AN540

Implementing IIR Digital Filters

Author: *Amar Palacherla*
Microchip Technology Inc.

INTRODUCTION

This application note describes the implementation of various digital filters using the PIC17C42, the first member of Microchip's 2nd generation of 8-bit microcontrollers. The PIC17C42 is a very high speed 8-bit microcontroller with an instruction cycle time of 250 ns (@ 16 MHz input clock). Even though the PIC17C42 is an 8-bit device, it's high speed and efficient instruction set allows implementation of digital filters for practical applications. Traditionally digital filters have been implemented using expensive Digital Signal Processors (DSPs). In a system the DSP is normally a slave processor being controlled by either an 8-bit or 16-bit microcontroller. Where sampling rates are not high (especially in mechanical control systems), a single chip solution is possible using the PIC17C42.

This application note provides a few examples of implementing digital filters. Example code for 2nd order Infinite Impulse Response (IIR) filters is given. The following type of filters are implemented:

- Low Pass
- High Pass
- Band Pass
- Band Stop (notch) filter

This application note does not explain how to design a filter. Filter design theory is well established and is beyond the scope of this application note. It is assumed that a filter is designed according to the desired specifications. The desired digital filters may be designed using either standard techniques or using commonly available digital filter design software packages.

Finite Impulse Response (FIR) filters have many advantages over IIR filters, but are much more resource intensive (both in terms of execution time and RAM). On the other hand, IIR filters are quite attractive for implementing with the PIC17C42 resources. Especially where phase information is not so important, IIR filters are a good choice (FIR filters have a linear phase response). Of the various forms used for realizing digital filters (like, Direct form, Direct II form, Cascade form, Parallel, Lattice structure, etc.) the Direct II form is used in this application note. It is easy to understand and simple macros can be built using these structures.

THEORY OF OPERATION

Digital filters in most cases assume the following form of relationship between the output and input sequences.

$$y(n) = \sum_{i=o}^{M} a_i y(n-i) + \sum_{j=o}^{N} b_j x(n-j)$$

The above equation basically states that the present output is a weighted sum of the past inputs and past outputs. In case of FIR filters, the weighted constants $ai = 0$ and in case of IIR filters, at least one of the ai constants is non zero. In case of IIR, the above formula may be rewritten in terms of Z transform as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=o}^{M} b_k Z^{-k}}{1 + \sum_{k=1}^{N} a_k Z^{-k}}$$

The above equation can further be rewritten in difference equation format as follows:

$$y(n) = \sum_{i=1}^{M} a_i y(n-i) + \sum_{j=o}^{N} b_j x(n-j)$$

Realization of the above equation is called the Direct Form II structure. For example, in case of a second order structure, $M = N = 2$, gives the following difference equations:

EQUATION 1:

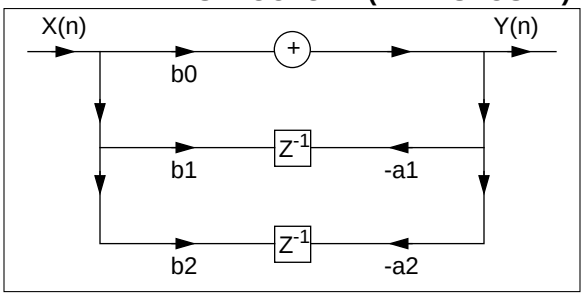
$$d(n) = x(n) + a_1 d(n-1) + a_2 d(n-2)$$

EQUATION 2:

$$y(n) = b_0 d(n) + b_1 d(n-1) + b_2 (d(n-2))$$

The above difference equations may be represented as shown in Figure 1.

FIGURE 1: 2ND ORDER DIRECT FORM II STRUCTURE (TRANSPOSED)



The structure as shown in Figure 1 may be cascaded to attain a higher order filter. For example, if two stages are cascaded together, a 4th Order IIR Filter is obtained. This way, the output of the 1st stage becomes the input to the second stage. Multiple order filters are thus implemented by cascading a 2nd order filter structure as shown in Figure 1.

IMPLEMENTATION

A 4th order IIR Filter is implemented by cascading two of the structures shown in Figure 1. The output Y (output of each filter stage) is computed by direct implementation of Equation 1 and Equation 2. Since each stage is similar algorithmically, it is implemented as a macro using Microchip's, Assembler/Linker for PIC17C42. This Macro (labelled `BIQUAD`) is called twice for implementing a 4th order filter. The output of the 1st stage is directly fed to the input of the second stage without any scaling.

Scaling may be required depending on the particular application. The user can modify the code very easily without any penalty on speed. Also, saturation arithmetic is not used. Overflows can be avoided by limiting the input sequence amplitude. All numbers are assumed to be 16 bits in Q15 format (15 decimal points, MSb is sign bit). Thus the user must scale and sign extend the input sequence accordingly. For example, if the input is from a 12-bit A/D converter, the user must sign extend the 12-bit input if bit 11 is a one.

The `BIQUAD` macro is a generic macro and can be used for all IIR filters whether it is Low Pass, High Pass, Band Pass or Band Stop. A general purpose 16x16 multiplier routine is also provided. This routine is implemented as a straight line code for speed considerations.

The 4th order IIR filter implemented is a Low Pass Filter with the specifications shown in Table 1.

TABLE 1: FILTER CONSTANTS

	BAND1	BAND2
Lower Band Edge	0.0	600 Hz
Upper Band Edge	500 Hz	1 kHz
Nominal Gain	1.0	0.0
Nominal Ripple	0.01	0.05
Maximum Ripple	0.00906	0.04601
Ripple in dB	0.07830	-26.75
Sampling Frequency = 2 kHz		

The Low Pass Filter is designed using a digital filter design package (DFDP™ by Atlanta Signal Processors Inc.). The filter package produces filter constants of the structure shown in Table 1. Table 2 shows the filter co-efficients that are obtained for the above Low Pass filter specification.

TABLE 2: FILTER CO-EFFICIENTS

Stage	Co-efficients				
	a1	a2	b0	b1	b2
1	-0.133331	0.167145	0.285431	0.462921	0.285431
2	0.147827	0.765900	0.698273	0.499908	0.698273

The above filter co-efficients (5 per stage) are quantized to Q15 format (i.e they are multiplied by 32768) and saved in program memory (starting at label `_coeff_lpass`). The constants for both the stages are read into data memory using `TLRD` and `TABLRD` instructions in the Initialization Routine (labelled `initFilter`). The user may read the coefficients of only one stage at a time and save some RAM at the expense of speed.

The sample 4th order Low Pass IIR Filter is tested by analyzing the impulse response of the filter. An impulse signal is fed as input to the filter. This is simulated by forcing the input to the filter by a large quantity (say 7F00h) on the first input sample, and the all zeros from the 2nd sample onwards. The output sequence is the filter's impulse response and is captured into the PICMASTER's (Microchip's Universal In-Circuit Emulator) real-time trace buffer. This captured data from PICMASTER is saved to file and analyzed. Analysis was done using MathCad™ for Windows® and a DSP Analysis program from Burr-Brown (DSPLAY™). The Fourier Transform of this impulse response of the filter should display the filter's frequency response, in this case being a Low Pass type. The plots of the impulse response and the frequency response are shown in Figure 2, Figure 3 and Figure 4.

MathCad is a trademark of MathSoft, Inc.
DSPLAY is a trademark of Burr-Brown
DFDP is a trademark of Atlanta Signal Processing Inc.
Windows is a registered trademark of Microsoft Corporation.

FIGURE 2: IMPULSE RESPONSE CAPTURED FROM PICMASTER

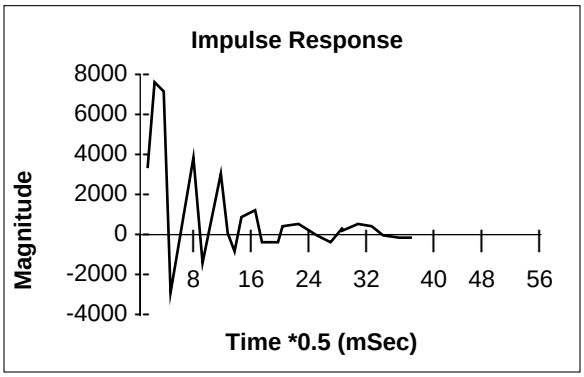
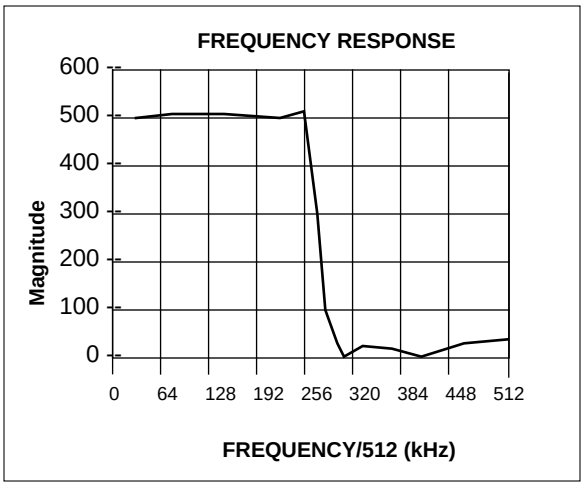


FIGURE 3: SPECTRUM COMPUTED FROM IMPULSE RESPONSE



PERFORMANCE

The resource requirements for filter implementations using a PIC17C42 is given in Table 3. These numbers can be used to determine whether a higher order filter can be executed in real-time. The same information may be used to determine the highest sampling rate possible.

FILTER APPLICATIONS

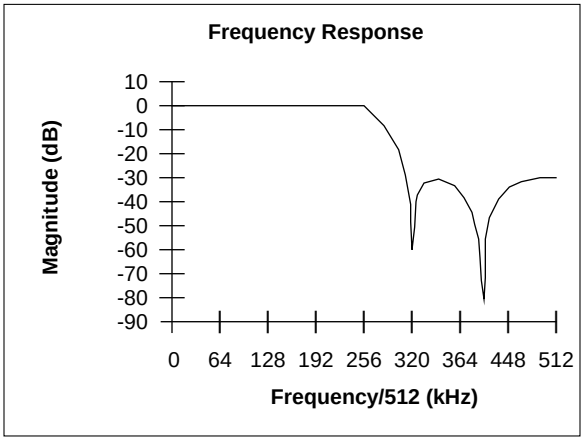
Digital filters find applications in many areas especially those involving processing of real world signals. In some applications like ABS systems in an automobile, digital filtering becomes a must. In this case elimination of noise (especially glitches and false readings of sensors) is very critical and thus becomes a requirement of digital signal processing.

TABLE 3: RESOURCE REQUIREMENTS

Timing (Cycles)(1)	#of Filter Stages*775 + 16
Program Memory (locations)(1)	#of Filter Stages*68 + 290
RAM (File Registers)	#of Filter Stages*16 + 16

Note 1: The above numbers do not include the initialization routine.

FIGURE 4: LOG MAGNITUDE SPECTRUM OF IMPULSE RESPONSE



Digital filters are also needed in Process Control where notch filters and low pass filters are desired because the signals from sensors are transmitted over long lines, especially in a very noisy environment. In these cases, typically a notch filter (centering 50 Hz or 60 Hz) is used. In cases of eliminating background noise, a band stop filter (e.g., 40 Hz to 120 Hz) is used. The sample code given in this application note can be used to design a feedback control system's digital compensator. For example, a typical DC Motor's digital compensator (like a dead-beat compensator) is of second order and has the same filter structure that is implemented in this application note.

TABLE 4: RESOURCE REQUIREMENTS

Filter Order	Cycles	Real Time (@16 MHz)	Maximum Sampling	Program Memory(1)	RAM
2	791	197.75 µs	5.05 kHz	358	32
4	1566	391.5 µs	2.55 kHz	426	48
6	2341	585.25 µs	1.7 kHz	494	64
8	3116	779.0 µs	1.28 kHz	562	80
10	3891	972.75 µs	1.0 kHz	630	94

Note 1: The above numbers do not include the initialization routine.

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: IIR.LS0T

```
MPASM 01.40 Released             IIR.ASM   1-16-1997  14:48:37         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

             00001         TITLE   “Digital IIR Filter Using PIC17C42”
             00002
             00003         LIST    P=17C42, columns=120,WRAP, R = DEC
             00004
             00005                 #include <p17c42.inc>
             00001         LIST
             00002 ;P17C42.INC Standard Header File, Version 1.03 Microchip Technology, Inc.
             00264         LIST
             00006
             00007         #define true    1
             00008         #define false   0
             00009         #define TRUE    1
             00010         #define FALSE   0
             00011
             00012         #define LSB     0
             00013         #define MSB     7
             00014
  00000001   00015 _INC     equ     1
  00000000   00016 _NO_INC  equ     0
  00000000   00017 _LOW     equ     0
  00000001   00018 _HIGH    equ     1
             00019
             00020                 #include <17c42.mac>
             00945         LIST
             00021
             00022                 #include <17c42iir.mac>
             00001         LIST
             00002 ;*****************************************************************
             00003 ;                       PIC17C42 MACRO
             00004 ;
             00005 ;Macro For A Bi-Quad IIR Filter
             00006 ;       2nd order Direct Form (Transposed) Type
             00007 ;
             00008 ; Filter co-efficients B0 & B2 are assumed equal
             00009 ;
             00010 ; The difference equations for each cascade section is given by :
             00011 ;         Y(n) = B0*D(n) + B1*D(n-1) + B2*D(n-2)
             00012 ;         D(n) = X(n) - A1*D(n-1) -  A2*D(n-2)
             00013 ;    where X(n) = input sample, Y(n) = output of filter
             00014 ;    and A1, A2, B0, B1, B2 are the Filter Co-efficients
             00015 ;
             00016 ;  The above difference equations are only for 1 section of a
             00017 ; 2nd order Direct_Form II Filter structure (IIR)
             00018 ;
             00019 ;   NOTE :
             00020 ;       It is possible to design the above structures
             00021 ;       such that the co-efficients B0 = B2. If this is the
             00022 ;       case,
             00023 ;       Y(n) = B0*[D(n) + D(n-2)] = B2*[D(n) + D(n-2)]
             00024 ;       This way, one multiplication can be avoided
             00025 ;
             00026 ;  If a 4th order filter is to be implemented, the output of
             00027 ; the 1st structure should be input to the 2nd cascade section
             00028 ;
```

```
            00029 ;       Timing (WORST CASE) :
            00030 ;              59+4*179 = 775 Cycles
            00031 ;              (194 uS @ 16 Mhz)
            00032 ;              Program Memory  :
            00033 ;              63 locations
            00034 ;
            00035 ;*****************************************************************
            00036 ;       The sample filters are designed so that B0=B2
            00037 ;       This saves 1 multiplication
            00038 ;
 00000001   00039 B0_EQUALS_B2    equ     TRUE
            00040 ;
            00041 ;*****************************************************************
            00042 ;  Parameters to BIQUAD Macro :
            00043 ;               Filter Constants A1, A2, B0, B1, B2
            00044 ;               & D(n), D(n-1), D(n-2), filter stage #
            00045 ;
            00046 BIQUAD  MACRO   Ax1,Ax2,Bx0,Bx1,Dn,Dn_1,Dn_2,stage
            00047
            00048 ;
            00049 ; Compute Ax2*D(n-2)
            00050 ;
            00051  MOVFP16  Dn_2,AARG      ; D(n-2) = multiplier
            00052  MOVFP16  Ax2,BARG       ; A2 = multiplicand
            00053         call    DblMult  ; (ACCd,ACCc) = A2*D(n-2)
            00054 ;
            00055 ; Add product to output of 1st section
            00056 ; Save result in 32 bit Accumulator
            00057 ;
            00058         ADD32    DPX,ACC
            00059 ;
            00060 ;  Compute A1*D(n-1)
            00061 ;
            00062  MOVFP16  Dn_1,AARG      ; AARG = D(n-2) = multiplier
            00063  MOVFP16  Ax1,BARG       ; BARG = A2 = multiplicand
            00064         call    DblMult         ; (ACCd,ACCc) = A1*D(n-1)
            00065 ;
            00066 ; Compute A1*D(n-1) + A2*D(n-2) + output of previous section
            00067 ;       multiplications already done, so simply perform a 32 bit add
            00068 ;       of previously obtained multiplication results
            00069 ;
            00070   ADD32   DPX,ACC ; ACC = A1*D(n-1) + A2*D(n-2) + (output of 1st section)
            00071 ;
            00072 ;
            00073 ; save the upper 16 bits  of D(n) from the 32 bit accumulator
            00074 ; left shift the result by 1, to adjust the decimal point after
            00075 ; a Q15*Q15 multiplication
            00076 ;
            00077         rlcf     ACC+B1,w
            00078         rlcf     ACC+B2,w
            00079         movwf     Dn
            00080         rlcf     ACC+B3,w        ; decimal adjust ( mult by 2)
            00081         movwf     Dn+B1
            00082 ;
            00083 ; Compute B2 * [D(n) + D(n-2)]
            00084 ;
            00085         #if B0_EQUALS_B2
            00086
            00087         ADD16ACC   Dn_2,Dn,AARG    ; AARG = Dn + D(n-2) = multiplier
            00088  MOVFP16    Bx0,BARG        ; BARG = A2 = multiplicand
            00089         call       DblMult         ; (ACCd,ACCc) = B2*[D(n)+D(n-2)]
            00090  MOVPF32    DPX,ACC
            00091
            00092         #else
            00093
            00094  MOVFP16    Bx0,BARG
```

```
00095  MOVFP16    Dn,AARG
00096          call       DblMult         ; B0*D(n)
00097  MOVPF32    DPX,ACC
00098
00099  MOVFP16    Bx2,BARG
00100  MOVFP16    Dn_2,AARG
00101          call       DblMult         ; B2*D(n-2)
00102          ADD32      DPX,ACC
00103
00104          #endif
00105 ;
00106 ; Shift down D(n-1) to D(n-2) after D(n-2) usage is no longer required.
00107 ; This way in the next iteration D(n-2) is equal to the present D(n-1)
00108 ;
00109          movfp     Dn_1,AARG+B0
00110          movpf     AARG+B0,Dn_2     ; Shift down D(n-1)
00111          movfp     Dn_1+B1,AARG+B1
00112          movpf     AARG+B1,Dn_2+B1  ; AARG = D(n-1) = multiplier
00113
00114  MOVFP16   Bx1,BARG         ; BARG = B1 = multiplicand
00115
00116          call      DblMult          ; (ACCd,ACCc) = B1*D(n-1)
00117 ;
00118 ; Compute Output Y = B1*D(n-1) + B2*D(n-2) + B0*D(n)
00119 ;                 = B1*D(n-1) + B0*[D(n) + D(n-2)]
00120 ; Since all multiplications are already done, simply perform a
00121 ; 32 bit addition
00122 ;
00123          ADD32     DPX,ACC ; ACC = B1*D(n-1) + B2*D(n-2) + B0*D(n)
00124 ;
00125 ; Shift down D(n) to D(n-1) so that in the next iteration, the new
00126 ; D(n-1) is the present D(n)
00127 ;
00128  MOV16     Dn,Dn_1 ; Shift down D(n) to D(n-1)
00129 ;
00130          ENDM
00131
00132          LIST
00023 ;
00024 ;*****************************************************************
00025 ;               Second Order Direct Form IIR Filter
00026 ;
00027 ;
00028 ; In the code given below, a 4th order IIR Elliptic Lowpass Filter
00029 ; is implemented. Other order filters may be implemented by
00030 ; taking the following example code as a basis.
00031 ;
00032 ;
00033 ;       Program:          IIR.ASM
00034 ;       Revision Date:
00035 ;                         1-13-97      Compatibility with MPASMWIN 1.40
00036 ;
00037 ;*****************************************************************
00038 ;
00039 ;       The specifications of the filter are :
00040 ;
00041 ;       Sampling Frequency = 2.0 Khz
00042 ;
00043 ;       Filter Type = 4th Order Elliptic Lowpass Filter
00044 ;
00045 ;                                Band1           Band2
00046 ;       Lower Band Edge  0.0              600 Hz
00047 ;       Upper Band Edge          500 Hz          1 Khz
00048 ;       Nominal Gain             1.0      0.0
00049 ;       Nominal Ripple   0.01     0.05
00050 ;Maximum Ripple   0.00906  0.04601
```

```
00051 ;       Ripple in dB     0.07830         -26.75
00052 ;
00053 ;  The Filter Co-efficients for the above specifications
00054 ; of the filter are computed as follows :
00055 ;
00056 ;      1st Section :
00057 ;                       A11 = -0.133331
00058 ;                       A12 = 0.167145
00059 ;                       B10 = 0.285431
00060 ;                       B11 = 0.462921
00061 ;                       B12 = 0.285431
00062 ;      2nd Section
00063 ;                       A21 = 0.147827
00064 ;                       A22 = 0.765900
00065 ;                       B20 = 0.698273
00066 ;                       B21 = 0.499908
00067 ;                       B22 = 0.698273
00068 ;
00069 ;
00070 ;       Performance (WORST Case):
00071 ;
00072 ;                       Cycles    = #of Filter Stages*775 + 16
00073 ;                                 = 2*775+16 = 1566 Cycles
00074 ;                                        ( 391 uSec)
00075 ;                                per each sample. Initialization
00076 ;                                time after reset is not counted
00077 ;                                Timing measured with B0_EQUALS_B2
00078 ;                                set to TRUE (see BIQUAD Macro for
00079 ;                                explanation)
00080 ;
00081 ;               Program Memory :
00082 ;                                = 16+ # of FilterStages * (BIQUAD Memory
00083 ;                                          + filter co-efficients)
00084 ;                                          + multiplier
00085 ;                                = 16+2*(63+5)+274 = 421 locations
00086 ;                                  (excluding initialization)
00087 ;
00088 ;                       RAM usage = 48 file registers
00089 ;                       RAM usage/each additional stage = 16 file regs
00090 ;
00091 ;
00092 ;                       This time is less than 2 Khz (500 uSec),
00093 ;                       which means real time filtering is possible
00094 ;
00095 ;*****************************************************************
00096
00097 ;*****************************************************************
00098 ;
00099 ;
00100         CBLOCK  0
00000000 00101                 BB0,BB1,BB2,BB3
00102         ENDC
00103 ;*****************************************************************
00104 ;
00105     CBLOCK  0x18
00000018 00106         DPX,DPX1,DPX2,DPX3              ; arithmetic accumulator
0000001C 00107         AARG,AARG1,BARG,BARG1           ; multiply arguments
00108     ENDC
00109 ;
00110     CBLOCK
00000020 00111             Dn1,   Dn1_Hi
00000022 00112             Dn1_1, Dn1_1_Hi
00000024 00113             Dn1_2, Dn1_2_Hi
00114
00000026 00115             Dn2,   Dn2_Hi
00000028 00116             Dn2_1, Dn2_1_Hi
```

```
  0000002A   00117             Dn2_2, Dn2_2_Hi
             00118     ENDC
             00119
             00120     CBLOCK
  0000002C   00121             A11, A11_Hi
  0000002E   00122             A12, A12_Hi
  00000030   00123             B10, B10_Hi
  00000032   00124             B11, B11_Hi           ; 1st Section Filter Co-efficients
  00000034   00125             B12, B12_Hi
             00126
  00000036   00127             A21, A21_Hi
  00000038   00128             A22, A22_Hi
  0000003A   00129             B20, B20_Hi
  0000003C   00130             B21, B21_Hi           ; 2nd Section Filter Co-efficients
  0000003E   00131             B22, B22_Hi
             00132     ENDC
             00133
             00134     CBLOCK
  00000040   00135             X, X1                 ; 16 bits of input stream
  00000042   00136             Y, Y1                 ; 16 bits of filter output
             00137
  00000044   00138             ACC, ACC1, ACC2, ACC3 ; 32 bit accumulator for computations
             00139     ENDC
             00140 ;
  00000002   00141 FltStage  .set  2
  0000000A   00142 NumCoeff  equ   (5*FltStage)      ; 5 Co-eff per stage
             00143 ;
  00000001   00144 F         equ   1
             00145 ;
  00000001   00146 LPASS     .set  TRUE
  00000000   00147 HPASS     .set  FALSE
  00000000   00148 BPASS     .set  FALSE             ; select the desired filter type
  00000000   00149 BSTOP     .set  FALSE
             00150 ;
  00000001   00151 SIGNED  equ     TRUE              ; Set This To 'TRUE' for signed
             00152 ;                                 ; multiplication and 'FALSE' for unsigned.
             00153 ;
             00154 ;*****************************************************************
             00155 ;               Test Program For Low Pass Filter
             00156 ;*****************************************************************
             00157
0000         00158         ORG     0x0000
             00159 ;
0000         00160 start
0000 E00D    00161         call    initFilter
0001 B000    00162         movlw   0x00
0002 0140    00163         movwf   X
0003 B07F    00164         movlw   0x7f               ; set initial Xn = X(0) = 0x7f00
0004 0141    00165         movwf   X+BB1              ; test for impulse response
             00166 ;
0005         00167 NextPoint
0005 E022    00168         call    IIR_Filter
0006 A442    00169         tlwt    _LOW,Y
0007         00170 tracePoint
0007 AE43    00171         tablwt  _HIGH,0,Y+BB1
0008 0000    00172         nop
0009 2940    00173         clrf    X, F               ; set X(n) = 0 , n <> 0
000A 2941    00174         clrf    X+BB1, F           ; for simulating an Impulse
000B C005    00175         goto    NextPoint
             00176 ;
000C C00C    00177 self    goto    self
             00178 ;
             00179 ;*****************************************************************
             00180 ;
000D         00181 initFilter
             00182 ;
```

```
                00183 ;   At first read the Filter Co-efficients from Prog. Mem to Data RAM
                00184 ;
                00185         #if     LPASS
000D B0C1       00186                 movlw    LOW     _coeff_lpass
000E 010D       00187                 movwf    TBLPTRL
000F B001       00188                 movlw    HIGH    _coeff_lpass
0010 010E       00189                 movwf    TBLPTRH
                00190         #endif
                00191         #if     HPASS
                00192                 movlw    LOW     _coeff_hpass
                00193                 movwf    TBLPTRL
                00194                 movlw    HIGH    _coeff_hpass
                00195                 movwf    TBLPTRH
                00196         #endif
                00197         #if     BPASS
                00198                 movlw    LOW     _coeff_bpass
                00199                 movwf    TBLPTRL
                00200                 movlw    HIGH    _coeff_bpass
                00201                 movwf    TBLPTRH
                00202         #endif
                00203         #if     BSTOP
                00204                 movlw    LOW     _coeff_bstop
                00205                 movwf    TBLPTRL
                00206                 movlw    HIGH    _coeff_bstop
                00207                 movwf    TBLPTRH
                00208         #endif
                00209
                00210 ;
0011 B02C       00211         movlw    A11
0012 0101       00212         movwf    FSR0
0013 8404       00213         bsf      ALUSTA,FS0
0014 8D04       00214         bcf      ALUSTA,FS1             ; auto increment
                00215 ;
                00216 ; Read Filter Co-efficients from Program Memory
                00217 ;
0015 B00A       00218         movlw    NumCoeff
0016 A92C       00219         tablrd   _LOW,_INC,A11        ; garbage
0017            00220 NextCoeff
0017 A000       00221         tlrd     _LOW,INDF0
0018 AB00       00222         tablrd   _HIGH,_INC,INDF0
0019 170A       00223         decfsz   WREG, F
001A C017       00224         goto     NextCoeff
                00225 ;
                00226 ; Initialize “Dn”s to zero
                00227 ;
001B B020       00228         movlw    Dn1
001C 0101       00229         movwf    FSR0
001D B00C       00230         movlw    6*FltStage
001E            00231 NextClr
001E 2900       00232         clrf     INDF0, F
001F 170A       00233         decfsz   WREG, F
0020 C01E       00234         goto     NextClr
                00235 ;
0021 0002       00236         return
                00237 ;
                00238 ;*****************************************************************
                00239 ;                     1st Cascade Section
                00240 ;*****************************************************************
                00241 ;
0022            00242 IIR_Filter
                00243 ;
                00244 ; Compute  D(n) = X(n) + A1*D(n-1) + A2*D(n-2)
                00245 ;          Since the filter constants are computated in Q15 format,
                00246 ;          X(n) must be multiplied by 2**15 and then added to the
                00247 ;          other terms.
                00248 ;
```

```
            00249 ; Move Input to accumulator after proper scaling
            00250 ;
0022 8804   00251         bcf     ALUSTA,C
0023 1941   00252         rrcf    X+BB1, F
0024 1940   00253         rrcf    X, F
0025 290A   00254         clrf    WREG, F    ; Scale the input X
0026 190A   00255         rrcf    WREG, F
0027 0145   00256         movwf   ACC+BB1
0028 6A40   00257         movfp   X,WREG
0029 0146   00258         movwf   ACC+BB2
002A 6A41   00259         movfp   X+BB1,WREG
002B 0147   00260         movwf   ACC+BB3    ; ACC = scaled input : X*(2**15)
            00261 ;
            00262 ; 1st Biquad filter section
            00263 ;
            00264         BIQUAD  A11,A12,B10,B11,Dn1,Dn1_1,Dn1_2,1
                M
                M ;
                M ; Compute Ax2*D(n-2)
                M ;
                M   MOVFP16 Dn1_2,AARG       ; D(n-2) = multiplier
                M
002C 7C24       M   MOVFP   Dn1_2+B0,AARG+B0 ; move A(B0) to B(B0)
002D 7D25       M   MOVFP   Dn1_2+B1,AARG+B1 ; move A(B1) to B(B1)
                M
                M   MOVFP16 A12,BARG         ; A2 = multiplicand
                M
002E 7E2E       M   MOVFP   A12+B0,BARG+B0   ; move A(B0) to B(B0)
002F 7F2F       M   MOVFP   A12+B1,BARG+B1   ; move A(B1) to B(B1)
                M
0030 E0AF       M   call    DblMult          ; (ACCd,ACCc) = A2*D(n-2)
                M ;
                M ; Add product to output of 1st section
                M ; Save result in 32 bit Accumulator
                M ;
                M   ADD32   DPX,ACC
                M
0031 6A18       M   MOVFP   DPX+B0,WREG       ; get lowest byte of a into w
0032 0F44       M   ADDWF   ACC+B0, F         ; add lowest byte of b, save in b(B0)
0033 6A19       M   MOVFP   DPX+B1,WREG       ; get 2nd byte of a into w
0034 1145       M   ADDWFC  ACC+B1, F         ; add 2nd byte of b, save in b(B1)
0035 6A1A       M   MOVFP   DPX+B2,WREG       ; get 3rd byte of a into w
0036 1146       M   ADDWFC  ACC+B2, F         ; add 3rd byte of b, save in b(B2)
0037 6A1B       M   MOVFP   DPX+B3,WREG       ; get 4th byte of a into w
0038 1147       M   ADDWFC  ACC+B3, F         ; add 4th byte of b, save in b(B3)
                M
                M ;
                M ;  Compute A1*D(n-1)
                M ;
                M   MOVFP16  Dn1_1,AARG       ; AARG = D(n-2) = multiplier
                M
0039 7C22       M   MOVFP   Dn1_1+B0,AARG+B0  ; move A(B0) to B(B0)
003A 7D23       M   MOVFP   Dn1_1+B1,AARG+B1  ; move A(B1) to B(B1)
                M
                M   MOVFP16  A11,BARG         ; BARG = A2 = multiplicand
                M
003B 7E2C       M   MOVFP   A11+B0,BARG+B0    ; move A(B0) to B(B0)
003C 7F2D       M   MOVFP   A11+B1,BARG+B1    ; move A(B1) to B(B1)
                M
003D E0AF       M   call    DblMult           ; (ACCd,ACCc) = A1*D(n-1)
                M ;
                M ; Compute A1*D(n-1) + A2*D(n-2) + output of previous section
                M ;       multiplications already done, so simply perform a 32 bit add
                M ;       of previously obtained multiplication results
                M ;
                M   ADD32   DPX,ACC                ; ACC = A1*D(n-1) + A2*D(n-2) +
```

```
                M                                       ;           (output of 1st section)
003E 6A18       M   MOVFP   DPX+B0,WREG                 ; get lowest byte of a into w
003F 0F44       M   ADDWF   ACC+B0, F                   ; add lowest byte of b, save in b(B0)
0040 6A19       M   MOVFP   DPX+B1,WREG                 ; get 2nd byte of a into w
0041 1145       M   ADDWFC  ACC+B1, F                   ; add 2nd byte of b, save in b(B1)
0042 6A1A       M   MOVFP   DPX+B2,WREG                 ; get 3rd byte of a into w
0043 1146       M   ADDWFC  ACC+B2, F                   ; add 3rd byte of b, save in b(B2)
0044 6A1B       M   MOVFP   DPX+B3,WREG                 ; get 4th byte of a into w
0045 1147       M   ADDWFC  ACC+B3, F                   ; add 4th byte of b, save in b(B3)
                M
                M ;
                M ;
                M ; save the upper 16 bits  of D(n) from the 32 bit accumulator
                M ; left shift the result by 1, to adjust the decimal point after
                M ; a Q15*Q15 multiplication
                M ;
0046 1A45       M   rlcf    ACC+B1,w
0047 1A46       M   rlcf    ACC+B2,w
0048 0120       M   movwf   Dn1
0049 1A47       M   rlcf    ACC+B3,w                    ; decimal adjust ( mult by 2)
004A 0121       M   movwf   Dn1+B1
                M ;
                M ; Compute B2 * [D(n) + D(n-2)]
                M ;
                M   #if B0_EQUALS_B2
                M
                M    ADD16ACC  Dn1_2,Dn1,AARG     ; AARG = Dn + D(n-2) = multiplier
                M
004B 6A24       M    movfp   Dn1_2+B0,WREG
004C 0E20       M    addwf   Dn1+B0,w
004D 011C       M    movwf   AARG+B0
004E 6A25       M    movfp   Dn1_2+B1,WREG
004F 1021       M    addwfc  Dn1+B1,w
0050 011D       M    movwf   AARG+B1
                M
                M    MOVFP16  B10,BARG              ; BARG = A2 = multiplicand
                M
0051 7E30       M    MOVFP   B10+B0,BARG+B0         ; move A(B0) to B(B0)
0052 7F31       M    MOVFP   B10+B1,BARG+B1         ; move A(B1) to B(B1)
                M
0053 E0AF       M    call    DblMult                ; (ACCd,ACCc) = B2*[D(n)+D(n-2)]
                M    MOVPF32  DPX,ACC
                M
0054 5844       M    MOVPF   DPX+B0,ACC+B0          ; move A(B0) to B(B0)
0055 5945       M    MOVPF   DPX+B1,ACC+B1          ; move A(B1) to B(B1)
0056 5A46       M    MOVPF   DPX+B2,ACC+B2          ; move A(B2) to B(B2)
0057 5B47       M    MOVPF   DPX+B3,ACC+B3          ; move A(B3) to B(B3)
                M
                M
                M    #else
                M
                M     MOVFP16 B10,BARG
                M     MOVFP16 Dn1,AARG
                M     call    DblMult                ; B0*D(n)
                M     MOVPF32 DPX,ACC
                M
                M     MOVFP16 Bx2,BARG
                M     MOVFP16 Dn1_2,AARG
                M     call    DblMult                ; B2*D(n-2)
                M     ADD32   DPX,ACC
                M
                M     #endif
                M ;
                M ; Shift down D(n-1) to D(n-2) after D(n-2) usage is no longer required.
                M ; This way in the next iteration D(n-2) is equal to the present D(n-1)
                M ;
```

```
0058 7C22        M    movfp    Dn1_1,AARG+B0
0059 5C24        M    movpf    AARG+B0,Dn1_2       ; Shift down D(n-1)
005A 7D23        M    movfp    Dn1_1+B1,AARG+B1
005B 5D25        M    movpf    AARG+B1,Dn1_2+B1    ; AARG = D(n-1) = multiplier
                 M
                 M    MOVFP16  B11,BARG            ; BARG = B1 = multiplicand
                 M
005C 7E32        M    MOVFP   B11+B0,BARG+B0       ; move A(B0) to B(B0)
005D 7F33        M    MOVFP   B11+B1,BARG+B1       ; move A(B1) to B(B1)
                 M
                 M
005E E0AF        M    call    DblMult              ; (ACCd,ACCc) = B1*D(n-1)
                 M ;
                 M ; Compute Output Y = B1*D(n-1) + B2*D(n-2) + B0*D(n)
                 M ;                  = B1*D(n-1) + B0*[D(n) + D(n-2)]
                 M ; Since all multiplications are already done, simply perform a
                 M ; 32 bit addition
                 M ;
                 M    ADD32   DPX,ACC ; ACC = B1*D(n-1) + B2*D(n-2) + B0*D(n)
                 M
005F 6A18        M    MOVFP   DPX+B0,WREG          ; get lowest byte of a into w
0060 0F44        M    ADDWF   ACC+B0, F            ; add lowest byte of b, save in b(B0)
0061 6A19        M    MOVFP   DPX+B1,WR            ; get 2nd byte of a into w
0062 1145        M    ADDWFC  ACC+B1, F            ; add 2nd byte of b, save in b(B1)
0063 6A1A        M    MOVFP   DPX+B2,WREG          ; get 3rd byte of a into w
0064 1146        M    ADDWFC  ACC+B2, F            ; add 3rd byte of b, save in b(B2)
0065 6A1B        M    MOVFP   DPX+B3,WREG          ; get 4th byte of a into w
0066 1147        M    ADDWFC  ACC+B3, F            ; add 4th byte of b, save in b(B3)
                 M
                 M ;
                 M ; Shift down D(n) to D(n-1) so that in the next iteration, the new
                 M ; D(n-1) is the present D(n)
                 M ;
                 M    MOV16   Dn1,Dn1_1 ; Shift down D(n) to D(n-1)
                 M
0067 6A20        M    MOVFP   Dn1+B0,WREG          ; get byte of a into w
0068 0122        M    MOVWF   Dn1_1+B0             ; move to b(B0)
0069 6A21        M    MOVFP   Dn1+B1,WREG          ; get byte of a into w
006A 0123        M    MOVWF   Dn1_1+B1             ; move to b(B1)
                 M
                 M ;
              00265 ;
              00266 ; 2nd Biquad filter section
              00267 ;
              00268    BIQUAD  A21,A22,B20,B21,Dn2,Dn2_1,Dn2_2,2
                 M
                 M ;
                 M ; Compute Ax2*D(n-2)
                 M ;
                 M    MOVFP16  Dn2_2,AARG          ; D(n-2) = multiplier
                 M
006B 7C2A        M    MOVFP   Dn2_2+B0,AARG+B0     ; move A(B0) to B(B0)
006C 7D2B        M    MOVFP   Dn2_2+B1,AARG+B1     ; move A(B1) to B(B1)
                 M
                 M    MOVFP16  A22,BARG            ; A2 = multiplicand
                 M
006D 7E38        M    MOVFP   A22+B0,BARG+B0       ; move A(B0) to B(B0)
006E 7F39        M    MOVFP   A22+B1,BARG+B1       ; move A(B1) to B(B1)
                 M
006F E0AF        M    call    DblMult              ; (ACCd,ACCc) = A2*D(n-2)
                 M ;
                 M ; Add product to output of 1st section
                 M ; Save result in 32 bit Accumulator
                 M ;
                 M    ADD32    DPX,ACC
                 M
```

```
0070 6A18        M    MOVFP   DPX+B0,WREG          ; get lowest byte of a into w
0071 0F44        M    ADDWF   ACC+B0, F            ; add lowest byte of b, save in b(B0)
0072 6A19        M    MOVFP   DPX+B1,WREG          ; get 2nd byte of a into w
0073 1145        M    ADDWFC  ACC+B1, F            ; add 2nd byte of b, save in b(B1)
0074 6A1A        M    MOVFP   DPX+B2,WREG          ; get 3rd byte of a into w
0075 1146        M    ADDWFC  ACC+B2, F            ; add 3rd byte of b, save in b(B2)
0076 6A1B        M    MOVFP   DPX+B3,WREG          ; get 4th byte of a into w
0077 1147        M    ADDWFC  ACC+B3, F            ; add 4th byte of b, save in b(B3)
                 M
                 M ;
                 M ;  Compute A1*D(n-1)
                 M ;
                 M    MOVFP16  Dn2_1,AARG          ; AARG = D(n-2) = multiplier
                 M
0078 7C28        M    MOVFP   Dn2_1+B0,AARG+B0     ; move A(B0) to B(B0)
0079 7D29        M    MOVFP   Dn2_1+B1,AARG+B1     ; move A(B1) to B(B1)
                 M
                 M    MOVFP16  A21,BARG            ; BARG = A2 = multiplicand
                 M
007A 7E36        M    MOVFP   A21+B0,BARG+B0       ; move A(B0) to B(B0)
007B 7F37        M    MOVFP   A21+B1,BARG+B1       ; move A(B1) to B(B1)
                 M
007C E0AF        M    call    DblMult              ; (ACCd,ACCc) = A1*D(n-1)
                 M ;
                 M ; Compute A1*D(n-1) + A2*D(n-2) + output of previous section
                 M ;       multiplications already done, so simply perform a 32 bit add
                 M ;       of previously obtained multiplication results
                 M ;
                 M   ADD32   DPX,ACC              ; ACC = A1*D(n-1) + A2*D(n-2) +
                 M                                ;              (output of 1st section)
007D 6A18        M   MOVFP   DPX+B0,WREG          ; get lowest byte of a into w
007E 0F44        M   ADDWF   ACC+B0, F            ; add lowest byte of b, save in b(B0)
007F 6A19        M   MOVFP   DPX+B1,WREG          ; get 2nd byte of a into w
0080 1145        M   ADDWFC  ACC+B1, F            ; add 2nd byte of b, save in b(B1)
0081 6A1A        M   MOVFP   DPX+B2,WREG          ; get 3rd byte of a into w
0082 1146        M   ADDWFC  ACC+B2, F            ; add 3rd byte of b, save in b(B2)
0083 6A1B        M   MOVFP   DPX+B3,WREG          ; get 4th byte of a into w
0084 1147        M   ADDWFC  ACC+B3, F            ; add 4th byte of b, save in b(B3)
                 M
                 M ;
                 M ;
                 M ; save the upper 16 bits  of D(n) from the 32 bit accumulator
                 M ; left shift the result by 1, to adjust the decimal point after
                 M ; a Q15*Q15 multiplication
                 M ;
0085 1A45        M   rlcf    ACC+B1,w
0086 1A46        M   rlcf    ACC+B2,w
0087 0126        M   movwf   Dn2
0088 1A47        M   rlcf    ACC+B3,w             ; decimal adjust ( mult by 2)
0089 0127        M   movwf   Dn2+B1
                 M ;
                 M ; Compute B2 * [D(n) + D(n-2)]
                 M ;
                 M   #if B0_EQUALS_B2
                 M
                 M   ADD16ACC Dn2_2,Dn2,AARG      ; AARG = Dn + D(n-2) = multiplier
                 M
008A 6A2A        M   movfp   Dn2_2+B0,WREG
008B 0E26        M   addwf   Dn2+B0,w
008C 011C        M   movwf   AARG+B0
008D 6A2B        M   movfp   Dn2_2+B1,WREG
008E 1027        M   addwfc  Dn2+B1,w
008F 011D        M   movwf   AARG+B1
                 M
                 M   MOVFP16 B20,BARG            ; BARG = A2 = multiplicand
                 M
```

```
0090 7E3A        M   MOVFP   B20+B0,BARG+B0       ; move A(B0) to B(B0)
0091 7F3B        M   MOVFP   B20+B1,BARG+B1       ; move A(B1) to B(B1)
                 M
0092 E0AF        M   call    DblMult              ; (ACCd,ACCc) = B2*[D(n)+D(n-2)]
                 M   MOVPF32 DPX,ACC
                 M
0093 5844        M   MOVPF   DPX+B0,ACC+B0        ; move A(B0) to B(B0)
0094 5945        M   MOVPF   DPX+B1,ACC+B1        ; move A(B1) to B(B1)
0095 5A46        M   MOVPF   DPX+B2,ACC+B2        ; move A(B2) to B(B2)
0096 5B47        M   MOVPF   DPX+B3,ACC+B3        ; move A(B3) to B(B3)
                 M
                 M
                 M   #else
                 M
                 M   MOVFP16 B20,BARG
                 M   MOVFP16 Dn2,AARG
                 M   call    DblMult              ; B0*D(n)
                 M   MOVPF32 DPX,ACC
                 M
                 M   MOVFP16 Bx2,BARG
                 M   MOVFP16 Dn2_2,AARG
                 M   call    DblMult              ; B2*D(n-2)
                 M   ADD32   DPX,ACC
                 M
                 M   #endif
                 M ;
                 M ; Shift down D(n-1) to D(n-2) after D(n-2) usage is no longer required.
                 M ; This way in the next iteration D(n-2) is equal to the present D(n-1)
                 M ;
0097 7C28        M  movfp    Dn2_1,AARG+B0
0098 5C2A        M  movpf    AARG+B0,Dn2_2        ; Shift down D(n-1)
0099 7D29        M  movfp    Dn2_1+B1,AARG+B1
009A 5D2B        M  movpf    AARG+B1,Dn2_2+B1     ; AARG = D(n-1) = multiplier
                 M
                 M  MOVFP16  B21,BARG             ; BARG = B1 = multiplicand
                 M
009B 7E3C        M  MOVFP   B21+B0,BARG+B0        ; move A(B0) to B(B0)
009C 7F3D        M  MOVFP   B21+B1,BARG+B1        ; move A(B1) to B(B1)
                 M
                 M
009D E0AF        M  call    DblMult               ; (ACCd,ACCc) = B1*D(n-1)
                 M ;
                 M ; Compute Output Y = B1*D(n-1) + B2*D(n-2) + B0*D(n)
                 M ;                  = B1*D(n-1) + B0*[D(n) + D(n-2)]
                 M ; Since all multiplications are already done, simply perform a
                 M ; 32 bit addition
                 M ;
                 M   ADD32   DPX,ACC              ; ACC = B1*D(n-1) + B2*D(n-2) + B0*D(n)
                 M
009E 6A18        M   MOVFP   DPX+B0,WREG          ; get lowest byte of a into w
009F 0F44        M   ADDWF   ACC+B0, F            ; add lowest byte of b, save in b(B0)
00A0 6A19        M   MOVFP   DPX+B1,WREG          ; get 2nd byte of a into w
00A1 1145        M   ADDWFC  ACC+B1, F            ; add 2nd byte of b, save in b(B1)
00A2 6A1A        M   MOVFP   DPX+B2,WREG          ; get 3rd byte of a into w
00A3 1146        M   ADDWFC  ACC+B2, F            ; add 3rd byte of b, save in b(B2)
00A4 6A1B        M   MOVFP   DPX+B3,WREG          ; get 4th byte of a into w
00A5 1147        M   ADDWFC  ACC+B3, F            ; add 4th byte of b, save in b(B3)
                 M
                 M ;
                 M ; Shift down D(n) to D(n-1) so that in the next iteration, the new
                 M ; D(n-1) is the present D(n)
                 M ;
                 M   MOV16   Dn2,Dn2_1            ; Shift down D(n) to D(n-1)
                 M
00A6 6A26        M   MOVFP   Dn2+B0,WREG          ; get byte of a into w
00A7 0128        M   MOVWF   Dn2_1+B0             ; move to b(B0)
```

```
00A8 6A27        M   MOVFP   Dn2+B1,WREG          ; get byte of a into w
00A9 0129        M   MOVWF   Dn2_1+B1             ; move to b(B1)
                 M
                 M ;
                 00269 ;
                 00270 ; The filter output is now computed
                 00271 ; Save the Upper 16 Bits of 32 bit Accumulator into Y after
                 00272 ; adjusting the decimal point
                 00273 ;
                 00274
                 00275   MOV16   ACC+BB2,Y
                     M
00AA 6A46        M   MOVFP   ACC+BB2+B0,WREG      ; get byte of a into w
00AB 0142        M   MOVWF   Y+B0                 ; move to b(B0)
00AC 6A47        M   MOVFP   ACC+BB2+B1,WREG      ; get byte of a into w
00AD 0143        M   MOVWF   Y+B1                 ; move to b(B1)
                 M
                 00276 ;
00AE 0002        00277   return                   ; Output Y(n) computed
                 00278 ;
                 00279 ;*****************************************************************
                 00280 ;  Set SIGNED/UNSIGNED Flag Before Including 17c42MPY.mac
                 00281 ;
                 00282   #include        <17c42MPY.mac>
                 00178   LIST
                 00283
                 00284 ;*****************************************************************
                 00285 ;  Low Pass Filter Co-efficients
                 00286 ;
                 00287 ;
                 00288 ;                ELLIPTIC     LOWPASS FILTER
                 00289 ;
                 00290 ;                FILTER ORDER =    4
                 00291 ;                Sampling frequency =   2.000  KiloHertz
                 00292 ;
                 00293 ;                        BAND  1        BAND  2
                 00294 ;
                 00295 ;
                 00296 ; LOWER BAND EDGE         .00000         .60000
                 00297 ; UPPER BAND EDGE         .50000        1.00000
                 00298 ; NOMINAL GAIN           1.00000         .00000
                 00299 ; NOMINAL RIPPLE          .01000         .05000
                 00300 ; MAXIMUM RIPPLE          .00906         .04601
                 00301 ; RIPPLE IN dB            .07830      -26.74235
                 00302 ;
                 00303 ;  I    A(I,1)        A(I,2)        B(I,0)        B(I,1)        B(I,2)
                 00304 ;
                 00305 ;  1    -.133331       .167145       .285431       .462921       .285431
                 00306 ;       .147827       .765900       .698273       .499908       .698273
                 00307 ;
                 00308
01C1             00309 _coeff_lpass            ; co-efficients for 1st Cascade section
01C1 1111        00310         data     4369           ; -A11
01C2 EA9B        00311         data     -5477          ; -A12
01C3 2489        00312         data     9353           ; B10
01C4 3B41        00313         data     15169          ; B11
01C5 2489        00314         data     9353           ; B12
                 00315                         ; co-efficients for 2nd Cascade section
01C6 ED14        00316         data    -4844           ; -A21
01C7 9DF7        00317         data    -25097          ; -A22
01C8 5961        00318         data     22881          ; B20
01C9 3FFD        00319         data     16381          ; B21
01CA 5961        00320         data     22881          ; B22
                 00321 ;
                 00322 ;*****************************************************************
                 00323 ;
```

```
                00324 ;
                00325 ;*****************************************************************
                00326 ;  High Pass Filter Co-efficients
                00327 ;
                00328 ;
                00329 ;               ELLIPTIC    HIGHPASS FILTER
                00330 ;
                00331 ;               FILTER ORDER =    4
                00332 ;               Sampling frequency =   2.000  KiloHertz
                00333 ;
                00334 ;                       BAND  1        BAND  2
                00335 ;
                00336 ;
                00337 ; LOWER BAND EDGE         .00000         .50000
                00338 ; UPPER BAND EDGE         .40000        1.00000
                00339 ; NOMINAL GAIN            .00000        1.00000
                00340 ; NOMINAL RIPPLE          .04000         .02000
                00341 ; MAXIMUM RIPPLE          .03368         .01686
                00342 ; RIPPLE IN dB         -29.45335         .14526
                00343 ;
                00344 ;  I      A(I,1)        A(I,2)        B(I,0)        B(I,1)        B(I,2)
                00345 ;
                00346 ;  1     .276886       .195648       .253677      -.411407       .253677
                00347 ;  2    -.094299       .780396       .678650      -.485840       .678650
                00348 ;
                00349 ;
01CB            00350 _coeff_hpass            ; co-efficients for 1st Cascade section
01CB DC8F       00351         data     -9073          ; -A11
01CC E6F5       00352         data     -6411          ; -A12
01CD 2079       00353         data     8313           ; B10
01CE CB57       00354         data     -13481         ; B11
01CF 2079       00355         data     8313           ; B12
                00356 ;                       ; co-efficients for 2nd Cascade section
01D0 0C12       00357         data    3090            ; -A21
01D1 9C1C       00358         data    -25572          ; -A22
01D2 56DE       00359         data    22238           ; B20
01D3 C1D0       00360         data    -15920          ; B21
01D4 56DE       00361         data    22238           ; B22
                00362 ;
                00363 ;*****************************************************************
                00364 ;
                00365 ;*****************************************************************
                00366 ;  Band Pass Filter Co-efficients
                00367 ;
                00368 ;
                00369 ;               ELLIPTIC     BANDPASS FILTER
                00370 ;
                00371 ;               FILTER ORDER =    4
                00372 ;               Sampling frequency =   2.000  KiloHertz
                00373 ;
                00374 ;                       BAND  1        BAND  2        BAND  3
                00375 ;
                00376 ;
                00377 ; LOWER BAND EDGE         .00000         .30000         .90000
                00378 ; UPPER BAND EDGE         .10000         .70000        1.00000
                00379 ; NOMINAL GAIN            .00000        1.00000         .00000
                00380 ; NOMINAL RIPPLE          .05000         .05000         .05000
                00381 ; MAXIMUM RIPPLE          .03644         .03867         .03641
                00382 ; RIPPLE IN dB         -28.76779         .32956      -28.77647
                00383 ;
                00384 ;
                00385 ;  I      A(I,1)        A(I,2)        B(I,0)        B(I,1)        B(I,2)
                00386 ;
                00387 ;  1    -.936676       .550568       .444000      -.865173       .444000
                00388 ;  2     .936707       .550568       .615540      1.199402       .615540
                00389 ;
```

```
            00390
01D5        00391 _coeff_bpass                ; co-efficients for 1st Cascade section
01D5 3BF2   00392         data     30693/2          ; -A11
01D6 DCC4   00393         data     -18041/2         ; -A12
01D7 1C6A   00394         data     14549/2          ; B10
01D8 C8A1   00395         data     -28350/2         ; B11
01D9 1C6A   00396         data     14549/2          ; B12
            00397 ;                              ; co-efficients for 2nd Cascade section
01DA C40D   00398         data    -30694/2          ; -A21
01DB DCC4   00399         data    -18041/2          ; -A22
01DC 2765   00400         data     20170/2          ; B20
01DD 4CC3   00401         data     39302/2          ; B21
01DE 2765   00402         data     20170/2          ; B22
            00403 ;
            00404 ;*****************************************************************
            00405 ;
            00406 ;*****************************************************************
            00407 ;  Band Stop Filter Co-efficients
            00408 ;
            00409 ;
            00410 ;                      ELLIPTIC    BANDSTOP FILTER
            00411 ;
            00412 ;               FILTER ORDER =    4
            00413 ;               Sampling frequency =   2.000  KiloHertz
            00414 ;
            00415 ;                       BAND  1        BAND  2        BAND  3
            00416 ;
            00417 ;
            00418 ;LOWER BAND EDGE         .00000         .45000         .70000
            00419 ;UPPER BAND EDGE         .30000         .55000        1.00000
            00420 ;NOMINAL GAIN           1.00000         .00000        1.00000
            00421 ;NOMINAL RIPPLE          .05000         .05000         .05000
            00422 ;MAXIMUM RIPPLE          .03516         .03241         .03517
            00423 ;RIPPLE IN dB            .30015      -29.78523         .30027
            00424 ;
            00425 ;
            00426 ;  I      A(I,1)        A(I,2)        B(I,0)        B(I,1)        B(I,2)
            00427 ;
            00428 ;  1     .749420       .583282       .392685       .087936       .392685
            00429 ;  2     -.749390       .583282      1.210022      -.270935      1.210022
            00430 ;
01DF        00431 _coeff_bstop                ; co-efficients for 1st Cascade section
01DF D00A   00432         data     -24557/2         ; -A11
01E0 DAAC   00433         data     -19113/2         ; -A12
01E1 1922   00434         data     12868/2          ; B10
01E2 05A0   00435         data     2881/2           ; B11
01E3 1922   00436         data     12868/2          ; B12
            00437                              ; co-efficients for 2nd Cascade section
01E4 2FF6   00438         data    24556/2           ; -A21
01E5 DAAC   00439         data    -19113/2          ; -A22
01E6 4D71   00440         data    39650/2           ; B20
01E7 EEA9   00441         data    -8878/2           ; B21
01E8 4D71   00442         data    39650/2           ; B22
            00443 ;
            00444 ;*****************************************************************
            00445
            00446         END
```

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

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXX------- ----------------

All other memory blocks unused.

Program Memory Words Used:   489


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.

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

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

WORLDWIDE SALES AND SERVICE

AMERICAS

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

EUROPE

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

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

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

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

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

03/01/02