0210          00319          MOVF   Freq+2,W          ;
1
00BA 01F3          00320          ADDWF  Phase+2          ;
1
                   00321
00BB 0213          00322          MOVF   Phase+2,W          ;
1
00BC 0403          00323          BCF    CF          ;
1

```

```

00BD 06D3      00324      BTFSC      Phase+2,6      ;Is it the 2nd half of a half cycle?
1
00BE 0088      00325      SUBWF      Zero,W      ; then complement to effect a
1
00BF 0E7F      00326      ; subtract from 64
1      00327      ANDLW      07Fh      ;Reduce to 6 bits for 90 degrees
00C0 0968      00328      CALL      SineTbl      ; and get amplitude
10
00C1 0503      00329      ;
1      00330      BSF      CF
00C2 07F3      00331      BTFSS      Phase+2,7      ;Is amplitude supposed to be
negative? 1
00C3 0088      00332      SUBWF      Zero,W      ;Change polarity of sine amplitude
1
00C4 0026      00333      ;
1      00334      MOVWF     PortB      ;Output to DAC
00C5 0765      00335      BTFSS      RxD      ;Test for a start bit
1
00C6 0AB4      00336      GOTO      Loop      ;No start bit, keep looping
3
=30 Cycles      00337      ;
00338      ; else get a new frequency
00339
00C7 0940      00340 New      CALL      DecIn      ;Read in an integer frequency
00C8 0215      00341      MOVF      RegA,W      ;Update the frequency setting
00C9 002E      00342      MOVWF     Freq
00CA 0216      00343      MOVF      RegA+1,W
00CB 002F      00344      MOVWF     Freq+1
00CC 0217      00345      MOVF      RegA+2,W
00CD 0030      00346      MOVWF     Freq+2
00347
00CE 0AB4      00348      GOTO      Main      ;Restart at a new frequency
00349
00350      END

```

```

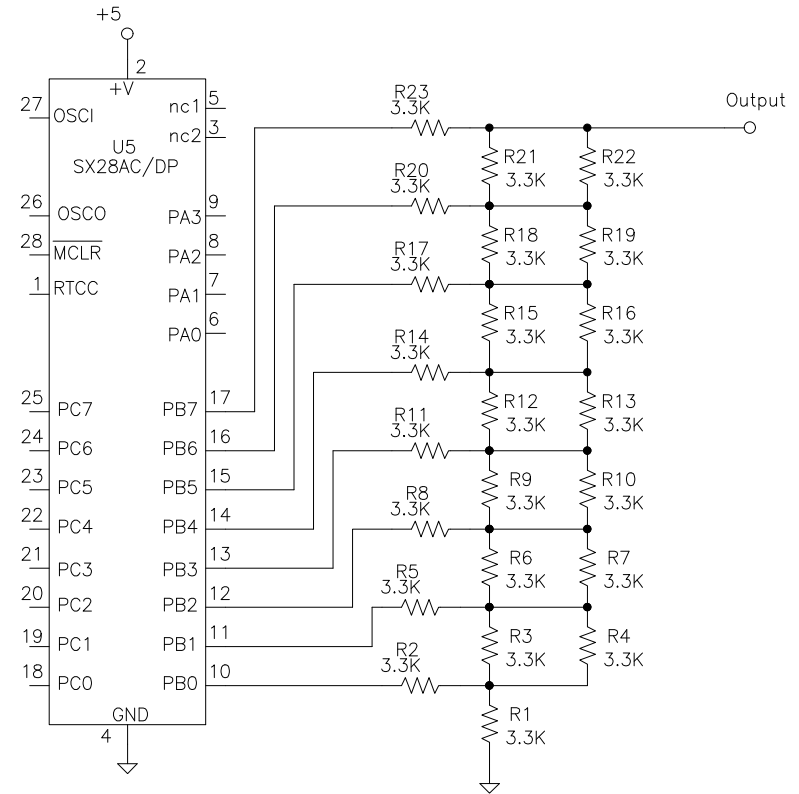
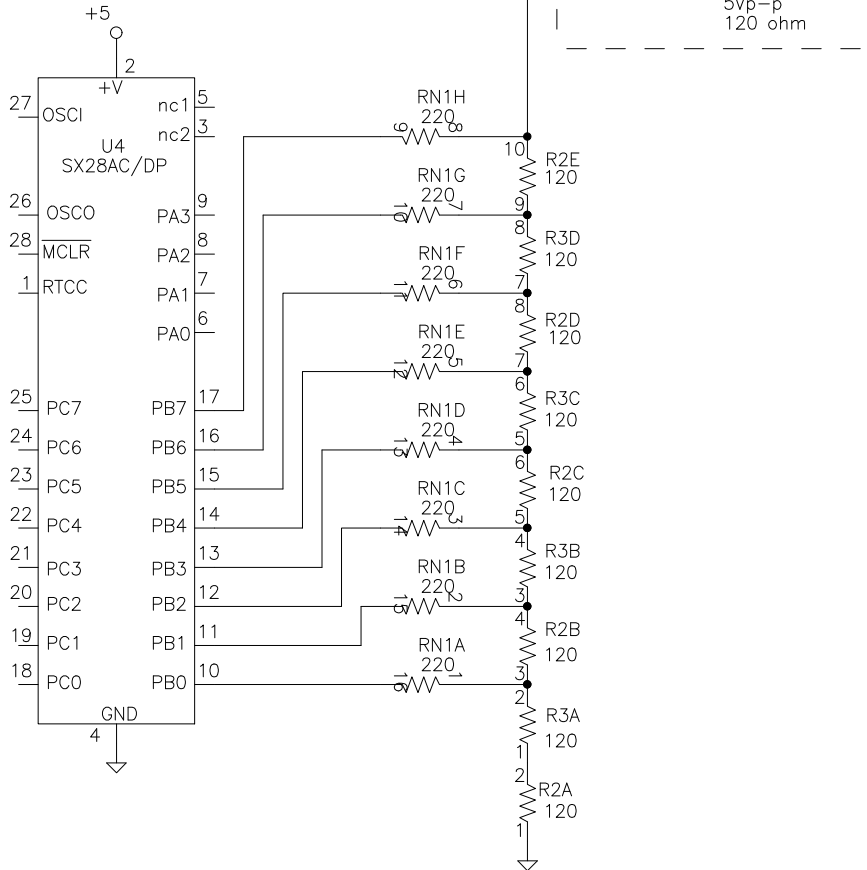
Errors      :      0
Warnings    :      0 reported,      19 suppressed
Messages    :      1 reported,      22 suppressed

```

Use a DIP resistor network for RN1 and SIP resistor networks for RN2 and RN3.

Mount the SIP networks right next to the pin 1 side of the DIP network.

Stagger the resistor network packages to place connected pins right next to each other. No wires should be needed to solder the network pins together.



All resistors are size 0805 surface mount.

The resistors should have 1% tolerance and come from one section of tape to get some matching.

Use a prototyping board with pads on 1/10 inch grid.

The parts are positioned just like the schematic.

To parallel the resistors, solder one on top of the other.

The resistors share pads so no wires are needed to connect them together.

Rho Enterprises Box 33 4100 W. Colfax Ave. Denver, CO 80204 www.rhoent.com (720) 359-1467		R2R on SX-28 Proto Port B Pins	
Sheet 1	of 1	Rev 1	Size A
Drawn by Richard Ottosen		Date 8/7/99	
Drawing No.			
Copyright 1999 Rho Enterprises			