TEXAS INSTRUMENTS

Application Report
SPRA482

Programmable Double Biquad Filter for Tone Detection on Fixed Point DSPs

Digital Signal Processing Solutions

Abstract

The filters described here are user programmable double biquad filters for tone detection. The filters are implemented on the Texas Instruments (TI™) TMS320C2xx digital signal processor (DSP). The filter program includes an energy estimation stage. Examples of applications are CPTD (call progress tone detection), fax tone detection, answer tone detection, etc. for telephony or modem.

Contents

Figures

Tables

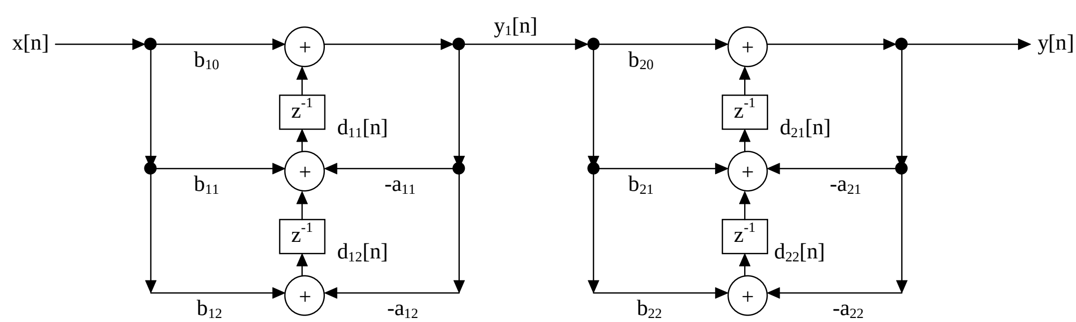
Introduction

The aim of this report is to describe tone detection by means of a programmable passband filter in combination with an energy estimation stage. The filtering operation described below allows the detection of single frequencies (with a tolerance band of ±x%) or a frequency band (e.g., tones used in the telephone net: dial tone, busy tone, etc.). All parameters related to the tone detection process are user programmable. These parameters include filter coefficients, scale factors and detection thresholds. The following sections describe the filter structure used for passband filtering , the different steps involved in the detection process, the software carrying out these operations, the interface between filter programs and application S/W layer and processor resources required.

Filter Structure and Difference Equations

This section gives a theoretical overview of the IIR filter used in the tone detection process. The filter structure implemented here is the so-called transposed form cascade structure, which is shown in Figure 1.

Figure 1. Transposed Form Cascade Structure for N=4



The corresponding difference equations are:

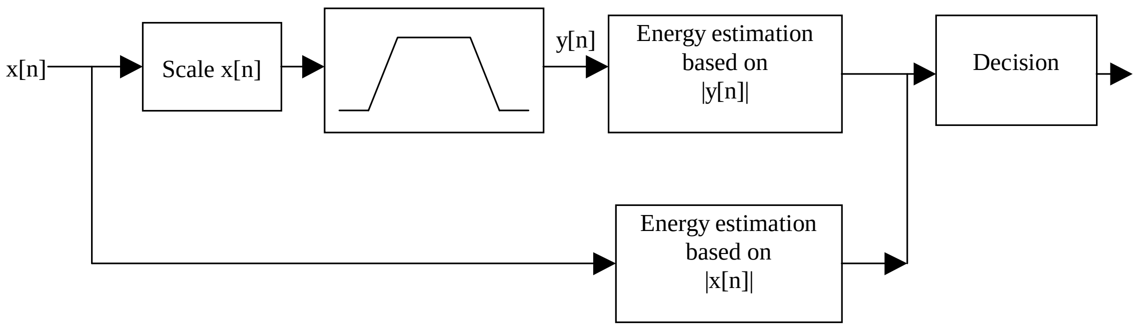
$$
\begin{aligned}
y_0 &= x_0 \\
y_i &= b_{i0} y_{i-1}[n] + d_{i1}[n-1] \\
d_{i1}[n] &= b_{i1} y_{i-1}[n] - a_{i1} y_i[n] + d_{i2}[n-1] \\
d_{i2}[n] &= b_{i2} y_{i-1}[n] - a_{i2} y_i[n] \\
i &= 1,2,\ldots,\left\lceil \frac{N+1}{2} \right\rceil \\
y[n] &= y_{\lceil (N+1)/2 \rceil}[n]
\end{aligned}
\qquad \text{(equation 1)}
$$

x[n] denotes the filter input, $y_i[n]$ is the filter output after the filter stage *i* and y[n] the global filter output. By means of a filter design tool using the filter structure shown in Figure 1 we can determine the filter coefficients of a double biquad filter (N=4) for the desired passband. An example of filter design will be given in the next section.

Description of the Tone Detection Procedure

The tone detection procedure can be divided into different stages as shown in Figure 2.

Figure 2. Stages of the Tone Detection Operation



First the main filtering operation is carried out. This consists of bandpass filtering the scaled input signal. This is followed by an energy estimation by means of exponential filters based on the filtered signal and the global signal.

The exponential filters are given by:

$$FilterOut[n] = \alpha|y[n]| + (1-\alpha)FilterOut[n-1]$$
$$TotOut[n] = \alpha|x[n]| + (1-\alpha)TotOut[n-1] \qquad \text{(equation 2)}$$

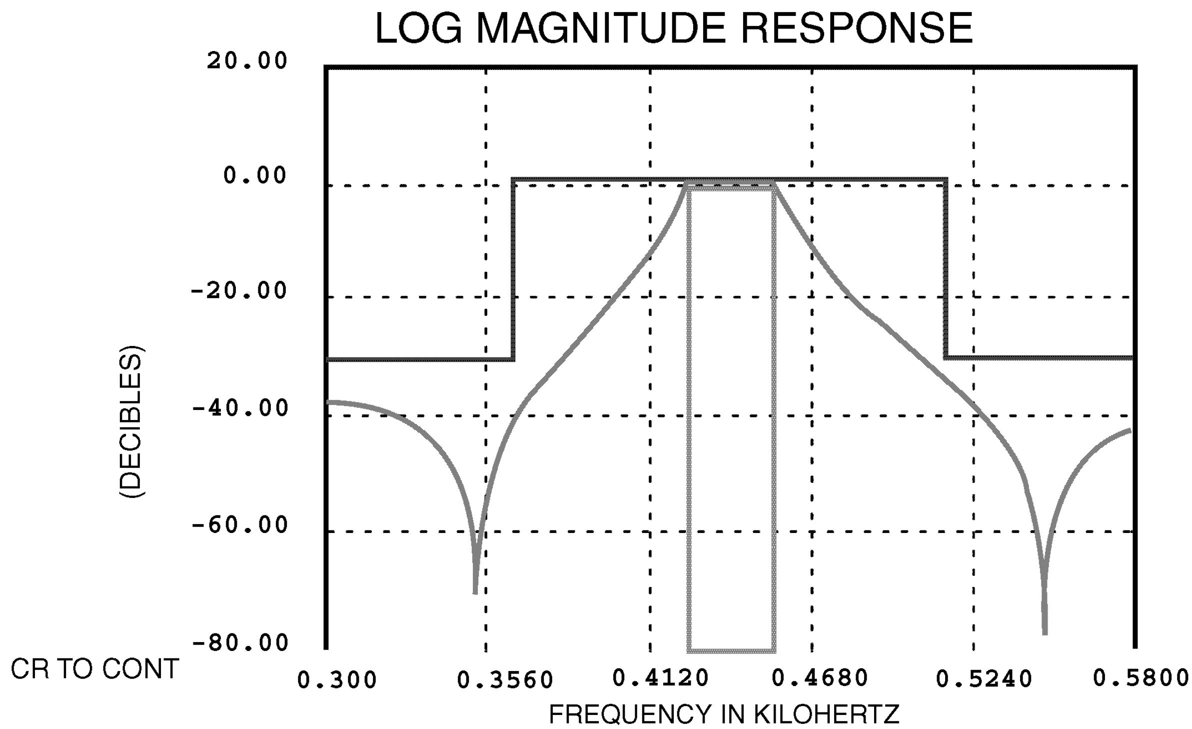
The last stage consists of the decision whether a tone has been detected or not. The detection criteria is specified as follows

$$FilterOut[n] \times Threshold \geq TotOut[n] \qquad \text{(equation 3)}$$

The bandpass filter is a double biquad filter based on equation 1. The filter coefficients have to be previously determined by means of a filter design tool. The bandpass filter is characterized by seven parameters: the sampling frequency, the lower and upper stopband frequencies and the lower and upper passband frequencies, as well as the passband ripple and the stopband ripple.

An example of filter design is shown in Figure 3. The filter coefficients generated by the design tool are stored in a C-header file, **Filters.h**. Before running the filter for the first time, initialization routines contained in the file **initFilt.c** have to be executed. The different programs and the interface C-Assembly language will be described in the following sections.

Figure 3. Example of Filter Design for Dial Tone Detector



Sampling frequency Fs=9.6kHz
Lower stopband frequency=365Hz
Lower passband frequency=425Hz
Upper passband frequency=455Hz
Upper stopband frequency=515Hz

Passband ripple=0.1
Stopband ripple=0.03

INFINITE IMPULSE RESPONSE (IIR),ELLIPTIC BANDPASS FILTER
UNQUANTIZED COEFFICIENTS,   FILTER ORDER = 4
SAMPLING FREQUENCY = 9.600 KILOHERTZ

I	A(I,1)	A(I,2)	B(I,0)	B(I,1)	B(I,2)
1	-1.903748	.990570	.144363	-.270309	.144363
2	-1.913757	991089	.141541	-.275604	.141541

```
/* DialFilter*/
/* Quantized coefficients: Q14 */
/* Coefficients for 1st biquad */
#define    Dial1_B0    2365
#define    Dial1_B1    -4428
#define    Dial1_B2    2365
#define    Dial1_A1    31191
#define    Dial1_A2    -16229

/* Coefficients for 2nd biquad */
#define    Dial2_B0    2319
#define    Dial2_B1    -4515
#define    Dial2_B2    2319
#define    Dial2_A1    31356
#define    Dial2_A2    -16238
```

Description of Filter Programs

This section deals with the filter programs and the parameters that have to be determined before running the filter. All parameters that directly influence detection are programmable. These parameters include the filter coefficients, the scale factor for the input sample applied to the filter and the detection threshold. They can be found in the file **Filters.h**, given in Appendix A. This file is used by the initialization routine **initFilt.c**, described below. As an example, a filter for dial and busy tone detection is implemented.

initFilt.c

This routine initializes the filter variables with the fixed parameter values. All variable names are chosen according to the following convention: **FilterName**Variable.

Example: filter name=Dial, variable=Threshold -> variable name=DialThreshold.

Each filter has the following variables:
---*Filter[14]:* Array of fourteen elements for filter coefficients and delays
---*Shift:* Scale factor for input sample
---*Threshold:* Factor used in the decision stage (cf. equation 3)
---*In:* Input to the exponential filter after bandpass filtering (|y[n]|)
---*Out:* Output of the exponential filter applied to decision stage
where --- stands for the filter name.

The elements of ---Filter[14] for a double biquad as shown in Figure 1 are:
---Filter[0]=d_{11}
---Filter[1]=d_{12}
---Filter[2]=d_{21}
---Filter[3]=d_{22}
---Filter[4]=b_{10}
---Filter[5]=b_{11}
---Filter[6]=$-a_{11}$
---Filter[7]=$-a_{12}$
---Filter[8]=b_{12}
---Filter[9]=b_{21}
---Filter[10]=b_{22}
---Filter[11]=$-a_{21}$
---Filter[12]=$-a_{22}$
---Filter[13]=b_{22}

The delays d_{11} through d_{22} are initialized to zero. The elements ---Filter[4] through ---Filter[13] are initialized with the filter coefficients specified in **Filters.h**. Likewise ---Shift and ---Threshold are set to the specified parameter values. The input and the output of the exponential filters are initialized to zero.

Biquad.asm

The file biquad.asm contains different stages of the filtering operation described in Figure 2. Several filters may be implemented in parallel. Currently, examples of dial tone and fax tone detection are implemented. The routine that calls the filters is named **CPTD**. This routine is called in the sample interrupt at F_s (sampling frequency), which implies that the filter design has been previously carried out with the same sampling frequency.

In the tone detection process, the absolute value of the input sample is first computed for the estimation of the global energy. In case of M ($1<M\leq 4$) filters being implemented in parallel, the exponential filter is based on the sum of M+1 absolute values of the input samples in order to reduce the computational load (MIPS). This means that the input of exponential filter for the global energy is now given by:

$$x_{M+1}[n] = \sum_{i=0}^{M} |x[i]| \qquad \text{(equation 4)}$$

After that, frequency filtering is carried out for each filter. Prior to the filtering operation, the input sample has to be scaled and the pointers to the filter coefficients and delays have to be set up. This is done by means of a macro **Filter** with the argument **Name**, where Name may be (for example) *Dial*. First the input sample is right shifted by the amount 16-**Name**Shift, i.e. a parameter value of 16 means no shift, 15 means a right shift by 1, 14 means right shift by 2 and so on. After scaling, AR0 is set to point to the first filter delay (**Name**Filter[0]) and AR1 to the first filter coefficient (**Name**Filter[4]). The PREG output shift is set to 1 (spm 1) and the sign extension mode is set (ssxm). Before the call of the basic filtering routine **BIQUAD**, the current ARP has to be set to AR1 (pointer to filter coefficients). **BIQUAD** performs the cascaded IIR filter according to equation 1 (N=4). This routine is called for each filter. For the fixed-point computation all filter coefficients are in Q14 format, the input sample and filter delays are assumed to be in Q15 format. The output of each filter is the input of the corresponding exponential filter for energy estimation in the passband.

These inputs of the exponential filters, after frequency filtering, are given by:

$$y_{M+1}[n] = \sum_{i=0}^{M} |y[i]| \qquad \text{(equation 5)}$$

As now only one exponential filter is called once every M samples, the complete routine uses about (M-1)*50 cycles less than the computation of all exponential filters in parallel (MIPS and memory occupation are given in more detail later).

Finally the exponential filters are computed and a decision is made whether there is enough energy in the specified passband or not. For the energy estimation in the different passbands a macro called **TestOut** with the argument **Name** (the same as for the macro **Filter**) is used.

The output of the exponential filter is calculated as specified by equation 2 with α=1/64. Then the output is compared to the output of the exponential filter for the global input:

$$(NameThreshold \times NameOut) \geq TotOut \times 16 \Rightarrow \text{Detection} \qquad \text{(equation 6)}$$

The global filter output is multiplied by 16 to allow more precision for the detection threshold. For instance, if the energy in the passband should be more than half of the global energy for detection, then the **Name**Threshold must be set to 32. Increasing the threshold means increasing the passband for detection.

In case of detection, a bit is set in the variable **CptdFilter** for each passband. Table 1 gives an example of bit masks for four different filters.

Table 1. Bit Masks For The Different Filters

Name	Value of CptdFilter
Filter1	1
Filter2	2
Filter3	4
Filter4	8

The variable **CptdFilter** can be used in a program on the upper level for a timing check. The interface between C-programs and the filter program in assembly language will be described in the next section.

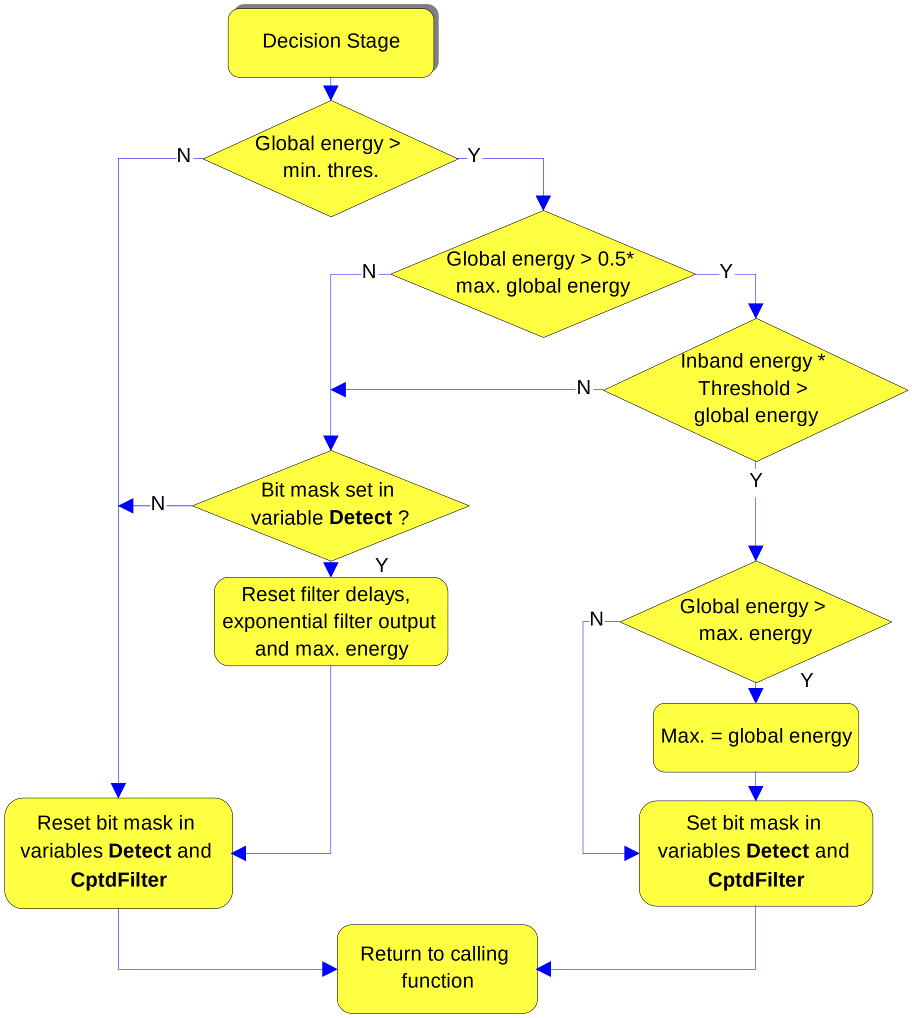
To enhance tone detection some more tests are carried out. The energy comparison is not executed if the global energy does not exceed a minimum threshold specified by the variable **MinEng**. This threshold sets the absolute value for the minimum input signal level that will be taken into account.

Another point is the fast detection of energy transitions such as off/on and on/off transitions for the busy tone. Due to the group delay of the passband filters, the comparison of energies (equation 6) may still result in detection even if there is no signal at the input any more. This is why an adaptive threshold test is carried out to detect energy transitions on/off. In fact if the output of the exponential filter for global energy estimation falls below the half of the maximum value determined during a detection phase then the decision result is non-detection:

$$TotOut \leq 0.5 \times TotMax \Rightarrow \text{No Detection} \qquad \text{(equation 7)}$$

The different steps involved in the detection phase are summarized in the flow chart below. The decision stage is executed separately for each filter.

Figure 4. Flow Chart of the Decision Stage in the Detection Process



Summary of Programmable Parameters

The parameters which are user programmable are the filter coefficients, scale factor, detection threshold and minimum energy threshold.

Filter Coefficients

The filter coefficients have to be generated by a design tool based on the cascade structure shown in Figure 1 (filter order N=4). The next step consists of quantizing the filter coefficients to obtain Q14 format. This means that all coefficients have to be multiplied by 2^{14}. In addition to that, all coefficients a_{i1}, a_{i2} have to be multiplied by –1. Then these values have to be defined in the file **Filters.h**. The steps to generate the quantized filter coefficients are illustrated in . Each filter contains 14 elements, four filter delays and ten filter coefficients. Initialization is carried out as described previously.

Scale Factor

The scale factor specifies the shift that is applied to the input sample before bandpass filtering. This shift has to be set in the file **Filters.h** by means of a constant called **Name**Scale. The scale factor may be set to a value of 16 or less. *16-Scale Factor* specifies the right shift applied to the input sample before the frequency filtering stage. Possible values for the scale factor and the corresponding right shift are given in Table 2.

Table 2. Scale Factors and Corresponding Right Shift of the Input Sample

Scale Factor	16	15	14	13	12	11	10	9	8	7	6	5	4
right shift	0	1	2	3	4	5	6	7	8	9	10	11	12

Common values for the scale factor are 14, 15 or 16, depending on the amplification of the input stage. It is important not to saturate the frequency or exponential filters.

Detection Threshold

The detection threshold specifies the minimum amount of energy that has to be present in the passband of the corresponding filter in comparison to the global energy. In other words the inband energy must be greater than a specified fraction of the global energy, typically:

$$InbandEnergy \geq 0.5 \times GlobalEnergy \qquad \text{(equation 8)}$$

where the energy fraction equals 0.5. The comparison carried out after the exponential filters is given by equation 6. The minimum fraction of energy for detection is then given by 16 divided by the detection threshold. Different values for the detection threshold and the corresponding energy fraction are given in Table 3.

Table 3. Detection Thresholds and Corresponding Energy Fraction

Detection Threshold	16	24	32	40	48
Energy Fraction	1	$^{2}/_{3}$	½	$^{2}/_{5}$	$^{1}/_{3}$

The smaller the energy fraction required for detection the larger the passband of the corresponding filter. A common value for the detection threshold is 32.

Minimum Energy Threshold

The minimum energy threshold specifies the minimum absolute signal level that may be detected. The minimum signal level typically takes values between –43 dBm and –48 dBm. The minimum energy threshold is hardware dependent, as the signal level at the input of the A/D converter is determined by an analog amplification stage. Consequently it has to be determined experimentally. In order to set the minimum energy threshold, the variable **TotOut** has to be monitored while injecting a signal at the input. **TotOut** contains the output of the exponential filter for the global energy that will be compared to the minimum energy threshold during the decision stage. So if a continuous signal of the minimum signal level that shall be detected is injected at the input, **TotOut** will take the value that equals the minimum energy threshold. The value obtained in this way can then be set in **Filters.h** and copied to the variable **MinEng** during the initialization phase.

Interface Between High Level Programs and the Filter Program

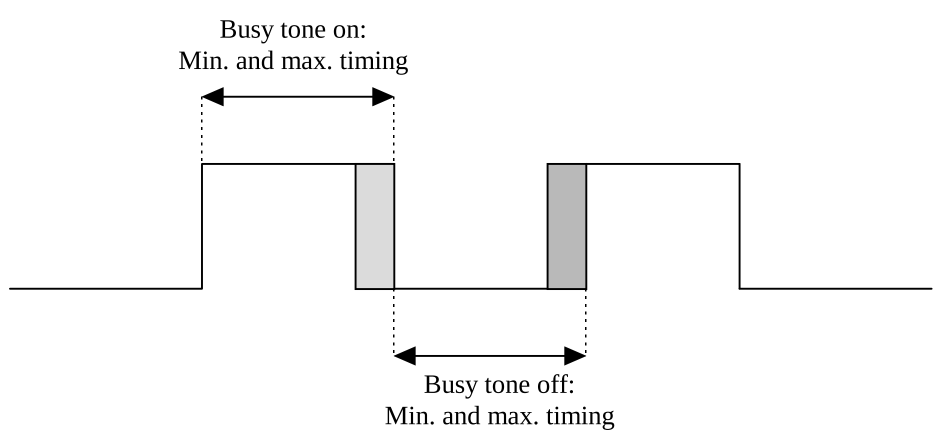
In this section the interface between a C program executing the main task and the filter program executed in the sample interrupt will be described.

The detection result of the tone detection procedure can be used by high level application S/W in order to carry out a timing check. For this purpose two variables are needed, which have to be referenced as external variables in the C program: **CptdFilter** and **Tim0**. **CptdFilter** contains the bit mask of the corresponding filter in case of detection (Table 3) and zero in case of no detection. **Tim0** is a timer that is incremented in the sample interrupt. The maximum value is 7fff hex, which corresponds to 4 seconds at 8 kHz.

All filters used in the program have to be initialized with the parameter values specified in the file **Filters.h**. This is done by the function Init---() that can be found in the file **initFilt.c** (where --- stands for the filter name). The routine InitTot() has to be called in order to initialize the exponential filter for global energy estimation.

An example of a C program which carries out the cadence check of the busy tone is given in Figure 5. The program main() calls a function for dialing which may include dial tone detection. After that the routine **CadenceCheck** is called, which checks the presence of a tone in the passband of the filter *Dial* (bit mask set in **CptdFilter**) and then carries out a timing check concerning the on/off sequence of the signal as shown in Figure 5:

Figure 5. Cadence Check for Busy Tone Detection



Tolerance concerning the timing is taken into account in the program by the constant values **BusyMin** and **BusyMax** which correspond to nominal timing –x% and nominal timing +x%, respectively.

The function **CadenceCheck** indicates busy tone detection by a return value of 1.

Processor Resources Used By Filter Programs

Table 4 summarizes the memory occupation (RAM and ROM) as well as computational load (MIPS) utilized by the filter functions contained in the file **biquad.asm**.

Table 4. Processor Resources Required for the Tone Detection Module

RAM	ROM	MIPS
80 words	500 words (biquad.asm) 300 words (initFilt.c)	4.5 (at 9.6 kHz) 3.8 (at 8 kHz)

RAM space is reserved for all filter variables in the file **biquad.asm**. The section containing these variables is called **Filter**. In the linker command file this section has to be put in a RAM block so as to be contained within one memory page (128 words).

In combination with the V22bis modem on the TMS320C2xx all filters are executed at a sampling rate of 8 kHz.

Reference

DFDP3/plus Digital Filter Design Package Instruction Manual; Atlanta Signal Processing Inc., 1991

Appendix A. Source Code

FILE: BIQUAD.ASM

```
****************************************
** File:     BIQUAD.ASM              **
**                                   **
** Author:   Katrin Matthes          **
**                                   **
** Description:                      **
** Implementation of programmable    **
** double biquad filter with         **
** detection stage (exponential      **
** filters)                          **
****************************************

NUMFILTER    .set   2; example of implementation:
                ; Dial/Busy tone and fax tone detector
             .def   CPTD
             .def   TMP
             .def   _CptdFilter
             .ref   FromAD,ForDA
             .def   _TotIn,   _TotOut
             .def   _MinEng
             .def   _FiltFunc
             .def   _InitFiltFunc
         .def   _DialShift,_DialIn,_DialOut,_DialThreshold,_DialFilter

             .if    NUMFILTER >=2
             .def _FaxShift,_FaxIn,_FaxOut,_FaxThreshold,_FaxFilter
```

```
            .elseif   NUMFILTER >=3
    .def  _Filt3Shift,_Filt3In,_Filt3Out,_Filt3Threshold,_Filt3Filter
            .elseif   NUMFILTER >=4
    .def  _Filt4Shift,_Filt4In,_Filt4Out,_Filt4Threshold,_Filt4Filter
            .endif

            .mmregs

;--------------------------------
;
; BIQUAD
; INDEXED
; y(n)=B0x(n)+d1(n-1)
; d1(n)=B1x(n)-A1y(n)+d2(n-1)
; d2(n)=B2x(n)-A2y(n)
;
; INPUT:
;           TMP    contains scaledinput sample
;           ARP    -> AR1      AR0 -> DNM1
;           AR1    -> B0       PM=1 (<<1)
;           SSXM
; OUTPUT
;           ARP    -> AR1      AR0 -> DNM1
;           AR1    -> B0
; MODIFIED
;           AR0, AR1
;
; 42 cycles
;----------------------------------
; DATA ORGANIZATION:
;D1NM1      .BSS  ; AR0
;D2NM1      .BSS
;B0         .BSS  ; AR1
;B1         .BSS
;A1         .BSS
;A2         .BSS
;B2         .BSS
BIQUAD
; all filter coefficients Q14
;*
;* SECOND-ORDER FILTER SECTION
;*
            ldp    #TMP
            LT     TMP           ;GET SCALED  INPUT
            MPY    *+,ar0        ;P = B0* INPUT
            lac    *+,15,ar1     ;AC= Z-1
            MPYA   *+,ar0        ;AC= Z-1 +   (B0* INPUT)
;                                ;P = B1   * INPUT
            ldp    #Output
            SACH   Output,1      ;Save in OUTPUT
            LTP    Output        ;AC= B1 * INPUT
            ADD    *-,15,ar1     ;AC= Z-2 +   (B1* INPUT)
            MPY    *+            ;P = A1* OUTPUT
            APAC
            MPY    *+,ar0        ;AC= Z-2 +   (B1*INPUT)+(A1*OUTPUT)
;                                ;P = A2   * OUTPUT
            SACH   *+,1,ar1      ;Save in Z-1
            Ldp    #TMP
```

```
            LTP    TMP       ;AC= A2 * OUTPUT
            MPY    *+,ar0    ;P = B2* INPUT
            APAC             ;AC= (B2 *INPUT)+(A2 * OUTPUT)
            SACH   *+,1,ar1  ;Save in Z-2

            Ldp    #Output
            lac    Output
            ldp    #TMP
            sacl   TMP

            LT     TMP             ;GET SCALED  INPUT
            MPY    *+,ar0          ;P = B0* INPUT
            Lac    *+,15,ar1       ;AC= Z-1
            MPYA   *+,ar0          ;AC= Z-1 +(B0* INPUT)
;                                  ;P = B1* INPUT
            ldp    #Output
            SACH   Output,1        ;Save in OUTPUT
            LTP    Output          ;AC= B1 * INPUT
            ADD    *-,15,ar1       ;AC= Z-2 +(B1* INPUT)
            MPY    *+              ;P = A1   * OUTPUT
            APAC
            MPY    *+,ar0          ;AC= Z-2 +(B1*INPUT)+(A1*OUTPUT)
;                                  ;P = A2   * OUTPUT
            SACH   *+,1,ar1        ;Save in Z-1
            ldp    #TMP
            LTP    TMP             ;AC= A2 * OUTPUT
            MPY    *+,ar0          ;P = B2* INPUT
            APAC                   ;AC= (B2 *INPUT) + (A2 * OUTPUT)
            SACH   *+,1,ar1        ;Save in Z-2
            ret


;------------------
;
; macro for a filter
; 25 cycles
;
;-----------------
Filter      .macro Name
            .NEWBLOCK
            ldp    #_:Name:Shift
            lt     _:Name:Shift
            LDPK   #FromAD
            LACT   FromAD       ;load input sample with specified
                                ;shift
            LDPK   #TMP
; MULTIPLY INPUT BY GAIN
            SACH   TMP

; SET POINTERS
            LAR    AR0,#_:Name:Filter
            spm    1
            MAR    *,AR0              ; AR0 -> d1(n-1)
            LARK   AR1,#4
            MAR    *,AR1
            MAR    *0+,AR1            ; AR1 -> B0

            CALL   BIQUAD
```

```
                ldp    #Output
                lac    Output    ; accumulate absolute value
                                 ; of 5 input samples
                abs
;               ldp    #_:Name:In
                add    _:Name:In
                sacl   _:Name:In
                sub    #MaxVal
                blz    $1
                lac    #MaxVal
                sacl   _:Name:In
$1
                .endm


;------------------
;
; CPTD FILTER
; 85 cycles per filter + 40 cycles
; = 4 * 85 + 40 (max) = 380
;------------------
CPTD
                SPM    1
                sovm
; DIAL TONE
                ldp    #FromAD
                lac    FromAD       ; accumulate absolute value
                                    ; of 5 input samples
                abs
                ldp    #_TotIn
                add    _TotIn
                sacl   _TotIn
                sub    #MaxVal
                blz    no_clip
                lac    #MaxVal
                sacl   _TotIn
no_clip
                Filter   Dial
                .if      NUMFILTER >= 2
                Filter   Fax
                .elseif  NUMFILTER >= 3
                Filter   Filt4
                .elseif  NUMFILTER >= 4
                Filter   Filt3
                .endif

                ldp      #_FiltFunc
                lac      _FiltFunc
                cala

                rovm
                ret

;-----------------
; Inits      FiltFunc
; called by  InitDial()
;-----------------
```

```
_InitFiltFunc
             ldp    #_FiltFunc
             lac    #TotExp
             sacl   _FiltFunc
             zac
             sacl   TotMax
             sacl   Detect
             ret


;-----------------------------------------------------
; Macro for calculation of the exponential filter based
; on the sum of 5 ABS(input sample)
; Comparison of the global output to the filtered (biquad)
; output
; if (global <=factor * filtered) then DETECTION
;
; This comparison is not carried out if
; 1) the minimum global  energy is below the threshold
;            MinEng
; 2) the energy after filtering is below the noise
;            threshold of the filter
; 3) the global energy decreases to 0.5* Max, indicating a
;            transition on/off
;
; 40 cycles
;-----------------------------------------------------
TestOut      .macro   Name
             .newblock
             ldp    #_:Name:Out
             zalr   _:Name:Out         ; ROUNDING
             add    _:Name:In,16-6     ; 1/64
             SUB    _:Name:Out,16-6
             SACH   _:Name:Out

             Lac    _TotOut            ; check   min. thres.
             Sub    _MinEng
             blez   $4

             lac    _TotOut,1       ; TotOut > 0.5*TotMax ?
             sub    TotMax          ; detection: ON/OFF   transition
             blez   $3
             lac    _:Name:Out      ; noise due to filter
             sub    #:Name:Min
             blez   $4

             lac    _TotOut,4
; Totout<<4- (factor *   FilterOut)<<1
             lt     _:Name:Threshold   ; Detection test
             mpy    _:Name:Out
             spac
             BLZ    $1
; UNDER      THRESHOLD
$3
             lac    Detect
             and    #:Name:Mask
             bz     $4
             lar    AR0,#_:Name:Filter
```

```
             mar    *,ar0
             zac
             sacl   *+       ; clear  D11(N-1)
             sacl   *+       ; clear  D12(N-1)
             sacl   *+       ; clear  D21(N-1)
             sacl   *        ; clear  D22(N-1)
             sacl   _:Name:Out
             sacl   TotMax
$4
             LALK   #~:Name:Mask
             and    Detect
             sacl   Detect
             ldp    #_CptdFilter
             AND    _CptdFilter
             B      $2
; OVER THRESHOLD
$1
             lac    _TotOut
             sub    TotMax
             blez   $5
             lac    _TotOut  ; look for maximum
             sacl   TotMax
$5
             LALK   #:Name:Mask
             or     Detect
             sacl   Detect
             OR     _CptdFilter
$2
             SACL   _CptdFilter
             zac
             sacl   _:Name:In

             .endm

;-------------------
; exponential filter for global signal
;-------------------
TotExp
             lac    #DialExp
             sacl   _FiltFunc
             zalr   _TotOut          ; ROUNDING
             ldp    #_TotIn
             add    _TotIn,16-6      ; 1/64
             LDPK   #_TotOut
             SUB    _TotOut,16-6
             SACH   _TotOut
             Ldp    #_TotIn
             zac
             sacl   _TotIn
             ret

;-------------------
; exponential filter for signal after Dialfilter
;-------------------
DialExp
             .if    NUMFILTER >= 2
             lac    #FaxExp
             .else
```

```
              lac    #TotExp
              .endif
              sacl   _FiltFunc
              TestOut   Dial
              ret

;-------------------
; exponential filter for signal after Faxfilter
;-------------------
              .if    NUMFILTER >=2
FaxExp
              .if    NUMFILTER >= 3
              lac    #Filt3Exp
              .else
              lac    #TotExp
              .endif
              sacl   _FiltFunc
              TestOut   Fax
              ret
              .endif

;-------------------
; exponential filter for signal after Filt3 filter
;-------------------
              .if    NUMFILTER >=3
Filt3Exp
              .if    NUMFILTER >=4
              lac    #Filt4Exp
              .else
              lac    #TotExp
              .endif
              sacl   _FiltFunc
              TestOut   Filt3
              ret
              .endif


;-------------------
; exponential filter for signal after Filt4 filter
;-------------------
              .if    NUMFILTER >=4
Filt4Exp
              lac    #TotExp
              sacl   _FiltFunc
              TestOut   Filt4
              ret
              .endif


_CptdFilter  .usect    "Filter",1
Output       .usect    "Filter",1
_TotIn       .usect    "Filter",1
_TotOut      .usect    "Filter",1
_DialFilter  .usect    "Filter",14
_DialIn      .usect    "Filter",1
_DialOut     .usect    "Filter",1
_DialShift   .usect    "Filter",1
_DialThreshold  .usect       "Filter",1
```

```
             .if    NUMFILTER >=2
_FaxFilter      .usect       "Filter",14
_FaxIn          .usect       "Filter",1
_FaxOut         .usect       "Filter",1
_FaxShift       .usect       "Filter",1
_FaxThreshold   .usect       "Filter",1
             .elseif   NUMFILTER >=3
_Filt3Filter    .usect       "Filter",14
_Filt3In        .usect       "Filter",1
_Filt3Out       .usect       "Filter",1
_Filt3Shift     .usect       "Filter",1
_Filt3Threshold .usect       "Filter",1
             .elseif   NUMFILTER >=4
_Filt4Filter    .usect       "Filter",14
_Filt4In        .usect       "Filter",1
_Filt4Out       .usect       "Filter",1
_Filt4Shift     .usect       "Filter",1
_Filt4Threshold .usect        "Filter",1
             .endif
_MinEng         .usect       "Filter",1
_FiltFunc       .usect       "Filter",1
Detect          .usect       "Filter",1
TotMax          .usect       "Filter",1
TMP             .usect       "Filter",1

MaxVal       .set   7fffh
DialMin      .set   90h
FaxMin       .set   5ah
Filt3Min     .set   55h
Filt4Min     .set   55h
DialMask     .set   0001h
FaxMask      .set   0002h
Filt3Mask    .set   0004h
Filt4Mask    .set   0008h
```

FILE: INITFILT.C

```
/**********************************************************/
/* File: INITFILT.C                                      */
/*                                                       */
/* Author: Katrin Matthes                                */
/*                                                       */
/*Routine initializes all CPTD Filters for double biquad */
/* Memory Organization:                                  */
/*         NameFilter:                                   */
/*            D11(N-1)                                   */
/*            D12(N-1)                                   */
/*            D21(N-1)                                   */
/*            D22(N-1)                                   */
/*            B10                                        */
/*            B11                                        */
/*            A11                                        */
/*            A12                                        */
/*            B12                                        */
/*            B20                                        */
/*            B21                                        */
```

```
/*              B22                                        */
/*              A21                                        */
/*              A22                                        */
/**********************************************************/

#include "Filters.h"

extern int DialFilter[14];
extern int DialThreshold;
extern int DialIn, DialOut, TotIn, TotOut, DialShift;
#if NUMFILTER >=2
extern int FaxFilter[14];
extern int FaxThreshold;
extern int FaxIn, FaxOut, FaxShift;
#elif NUMFILTER >=3
extern int Filt3Filter[14];
extern int Filt3Threshold;
extern int Filt3In, Filt3Out, Filt3Shift;
#elif NUMFILTER >=4
extern int Filt4Filter[14];
extern int Filt4Threshold;
extern int Filt4In, Filt4Out, Filt4Shift;
#endif

extern int MinEng;
int InitFiltFunc(void);


void InitTot(void)
{
            MinEng=MinThres;
            TotIn=0;
            TotOut=0;
            InitFiltFunc();
}


void InitDial(void)
{
            DialFilter[0]=0;
            DialFilter[1]=0;
            DialFilter[2]=0;
            DialFilter[3]=0;
            DialFilter[4]=Dial1_B0;
            DialFilter[5]=Dial1_B1;
            DialFilter[6]=Dial1_A1;
            DialFilter[7]=Dial1_A2;
            DialFilter[8]=Dial1_B2;
            DialFilter[9]=Dial2_B0;
            DialFilter[10]=Dial2_B1;
            DialFilter[11]=Dial2_A1;
            DialFilter[12]=Dial2_A2;
            DialFilter[13]=Dial2_B2;

            DialThreshold=DialThres;
            DialIn=0;
            DialOut=0;
            DialShift=DialScale;
```

```
}

#if NUMFILTER >= 2

void InitFax (void)
{
            FaxFilter[0]=0;
            FaxFilter[1]=0;
            FaxFilter[2]=0;
            FaxFilter[3]=0;
            FaxFilter[4]=Fax1_B0;
            FaxFilter[5]=Fax1_B1;
            FaxFilter[6]=Fax1_A1;
            FaxFilter[7]=Fax1_A2;
            FaxFilter[8]=Fax1_B2;
            FaxFilter[9]=Fax2_B0;
            FaxFilter[10]=Fax2_B1;
            FaxFilter[11]=Fax2_A1;
            FaxFilter[12]=Fax2_A2;
            FaxFilter[13]=Fax2_B2;

            FaxThreshold=FaxThres;
            FaxIn=0;
            FaxOut=0;
            FaxShift=FaxScale;
}

#elif NUMFILTER >= 3

void InitFilt3 (void)
{
            Filt3Filter[0]=0;
            Filt3Filter[1]=0;
            Filt3Filter[2]=0;
            Filt3Filter[3]=0;
            Filt3Filter[4]=Filt3_1_B0;
            Filt3Filter[5]=Filt3_1_B1;
            Filt3Filter[6]=Filt3_1_A1;
            Filt3Filter[7]=Filt3_1_A2;
            Filt3Filter[8]=Filt3_1_B2;
            Filt3Filter[9]=Filt3_2_B0;
            Filt3Filter[10]=Filt3_2_B1;
            Filt3Filter[11]=Filt3_2_A1;
            Filt3Filter[12]=Filt3_2_A2;
            Filt3Filter[13]=Filt3_2_B2;

            Filt3Threshold=Filt3Thres;
            Filt3In=0;
            Filt3Out=0;
            Filt3Shift=Filt3Scale;
}

#elif NUMFILTER >= 4

void InitFilt4 (void)
{
            Filt4Filter[0]=0;
            Filt4Filter[1]=0;
```

```
            Filt4Filter[2]=0;
            Filt4Filter[3]=0;
            Filt4Filter[4]=Filt4_1_B0;
            Filt4Filter[5]=Filt4_1_B1;
            Filt4Filter[6]=Filt4_1_A1;
            Filt4Filter[7]=Filt4_1_A2;
            Filt4Filter[8]=Filt4_1_B2;
            Filt4Filter[9]=Filt4_2_B0;
            Filt4Filter[10]=Filt4_2_B1;
            Filt4Filter[11]=Filt4_2_A1;
            Filt4Filter[12]=Filt4_2_A2;
            Filt4Filter[13]=Filt4_2_B2;

            Filt4Threshold=Filt4Thres;
            Filt4In=0;
            Filt4Out=0;
            Filt4Shift=Filt4Scale;
}

#endif
```

FILE: FILTERS.H

```
/**************************************/
/* File:     FILTERS.H                */
/* Author:   Katrin Matthes           */
/*                                    */
/* Include file containing            */
/* filter coefficients for            */
/* double biquad filters              */
/**************************************/

#define NUMFILTER  2
/* define number of filters executed in parallel */

/* Dial Filter*/
/* All coefficients Q14 */
/* Coefficients for 1st biquad */
#define      Dial1_B0        539
#define      Dial1_B1        -914
#define      Dial1_B2        539
#define      Dial1_A1        30507
#define      Dial1_A2        -16092
/* Coefficients for 2nd biquad */
#define      Dial2_B0        4602
#define      Dial2_B1        -9029
#define      Dial2_B2        4602
#define      Dial2_A1        30881
#define      Dial2_A2        16118

#define      DialScale       15    /*input sample >> 1*/

/* threshold for dial tone detection */
#define      DialThres       0x28  /* 0x31   */


/* Fax Filter 1100 Hz */
```

```
/* Coefficients for 1st biquad */
#define     Fax1_B0        1209
#define     Fax1_B1        -999
#define     Fax1_B2        1209
#define     Fax1_A1        20049
#define     Fax1_A2        -15773
/* Coefficients for 2nd biquad */
#define     Fax2_B0        4616
#define     Fax2_B1        -7426
#define     Fax2_B2        4616
#define     Fax2_A1        21726
#define     Fax2_A2        -15806

#define     FaxScale       15       /* input sample >>1 */

/* threshold for answer tone detection */
#define     FaxThres       0x25     /*0x2a*/


/* Filt3 Filter xxx Hz */
/* Coefficients for 1st biquad */
#define     Filt3_1_B0     0
#define     Filt3_1_B1     0
#define     Filt3_1_B2     0
#define     Filt3_1_A1     0
#define     Filt3_1_A2     0
/* Coefficients for 2nd biquad */
#define     Filt3_2_B0     0
#define     Filt3_2_B1     0
#define     Filt3_2_B2     0
#define     Filt3_2_A1     0
#define     Filt3_2_A2     0

#define     Filt3Scale     15       /* input sample >>1 */

/* threshold for Filt3  detection */
#define     Filt3Thres   0x18  /*0x38*/


/* Filt4 Filter xxx Hz */
/* Coefficients for 1st biquad */
#define     Filt4_1_B0     0
#define     Filt4_1_B1     0
#define     Filt4_1_B2     0
#define     Filt4_1_A1     0
#define     Filt4_1_A2     0
/* Coefficients for 2nd biquad */
#define     Filt4_2_B0     0
#define     Filt4_2_B1     0
#define     Filt4_2_B2     0
#define     Filt4_2_A1     0
#define     Filt4_2_A2     0

#define     Filt4Scale     15       /* input sample >>1 */

/* threshold for Filt4  detection */
#define     Filt4Thres     0x20        /*0x38*/
```

```
#define      MinThres     0x110
/* minimum detection threshold */

/* min. and max. timing for cadence check: busy tone */
#define BusyMin 42
/* value * 10ms, BusyMax+20ms to account for filter delay*/
#define BusyMax    57

/* Masks for the different Filters*/
#define      DialMask     0x0001
#define      FaxMask      0x0002
#define      Filt3Mask    0x0004
#define      Filt4Mask    0x0008

/* initialization routines for the implemented filters */
void InitTot(void);
void InitDial(void);
#if NUMFILTER >=2
void InitFax(void);
#elif NUMFILTER >=3
void InitFilt3(void);
#elif NUMFILTER >=4
void InitFilt4(void);
#endif
```

Appendix B. Glossary

CPTD Call Progress Tone Detection

IIR Infinite Impulse Response

TI Contact Numbers

INTERNET

TI Semiconductor Home Page
www.ti.com/sc

TI Distributors
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas
Phone +1(972) 644-5580
Fax +1(972) 480-7800
Email sc-infomaster@ti.com

Europe, Middle East, and Africa
Phone
Deutsch +49-(0) 8161 80 3311
English +44-(0) 1604 66 3399
Español +34-(0) 90 23 54 0 28
Francais +33-(0) 1-30 70 11 64
Italiano +33-(0) 1-30 70 11 67
Fax +44-(0) 1604 66 33 34
Email epic@ti.com

Japan
Phone
International +81-3-3457-0972
Domestic 0120-81-0026
Fax
International +81-3-3457-1259
Domestic 0120-81-0036
Email pic-japan@ti.com

Asia
Phone
International +886-2-23786800
Domestic
Australia 1-800-881-011
TI Number -800-800-1450
China 10810
TI Number -800-800-1450
Hong Kong 800-96-1111
TI Number -800-800-1450
India 000-117
TI Number -800-800-1450
Indonesia 001-801-10
TI Number -800-800-1450
Korea 080-551-2804
Malaysia 1-800-800-011
TI Number -800-800-1450
New Zealand 000-911
TI Number -800-800-1450
Philippines 105-11
TI Number -800-800-1450
Singapore 800-0111-111
TI Number -800-800-1450
Taiwan 080-006800
Thailand 0019-991-1111
TI Number -800-800-1450
Fax 886-2-2378-6808
Email tiasia@ti.com