Texas Instruments

Improving 32 Channel DTMF Decoders Using the TMS320C5x

Application Report

1996 Digital Signal Processing Solutions

Printed in U.S.A.

Improving 32 Channel DTMF Decoders Using the TMS320C5x

Peter Duh
CAC FAE Taiwan

SPRA085
June 1996

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Title *Page*

List of Illustrations

ABSTRACT

PBX office switching systems use dual tone multifrequency (DTMF) signals since the signals serve as data entry terminals. DTMF uses a pair of frequencies to produce a tone. PBX systems use multiple DTMF chips to encode or decode the tones. PBX systems also process other functions such as voice compression or expansion and voice mail through their systems. Extending a PBX system to encompass these functions is expensive. DSPs increase the performance and flexibility of the PBX systems, while decreasing the cost. This application report discusses improvements to the multi-channel DTMF decoder using a TMS320C5x DSP1[1].

DTMF TONE ENCODING

DTMF is a touch-tone signal comprised of two frequencies. One frequency is from the high frequency group, (1209 Hz, 1336 Hz, 1477 Hz, 1633 Hz) and one is from the low frequency group (697 Hz, 770 Hz, 852 Hz, 941 Hz). Figure 1 shows DTMF tone generation.

Pressing 1 on a 16-key pad generates a DTMF signal pair. This signal pair is 1209 Hz from the high frequency group and 697 Hz from the low frequency group.

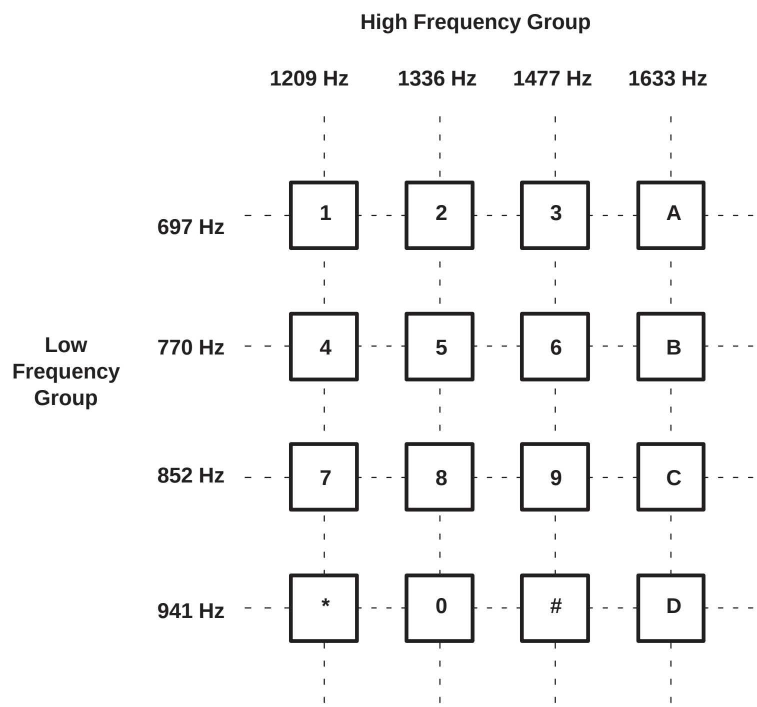


Figure 1. DTMF Touch-Tone Frequency

The DTMF encoder uses the two frequencies to calculate a value. The DTMF encoder equation is:

$$Y(n) = 2\cos(coef) \times Y(n-1) + (-1) \times Y(n-2) + [\sin(coef) \times X(n-1)] \quad (1)$$

where coef=360 × frequency/sample rate.

The encoder equation uses the following initial data:

$$Y(0) = 0;\ Y(1) = \sin(x);\ X(1) = 1;\ \text{and } X(n) = 0 \text{ for } X \neq 1.$$

The sample rate is 8 KHz.

DTMF TONE DECODING

DTMF tone decoding uses codes that are more complicated than DTMF encoding. The decoding program uses a discrete Fourier transform (DFT) algorithm, known as Goertzel's algorithm. The algorithm is a series of second-order infinite impulse response (IIR) filters. The DTMF decoder equation is:

$$Y(n) = X(n) + 2\cos(coef) \times Y(n-1) + (-1)Y(n-2) \qquad (2)$$

where coef = 360 × frequency/sample rate and X(n) is the current sample value. Y(n–1) and Y(n–2) are feedback storage elements for the frequency point, k. The value of k is:

$$k = N \times Freq/(Sample\ Rate) \qquad (3)$$

The decoder equation uses the initial data:

$$Y(-1) = 0,\ Y(-2)=0,\ Y(-3)=0,\ \ldots\ \text{for } n = 0, 1, 2, \ldots, n-1.$$

The sample rate is 8 KHz.

Figure 2 shows a diagram of a 16-tone DTMF decoder. Input data decodes linearly and a code compresses several phone signals. A compressed pulse coded modulation† (PCM) signal goes through the log-to-linear expansion, then is put into a stream form. Once it is in a stream form, the samples go to the DFT algorithm.

The samples are put into a parallel array containing the DFT algorithms. They are processed serially. There are sixteen algorithm for the tone detection of the DTMF frequencies and second harmonics. These second harmonics distinguish between speech and DTMF tones.

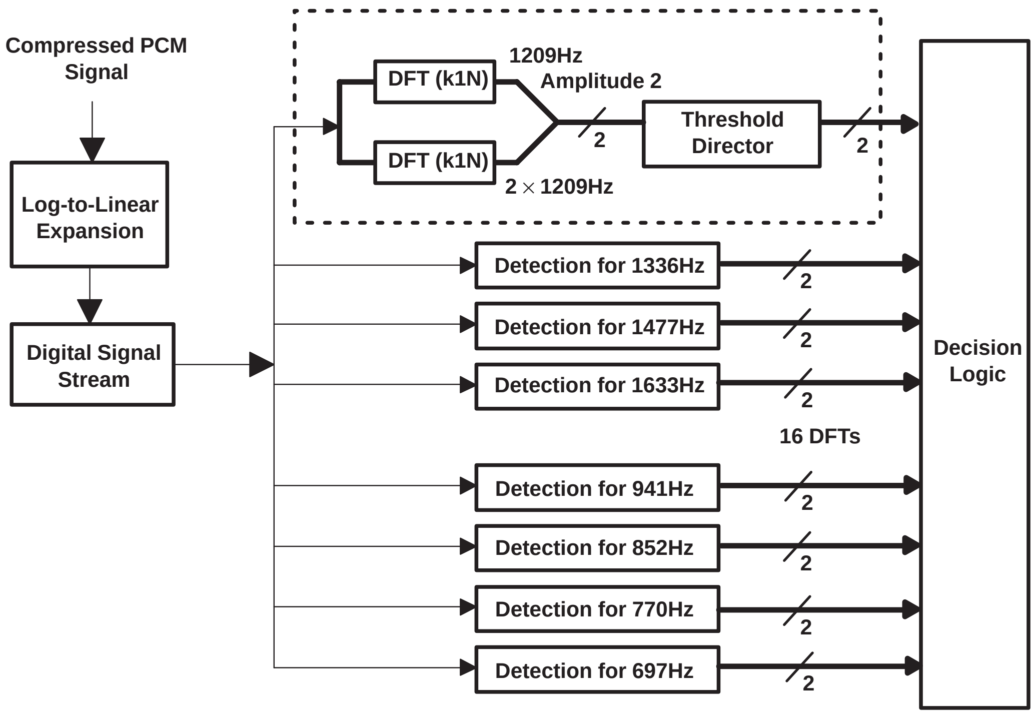


Figure 2. Tone Decoder — 16 DFT Transforms

† PCM is the most common form of digital waveform coding.

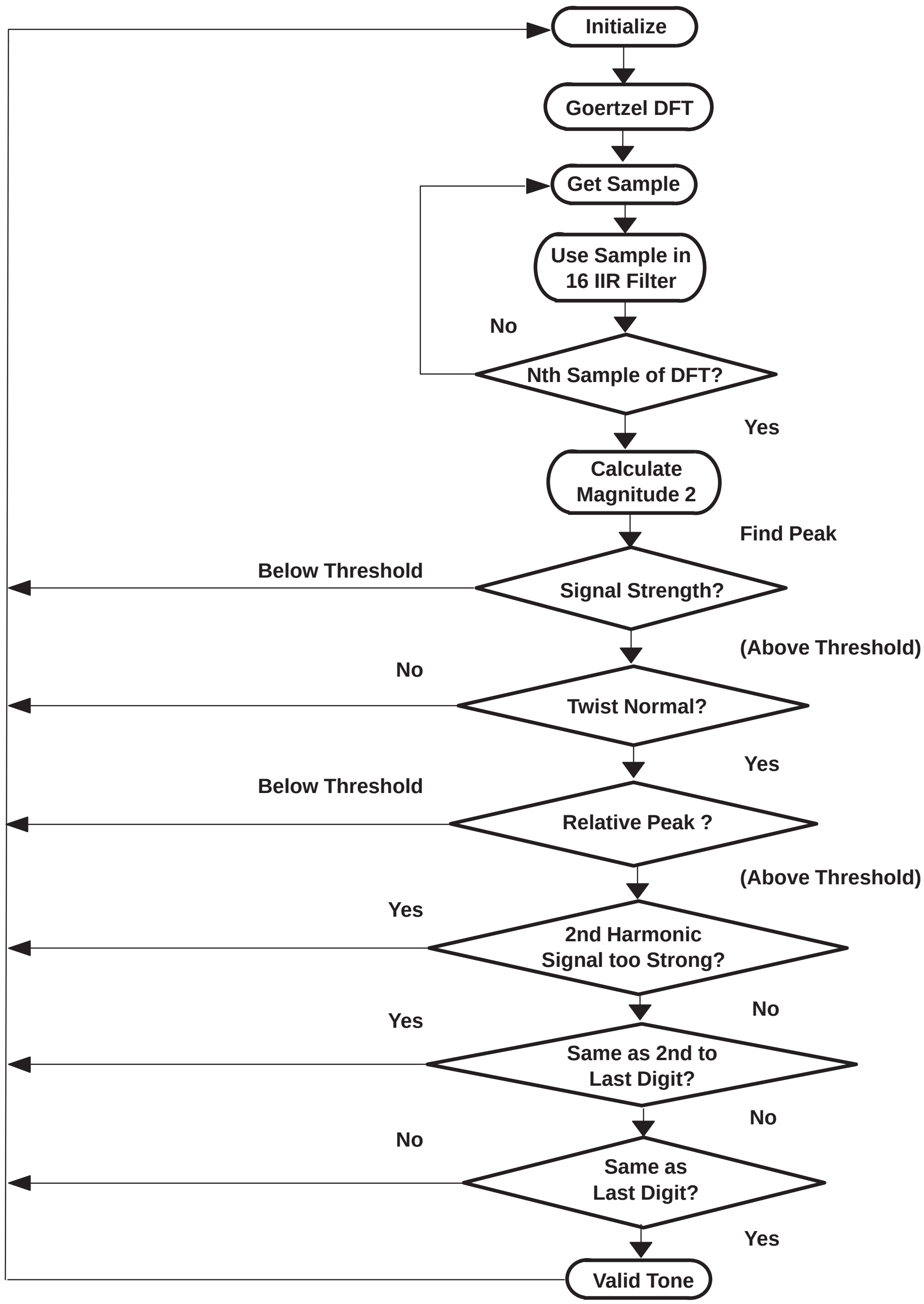


Figure 3. DTMF Tone Detection

Figure 3 shows the flow of data for DTMF tone detection. After processor initialization, the program sends the first sample to the 16 DFT IIR filters. The program compares the sample data with the thresholds to detect valid DTMF tones. A complete description of the tone decoder and DTMF tone detection is in the *TI DSP Application Book Volume 1*.

PERFORMANCE CONSIDERATION

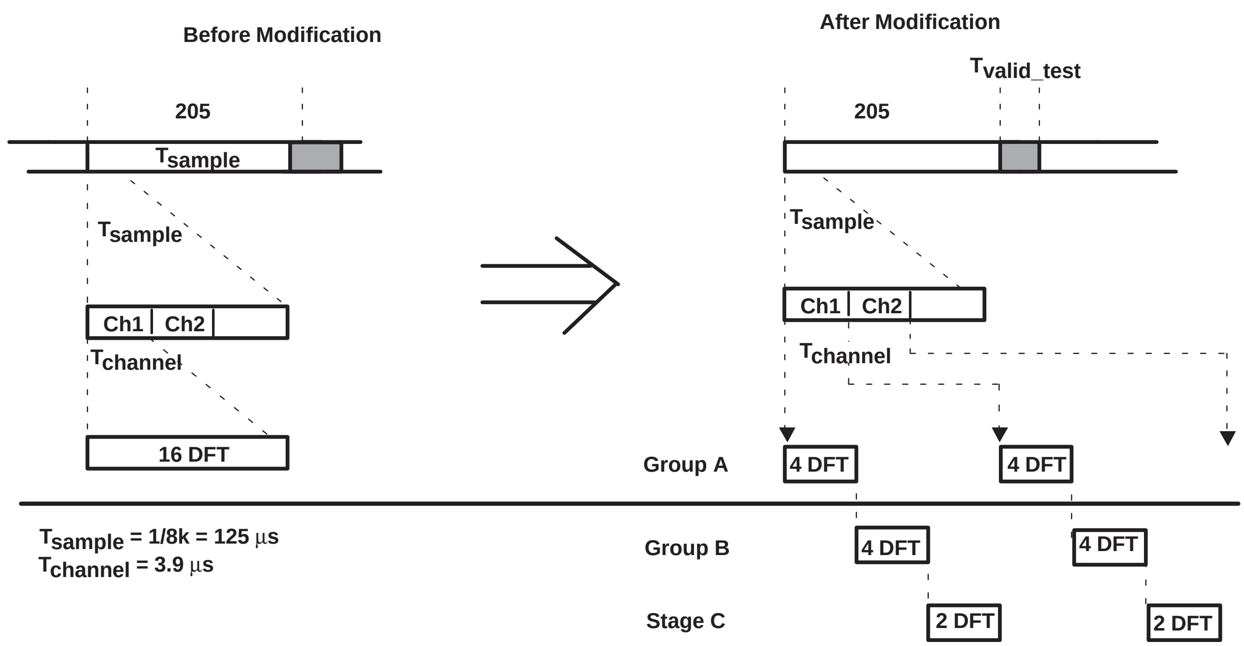
Before the application uses the ‘C5x, the device performance is evaluated to determine if it can process 32 channels simultaneously. Figure 4 shows multi-channel operation. The calculations use the input data stream to determine T_{sample}, $T_{channel}$, and N. T_{sample} is the time between interrupts or the sample period. $T_{channel}$ is the time to calculate the sample within a channel, and N is the number of possible channels. The equations for T_{sample} and N are:

$$T_{sample} = 1/8\ \text{Khz} \tag{4}$$

$$N = T_{sample}/T_{channel} \tag{5}$$

For example, if T_{sample} = 125 µs and N = 32, then solving for $T_{channel}$ = 3.9 µs. The ‘C5x takes 156 cycles with an instruction rate of 50 MIPS (20 ns).

These calculations show that the ‘C5x allows each channel 3.9 s to execute its frequency detection program. During this interval, the ‘C5x performs 16 DFTs and validates the signal. Since the time period is so small, this poses a performance limitation on the system. A solution to the performance problem is multi-processors. Multi-processors increase the product cost and work against the goal for an inexpensive telecommunications product.



Note 1: It takes approximately 195 cycles running at 100 MHz DSP (20 ns) to process 16 DFTs

Note 2: 16 DFTs include:

697,	697 × 2
770,	770 × 2
852,	852 × 2
941,	941 × 2
1209,	1209 × 2
1336,	1336 × 2
1477,	1477 × 2
1633,	1633 × 2

Note 1: It takes approximately 195 cycles running at 100 MHz DSP (20 ns) to process 10 DFTs or 4 DFTs if signal does not pass Group A.

Group A: 4 DFTs include:

1209
1336
1477
1633

Group B: 4 DFTs include:

697
770
852
941

Figure 4. Multi-Channel Decoding Timing

IMPROVEMENTS TO DTMF ENCODING

Three methods improve the performance of multi-channel DTMF decoders: optimizing the receive interrupt routine, decreasing the number of DFTs, and using a buffer to smooth processing time. Figure 5 shows a modified tone decoder and Figure 6 outlines the program flow for the decoder. Applying all three methods using the modified program flow of Figure 6 improves the performance of DTMF decoders.

Optimizing the Receive Interrupt Routine

The receive interrupt routine processes two channels in a single interrupt. This combination of channel processing saves one interrupt latency period. The input data uses a μ-law-to-linear-conversion look-up table†. This table allows the DSP to get a real value for the input sample of the input data.

The routine scales the input sample to eight bits. Two channels of input are read from the 16-bit data receive register (DRR); one from the high byte and one from the low byte.

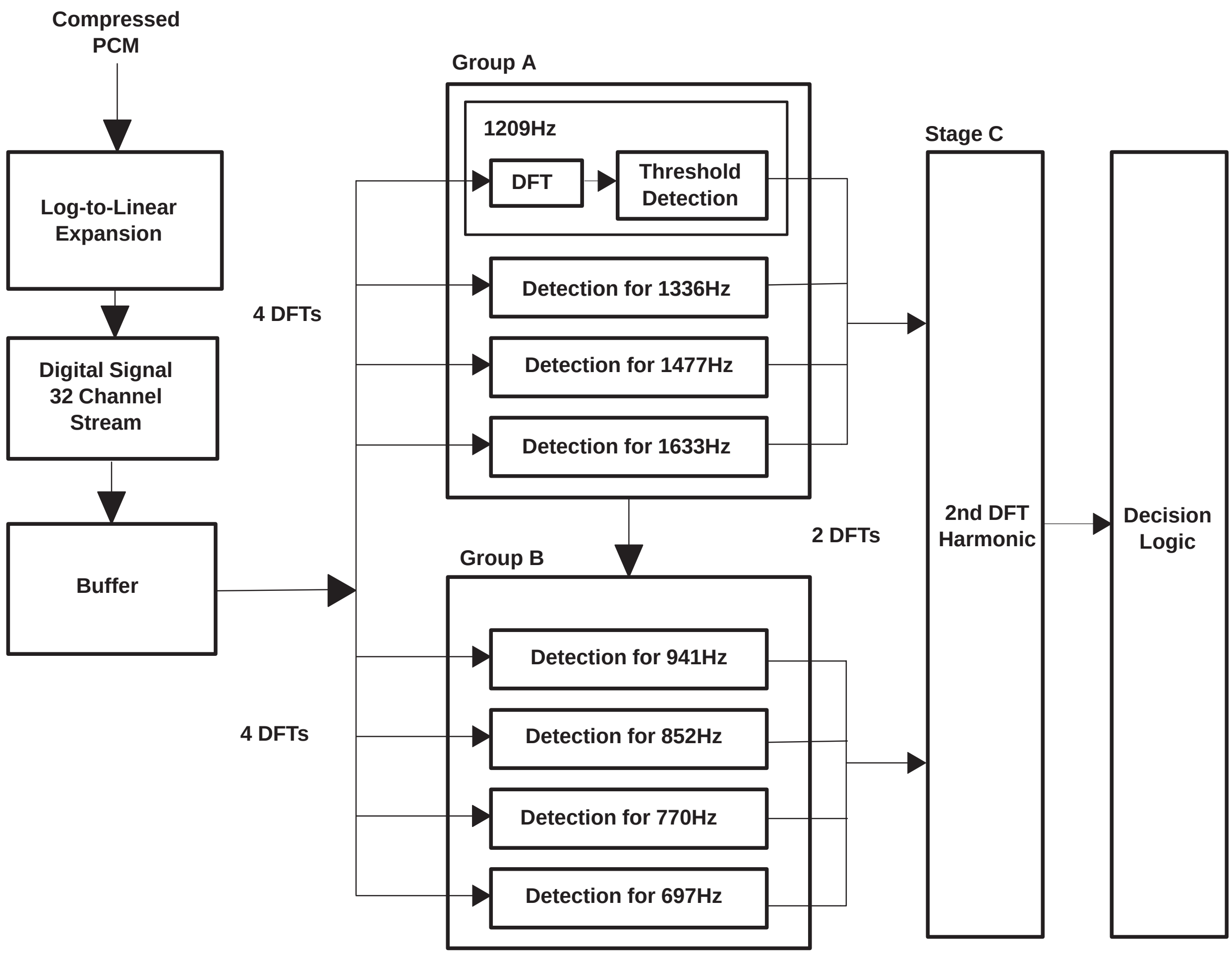


Figure 5. Modified DTMF Tone Decoder

† μ_law is a companding scheme used in public telephone networks of the US.

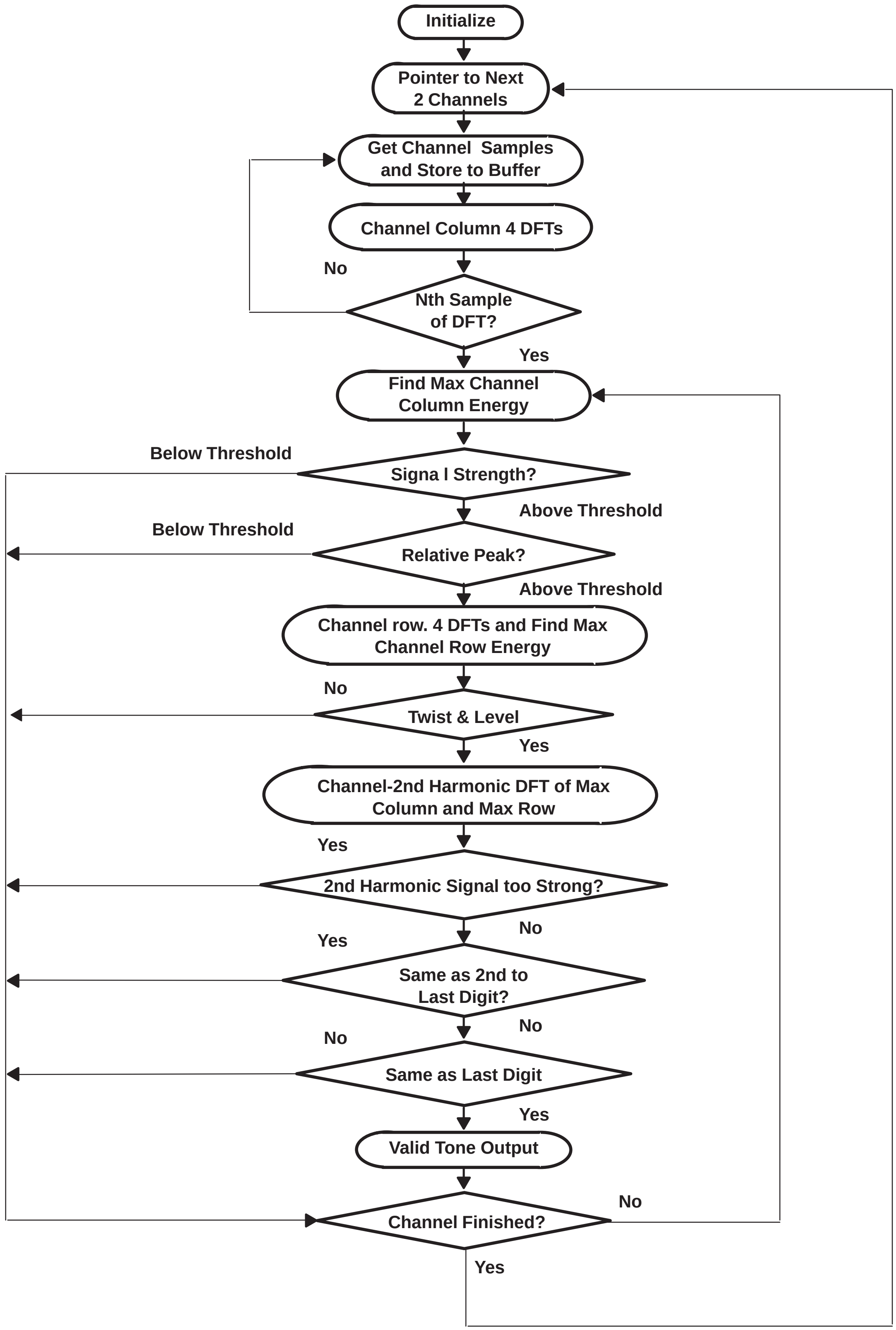


Figure 6. Modified DTMF Tone Detection

Decreasing the Number of DFTs

The modified tone decoder of Figure 5 shows a process reduction from 16 DFTs to 10 DFTs. This reduction saves processing time in multi-channel operation.

The 10 DFTs are separated into group A, group B and stage C. Group A detects the 4 DFTs belonging to the high frequencies column shown in Figure 1. Group B processes the low frequencies row. Stage C detects the second harmonics and rejects speech from the incoming DTMF tones.

Signal samples go through group A, group B, and stage C (in that order) to detect valid tones. Figure 4 shows the modified processing method.

Between group A and group B processing, the program tests for signal strength and relative peak. The processor squares each DTF value to calculate the magnitude of the signal strength and the maximum value of the four DFTs.

Between group B and stage C processing, the program tests for the twist ratio of the row to the column tone amplitude. The ratio ranges between values with a twist greater than 4 dB and a reverse twist greater than 8 dB†.

Using a Buffer

If all 32 channels are in use the above method of DTF processing does not save time. The method also does not ensure that all 32 channels are processed without data loss. A buffer solves this problem. The buffer stores the input stream that feeds the DTF processing. Since each channel is not constantly in use, the buffer provides the DSP with a greater tolerance for DTF processing and smooths task flow. Figure 7 shows the normalization for multi-channel DTMF.

The buffer size is based upon an AT&T specification that states that the maximum data rate for touch-tone signals is 100 ms/digit. A sample-store interval takes approximately 25 ms. The buffer uses four levels to store the input data during a 100 ms interval. Equation 6 calculates buffer size:

$$\text{Buffer Size} = N\ 4\ \text{levels} = 32\ \text{K words} \quad (6)$$

where the number of samples, N, is 200-256.

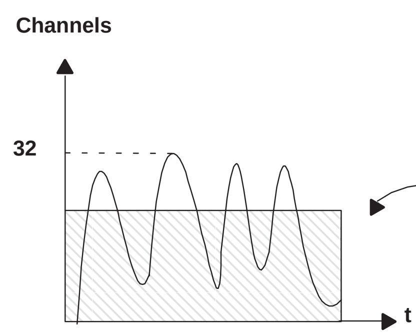


Figure 7. Normalization of the Distribution for Multi-Channel DTMF

† AT&T recommends more than 4 dB of twist or 8 dB of reverse twist.

SUMMARY

The DFT algorithm is compact and needs approximately 200 time-ordered sample lengths to calculate magnitude. The DTMF decoder source code is in Appendix A. The code uses the 10 DFT optimization and buffering to provide 32-channel DTMF processing.

The code in Appendix A needs modification to improve the performance of the DTMF processing. The receive interrupt routine is capable of processing two channels versus the one it now handles. Optimization of the DFTs reduces processing time. A buffer increases performance. All three of these modifications work toward 32-channel DTMF

All of the improvements achieve the goal processing 32 channels in a single TMS320C5x DSP. Other DSP functions, such as DTMF encoding, voice compression/expansion, and voice mail require either a multi-processor or a high performance DSP.

REFERENCES

1. Texas Instruments, Digital Signal Processing Applications with the TMS320 Family, Theory, Algorithms, and Implementations Volume 1, Topic 19, “Add DTMF Generation and Decoding to DSP–μP Design” (1989).

APPENDIX A—DTMF DECODE PROGRAM

```
;***********************************************************
;*TMS320C5x DTMF Tone decoder source code example
;***********************************************************
; Column
;                    1      2      3       4
;                 1209hz 1336hz 1477hz 1633hz
;                 +------------------------+
;           1         | 1  _| 2  _| 3  _| A  _|
;           697hz     |    0|    1|    2|    3|
;                 +------+-----+-----+-----+
;           2         | 4  _| 5  _| 6  _| B  _|
;        R  770hz     |    4|    5|    6|    7|
;        O        +------+-----+-----+-----+
;        W  3         | 7  _| 8  _| 9  _| C  _|
;          852hz      |    8|    9|    a|    b|
;                 +------+-----+-----+-----+
;           4         | *  _| 0  _| #  _| D  _|
;           941hz     |   |c|   |d|   |e|    f|
;                 +------------------------+
;
;------------------------- DTMF decoder ---------------------
;Y(n) = X(n) + 2Cos(coef)*Y(n–1) – Y(n–2) + 2Cos(coef)*Y(n–1)
; k = N * Freq / Sample rate
;------------------------- Coefficient ----------------------
;coefficent = 32768 * Cos(360 * Freq / Sample rate)
;coef = 360 * Freq / Sample rate
;Sample rate = 8kHz
;------------------------- Data Segment ---------------------
.data
Table  .set   $
Base1r .word  32768 * 8538/10000         ;27906:  697Hz
      .word  32768 * 8226/10000         ;26802:  770Hz
      .word  32768 * 7843/10000         ;25597:  851Hz
      .word  32768 * 7391/10000         ;24295:  941Hz
      .word  32768 * 5821/10000         ;19057: 1209Hz
      .word  32768 * 4982/10000         ;16527: 1336Hz
      .word  32768 * 3993/10000         ;12945: 1477Hz
      .word  32768 * 2843/10000         ; 9166: 1633Hz
;2nd harmonic rate coefficient
Harm1r .word  32768 * 4582/10000         ; 15036:2 x  697Hz
      .word  32768 * 3535/10000         ; 11287:2 x  770Hz
      .word  32768 * 2304/10000         ;  7363:2 x  851Hz
      .word  32768 * 925/10000          ;  3323:2 x  941Hz
      .word  32768 * –3224/10000        ;–10805:2 x 1209Hz
```

```
      .word  32768 *–5036/10000        ;–16384:2 x 1336Hz
      .word  32768 *–6811/10000        ;–22153:2 x 1477Hz
      .word  32768 *–8384/10000        ;–27440:2 x 1633Hz
;      word  27906,26802,25597,24295, 19057, 16527, 12945,  9166
;      word  15036,11287, 7363, 3323,–10805,–16384,–22153,–27440
      .word  1,205,0,0,0,0,0,0,0,0,B1QueCh0
      .word  B1F11Ch31,PProcFlg0,B1QueCh31,B1F31Ch31,0,PProcFlg0,3
      .word  8000h,2,3,4,0,204,200,06h,16h,B1QueCh0
Len .set   $ – Table
;––– u_law table –––
uTab  .word  0e0a1h,0e1a1h,0e2a1h,0e3a1h,0e4a1h,0e5a1h,0e6a1h,0e7a1h
      .word  0e8a1h,0e9a1h,0eaa1h,0eba1h,0eca1h,0eda1h,0eea1h,0efa1h
      .word  0f061h,0f0e1h,0f161h,0f1e1h,0f261h,0f2e1h,0f361h,0f3e1h
      .word  0f461h,0f4e1h,0f561h,0f5e1h,0f661h,0f6e1h,0f761h,0f7e1h
      .word  0f841h,0f881h,0f8c1h
      .word  0f901h,0f941h,0f981h,0f9c1h,0fa01h,0fa41h,0fa81h,0fac1h
      .word  0fb01h,0fb41h,0fb81h,0fbc1h
      .word  0fc01h,0fc31h,0fc51h,0fc71h,0fc91h,0fcb1h,0fcd1h,0fcf1h
      .word  0fd11h,0fd31h,0fd51h,0fd71h,0fd91h,0fdb1h,0fdd1h,0fdf1h
      .word  0fe11h,0fe29h,0fe39h,0fe49h,0fe59h,0fe69h,0fe79h,0fe89h
      .word  0fe91h,0fea9h,0feb9h,0fec9h,0fed9h,0fee9h,0fef9h
      .word  0ff09h,0ff19h,0ff25h,0ff2dh,0ff35h,0ff3dh,0ff45h,0ff4dh
      .word  0ff55h,0ff5dh,0ff65h,0ff6dh,0ff75h,0ff7dh,0ff85h,0ff8dh
      .word  0ff95h,0ff9dh
      .word  0ffa3h,0ffa7h,0ffabh,0ffafh,0ffb3h,0ffb7h,0ffbbh,0ffbfh
      .word  0ffc3h,0ffc7h,0ffcbh,0ffcfh,0ffd3h,0ffd7h,0ffdbh,0ffdfh
      .word  0ffe2h,0ffe4h,0ffe6h,0ffe8h,0ffeah,0ffech,0ffeeh
      .word  0fff0h,0fff2h,0fff4h,0fff6h,0fff8h,0fffah,0fffch,0fffeh
      .word  00000h
;negative
      .word  01f5fh,01e5fh,01d5fh,01c5fh,01b5fh,01a5fh,0195fh,0185fh
      .word  0175fh,0165fh,0155fh,0145fh,0135fh,0125fh,0115fh,0105fh
      .word  00f9fh,00f1fh,00e9fh,00e1fh,00d9fh,00d1fh,00c9fh,00c1fh
      .word  00b9fh,00b1fh,00a9fh,00a1fh,0099fh,0091fh,0089fh,0081fh
      .word  007bfh,0077fh,0073fh
      .word  006ffh,006bfh,0067fh,0063fh
      .word  005ffh,005bfh,0057fh,0053fh,004ffh,004bfh,0047fh,0043fh
      .word  003ffh,003cfh,003afh,0038fh,0036fh,0034fh,0032fh,0030fh
      .word  002efh,002cfh,002afh,0028fh,0026fh,0024fh,0022fh,0020fh
      .word  001efh,001d7h,001c7h,001b7h,001a7h,00197h,00187h,00177h
      .word  0016fh,00157h,00147h,00137h,00127h,00117h,00107h
      .word  000f7h,000e7h
      .word  000dbh,000d3h,000cbh,000c3h,000bbh,000b3h,000abh,000a3h
      .word  0009bh,00093h,0008bh,00083h,0007bh,00073h,0006bh,00063h
      .word  0005dh,00059h,00055h,00051h,0004dh,00049h,00045h,00041h
```

```
      .word  0003dh,00039h,00035h,00031h,0002dh,00029h,00025h,00021h
      .word  0001eh,0001ch,0001ah,00018h,00016h,00014h,00012h,00010h
      .word  0000eh,0000ch,0000ah,00008h,00006h,00004h,00002h,00000h
;---------- bss Segment ----------
             .bss   B1QueCh0,205*32 0      ;Queue Buffer Block1 205*32
B1QueCh31    .set   B1QueCh0+205*31
             .bss   B2QueCh0,205*32        ;Queue Buffer Block2 205*32
B2QueCh1     .set   B2QueCh0+205
             .bss   B3QueCh0,205*32        ;Queue Buffer Block3 205*32
B3QueCh1     .set   B3QueCh0+205
             .bss   B4QueCh0,205*32        ;Queue Buffer Block4 205*32
B4QueCh1     .set   B4QueCh0+205
             .bss   B1F11Ch31,6*32         ;Filter result Block1 6*32
B1F31Ch31    .set   B1F11Ch31+4
B1F11Ch30    .set   B1F11Ch31+6
             .bss   B2F11Ch31,6*32         ;Filter result Block2 6*32
B2F11Ch30    .set   B2F11Ch31+6
             .bss   B3F11Ch31,6*32         ;Filter result Block3 6*32
B3F11Ch30    .set   B3F11Ch31+6
             .bss   B4F11Ch31,6*32         ;Filter result Block4 6*32
B4F11Ch30    .set   B4F11Ch31+6
;----------- bK0 & bK1 =100h–2ffh ----------
CoefBase1r   .usect ”varity”,512
CoefBase4r   .set   CoefBase1r+3
CoefBase3c   .set   CoefBase1r+6
CoefHarm1r   .set   CoefBase1r+8
CoefHarm1c   .set   CoefBase1r+12
CoefHarm4c   .set   CoefBase1r+15
One          .set   CoefBase1r+16            ;*1
FilCout      .set   CoefBase1r+17            ;*receive code counter
Cout0Flg     .set   CoefBase1r+18            ;*count 0 first time=0–>–1 2’s =;–1–>0
PQueFlg      .set   CoefBase1r+19            ;*Recv int use Nth queue block buffer
PProcFlg0    .set   CoefBase1r+20            ;*Queue buffer blockN receive ok = –1
PProcFlg1    .set   CoefBase1r+21
PProcFlg2    .set   CoefBase1r+22
PProcFlg3    .set   CoefBase1r+23
DtmfFlag0    .set   CoefBase1r+24            ;*DTMF On/Off flag 16*2 = 32ch
DtmfFlag1    .set   CoefBase1r+25
PQue         .set   CoefBase1r+26            ;*Recv int use Queue pointer
PB1F11Ch31   .set   CoefBase1r+27            ;*pointer B1F11Ch31
PPProcFlg0   .set   CoefBase1r+28            ;*pointer PProcFlg0
PB1QueCh31   .set   CoefBase1r+29            ;*pointer B1QueCh31
PB1F31Ch31   .set   CoefBase1r+30            ;*pointer B1F31Ch31
PGropSave0   .set   CoefBase1r+31            ;*Main program ar0 save
PGropSave1   .set   CoefBase1r+32            ;*Main program ar1 save
```

```
PGropSave2   .set   CoefBase1r+33          ;*Main program ar2 save
NegMax       .set   CoefBase1r+34          ;*8000
Two          .set   CoefBase1r+35          ;*2
Three        .set   CoefBase1r+36          ;*3
Four         .set   CoefBase1r+37          ;*4
PChSave      .set   CoefBase1r+38          ;*Main program ch process save
BaseLen      .set   CoefBase1r+39          ;*204(205)
HarmLen      .set   CoefBase1r+40          ;*200(201)
D06h         .set   CoefBase1r+41          ;*06h
D16h         .set   CoefBase1r+42          ;*16h
PB1QueCh0    .set   CoefBase1r+43          ;*pointer B1QueCh0
;----------------------- Not Initialization -------------------------
FreqCout4    .set   CoefBase1r+50
TempInt      .set   CoefBase1r+51
TempInt1     .set   CoefBase1r+52
TempMain     .set   CoefBase1r+53
ColMaxNo     .set   CoefBase1r+54
RowMaxNo     .set   CoefBase1r+55
ColMaxDat    .set   CoefBase1r+56
RowMaxDat    .set   CoefBase1r+57
test         .set   CoefBase1r+58
test1        .set   CoefBase1r+59
FilR11b      .set   CoefBase1r+60          ;FilR11b–FilR42b =4*2=8
FilR12b      .set   CoefBase1r+61
FilR31b      .set   CoefBase1r+64
FilR32b      .set   CoefBase1r+65
FilR41b      .set   CoefBase1r+66
FilR42b      .set   CoefBase1r+67
PFilC11b     .set   CoefBase1r+68          ;FilC11b–FilC32b =3*2=6
PFilC22b     .set   CoefBase1r+71
PFilC31b     .set   CoefBase1r+72
PFilC32b     .set   CoefBase1r+73
PQueBuf      .set   CoefBase1r+74          ;Pointer Queue (Main program)
FilR01h      .set   CoefBase1r+75          ;FilC01h–Filc02h =2*2=4
FilR02h      .set   CoefBase1r+76
FilC01h      .set   CoefBase1r+77
FilC02h      .set   CoefBase1r+78
Last0Ch0     .set   CoefBase1r+79
Last1Ch0     .set   CoefBase1r+80          ;Last0Ch0–Last2Ch31 = 3*32=96
Last2Ch0     .set   CoefBase1r+81
Last2Ch31    .set   CoefBase1r+174
F11Ch31      .set   CoefBase1r+175         ;F11Ch31–F32ch0 = 6*32=192
F12Ch31      .set   CoefBase1r+176
F21Ch31      .set   CoefBase1r+177
F32Ch0       .set   CoefBase1r+366
```

```
;----------------------- Set up the ISR vector --------------
      .sect  ”vectors”
       b     Main
       b     Int1             ;02  INT1 interrupt
       b     Int2             ;04  INT2 interrupt
       b     Int3             ;06  INT3 interrupt
      .space    2 * 16        ;08  TINT interrupt
Rint:  b     RecvInt          ;0A  Serial port receive interrupt RI
XInt:  b     TransInt         ;0C  Serial port transmit interrupt X
TRnt:  b     TRecvInt         ;0E  TDM receive interrupt
TXnt:  b     TTransInt        ;10  TDM transmit interrupt
      .space    18 * 16       ;12
;-------------------------------------------------------------
      .text
      .mmregs
;*************************************************************
;                     Main program
;*************************************************************
;------------------------- Initialization -------------------
Main:
      setc   intm             ;ST0
      setc   ovm
      ldp    #0
      opl    #003ch,PMST      ;ndx=1 ram=ovly=1
      splk   #00c8h,SPC       ;soft=free=0 rrst=xrst=1 txm=0 mcm=0 fsm=1 fo=0
      splk   #00c8h,TSPC      ;same SPC
      spm    0                ;ST1
      setc   sxm
      clrc   cnf              ;B1 = data mem
      lmmr   INDX,#Two        ;indx=2
      zap
      sacl   CWSR             ;CWSR=PDWSR=IOWSR=0
      sacl   PDWSR
      sacl   IOWSR
      splk   #06h,IMR         ;Series Recv Int on & Int2 & Int3
;----------------------- block move -------------------------
      mar    *,ar1            ;table move
      lar    ar1,#CoefBase1r
      rpt    #Len–1
      blpd   #Table,*+
;-------------------------------------------------------------
      lar    ar1,#F11Ch31     ;clear filter result buffer 6*32
      rpt    #191
      sacl   *+
      clrc   intm
```

```
        ldp     #CoefBase1r          ;check if need to process
MainChk0:
        lar     ar0,PGropSave0
        lar     ar1,PGropSave1
        lar     ar2,PGropSave2
MainChk1
        lacc    *
        bcnd    MainProc0,neq        ;PProcFlg(n)=? –1
        mar     *+,ar0
        mar     *+,ar2
        banz    MainChk1,*–,ar1
        lar     ar0,#0
        lar     ar1,#PProcFlg0
        lar     ar2,#3
        b       MainChk1
MainProc0:
        zap
        sacl    *                    ;0 –> PProcFlg(n)
        sar     ar0,PGropSave0
        sar     ar1,PGropSave1
        sar     ar2,PGropSave2
        lacc    #31                  ;set PChSave=#31(32ch)
        sacl    PChSave
MainProc1
        lt      PGropSave0           ;Group start then set
        mpy     #192                 ;1.PFilC31b,PFilC31b
        pac
        adds    PB1F31Ch31           ;PB1F31Ch31
        sacl    PFilC31b
        add     One
        sacl    PFilC32b
        lt      PGropSave0           ;2.PB1QueCh31
        mpy     #205*32
        pac
        bd      MainProc2
        adds    PB1QueCh31           ;PB1QueCh31
        sacl    PQueBuf
MainProc3:
        sar     ar1,PChSave          ;channel change then
        lacc    PQueBuf,0            ;1.PQueBuf–205,channel–1
        sub     #205
        sacl    PQueBuf
        lacc    PFilC31b             ;2.PFilC31b+6&PFilC32b+6
        add     #6
        sacl    PFilC31b
```

```
        add     One
        sacl    PFilC32b
MainProc2:
        lmmr    INDX,#Two           ;indx=2
;***********************************************************
;       Caculate Energy at Col 1,2,3,4 and
;       Find Max Energy of Col 1,2,3,4
;***********************************************************
;ENERGY = [(Y(n-2)-Y(n-1))/2]^2 - Y(n-1)*Y(n-2)*(coef-1)
EnergyCol:
        lar     ar0,Two
        lar     ar1,PFilC32b
        lar     ar2,#CoefBase3c
EnergyCol0:
        call    EnergySr,*,ar2
        sach    *-,1,ar0
        banz    EnergyCol0,*-,ar1
;------------------- Find Col1,2,3 Energy Max ---------------
;ColMaxDat = col peak data  ColMaxNo= col peak number
FindColMax:
;       mar     *,ar1
        lacc    Two                 ;ColMaxNo=2
        sacl    ColMaxNo
FindCol
        lar     ar2,One             ;ar2=1
        lar     ar1,PFilC31b        ;ColMaxDat=(PFilC31b) & ar1=PFilC21b
        lacc    *0-
        sacl    ColMaxDat
FindCol0
        lacc    ColMaxDat
        sub     *
        bcnd    FindCol1,geq        ;col_max - (ar1)
        sar     ar2,ColMaxNo
        lacc    *
        sacl    ColMaxDat
FindCol1
        mar     *0-,ar2
        banz    FindCol0,*-,ar1
;***********************************************************
;*      Check Col 1,2,3,4 level and Relative peak
;***********************************************************
ChkLev:
        lacc    ColMaxDat           ;ColMaxDat must >2^4
        sub     One,4
        bcnd    NotCode,lt
```

```
;----------------- Check Col1,2,3 Relative ratio ------------
ChkRelCol:
;       mar    *,ar1
        lar    ar0,Two              ;ar0=2
        lar    ar1,PFilC31b         ;ar1=PFilC31b
        lar    ar2,One              ;ar2=1
        lt     ColMaxDat            ;P=ColMaxDat*683
        mpy    #683
ChkRelCol0
        lacc   *0-,12,ar0           ;(ar1) - ColMaxDat*683 & ar1-2
        spac
        bcnd   ChkRelCol1,lt
        mar    *,ar2                ;ar2 - 1
        mar    *-,ar0
ChkRelCol1
        banz   ChkRelCol0,*-,ar1    ;ar0 =?0
        mar    *,ar2
        banz   NotCode,*,ar1        ;ar2=?0
;----------------- Clear ROW & Harmonic Filter result -------
ClearFil:
        lar    ar1,#FilR11b
        zap
        rpt    #7
        sacl   *+
        lar    ar1,#FilR01h
        rpt    #3
        sacl   *+
;*************************************************************
;       Process DFT of Row 1,2,3,4
;*************************************************************
DFTRow:
        lar    ar0,BaseLen
        lar    ar1,#FilR12b
        lar    ar2,PQueBuf
        lar    ar3,#CoefBase1r
        lacc   Four
        sacl   FreqCout4
        call   DFTCall,*,ar3
;*************************************************************
;       Caculate Energy at Row 1,2,3,4 and
;       Find Max Energy of Row 1,2,3,4
;*************************************************************
;ENERGY = [(Y(n-2)-Y(n-1))/2]^2 - Y(n-1)*Y(n-2)*(coef-1)
EnergyRow:
        lar    ar0,Three
```

```
        lar    ar1,#FilR42b
        lar    ar2,#CoefBase4r
EnergyRow0:
        call   EnergySr,*,ar2          ;21
        sach   *–,1,ar0                ;23..ar1 – 1 (Y(n–2)
        banz   EnergyRow0,*–,ar1       ;24..ar0(FreqCount)=?0 ar0–1
;-------------------- Find Row 1,2,3,4 Energy Max -----------
;RowMaxDat = row peak data    RowMaxNo= row peak number
FindRowMax:
;       mar    *,ar1
        lacc   Three                   ;RowMaxNo=3
        sacl   RowMaxNo
FindRow
        lar    ar2,Two                 ;ar2=2
        lar    ar1,#FilR41b            ;RowMaxDat=(#FilR41b) & ar1=#FilR31b
        lacc   *0–
        sacl   RowMaxDat
FindRow0
        lacc   RowMaxDat
        sub    *
        bcnd   FindRow1,geq            ;RowMax – (ar1)
        sar    ar2,RowMaxNo
        lacc   *
        sacl   RowMaxDat
FindRow1
        mar    *0–,ar2
        banz   FindRow0,*–,ar1
;************************************************************
;       Check Peak level and Twist
;************************************************************
TwistLevel:
        lacc   ColMaxDat
        sub    RowMaxDat
        bcnd   TwistLevel0,geq
TwistRev
        lac    ColMaxDat               ;row > col
        sub    One,4
        bcnd   NotCode,lt
        lac    RowMaxDat               ;row_max – col_max*12
        sub    ColMaxDat,3             ;8dB = 6
        sub    ColMaxDat,2
        bcnd   NotCode,geq
        b      DFTHarmRow
TwistLevel0:
        lac    RowMaxDat               ;col > row
```

```
        sub     One,4
        bcnd    NotCode,lt
        lac     ColMaxDat               ;col_max – row_max*3
        sub     RowMaxDat               ;4dB = 3
        sub     RowMaxDat,1
        bcnd    NotCode,geq
;*************************************************************
;       Process 2nd Harmonic DFT of Max Row Energy and
;           Check Row Harmonic Energy
;*************************************************************
DFTHarmRow:
        mar     *,ar3
        lmmr    INDX,#RowMaxNo
        lar     ar0,HarmLen
        lar     ar1,#FilR02h
        lar     ar2,PQueBuf
        lar     ar3,#CoefHarm1r
        mar     *0+,ar3
        lt      *,ar2                   ;1..T = Cos(*)
DFTHarmRow0
        lacc    *+,12,ar1               ;3..X(n)
        sub     *–,16                   ;X(n)–Y(n–2)    ––> Acc
        mpy     *                       ;Cos(*) * Y(n–1)  ––> P
        ltd     *,ar3                   ;X(n)+ Cos(*) * Y(n–1)–Y(n–2)Y(n–1)––>Y(n–2)
        lta     *,ar1                   ;X(n)+2Cos(*) * Y(n–1)–Y(n–2)
        apac
        apac                            ;X(n)+2*2Cos(*)*Y(n–1)–Y(n–2)
        sach    *+,0,ar0                ;2..––> Y(n–1) Q30
        bcnd    HarmRowLev,ov           ; check overflow
        banz    DFTHarmRow0,*–,ar2      ;4..ar0(len)=?0
;–––––––––––––––––––– Check Row Harmonic Energy –––––––––––––
HarmRowLev:
;       mar     *,ar2
        lar     ar2,#CoefHarm1r         ;ar2 = harmonic coefficent pointer
        mar     *0+,ar2                 ;= #coeff + row_max*2
        lar     ar1,#FilR02h            ;ar1 = harmonic data pointer= #dat + row_mxp*2
        call    EnergySr,*,ar2
        lt      RowMaxDat               ;harmonic energy row_max/4
        mpy     #4096
        spac
        spac
        bcnd    NotCode,geq
```

```
;**********************************************************
;      Process 2nd Harmonic DFT of Max Col Energy and
;         Check Col Harmonic Energy
;***********************************************************
DFTHarmCol:
       mar    *,ar3
       lmmr   INDX,#ColMaxNo
       lar    ar0,HarmLen
       lar    ar1,#FilC02h
       lar    ar2,PQueBuf
       lar    ar3,#CoefHarm1c
       mar    *0+,ar3
       lt     *,ar2               ;1..T = Cos(*)
DFTHarmCol0
       lacc   *+,12,ar1           ;3..X(n)
       sub    *–,16               ;X(n)–Y(n–2)––> Acc
       mpy    *                   ;Cos(*) * Y(n–1)––> P
       ltd    *,ar3               ;X(n)+ Cos(*) * Y(n–1)–Y(n–2)Y(n–1)––>Y(n–2)
       lta    *,ar1               ; X(n)+2Cos(*) * Y(n–1)–Y(n–2)
       apac
       apac                       ; X(n)+2*2Cos(*)*Y(n–1)–Y(n–2)
       sach   *+,0,ar0            ;2..––> Y(n–1) Q30
       bcnd   HarmColLev,ov       ;check overflow
       banz   DFTHarmCol0,*–,ar2  ;4..ar0(len)=?0
;––––––––––––––––––––– Check Col Harmonic Energy ––––––––––––
HarmColLev:
       mar    *,ar2
       lar    ar2,#CoefHarm1c     ;ar2 = harmonic coefficent pointer
       mar    *0+,ar2             ;= #Coef + ColMax*2
       lar    ar1,#FilC02h        ;ar1=harmonic data pointer= #dat + ColMax*2
       call   EnergySr,*,ar2
       lt     ColMaxDat           ;harmonic energy ColMax/4
       mpy    #4096
       spac
       spac
       bcnd   NotCode,geq
;**********************************************************
;                    Check for valid number
;      Compare number with last number and 2nd to last number
;***********************************************************
;–––––––––––––––––––––– load recognized number ––––––––––––––
       mar    *,ar1
       lt     PChSave             ;ch*3 + #Last1Ch0
       mpy    Three
       lacc   #Last0Ch0
```

```
        apac
        sacl   TempMain,0
        lar    ar1,TempMain
        mar    *+,ar1              ;Last0Ch0+1 = Last1Ch0
        dmov   *-                  ;last1 --> last2
        dmov   *                   ;last0 --> last1
;------------------------- Check for new number ------------
ChkNewCode:
        lacc   RowMaxNo,2          ;row_mxp*2^2 + col_mxp + Ch*2^8--> last0
        add    ColMaxNo
        add    PChSave,8
        sacl   *+                  ;--> last0
        mar    *+,ar1
        sub    *-                  ;last0 =? last2
        bcnd   NotCode1,eq
        lac    *-                  ;last0 =? last1
        sub    *,0,ar1
        bcnd   ChkNewCode0,neq
        out    *,3                 ;Output Last0 Digital data
ChkNewCode0
        lar    ar1,PChSave
        banz   MainProc3,*-,ar1    ;PChSave=?0 & PChSave-1
        b      MainChk0
;------------------------- NOT New CODE ---------------------
NotCode:
        lt     PChSave             ;ch*3 + #Last0Ch0
        mpy    Three
        lacc   #Last0Ch0
        apac
        sacl   TempMain,0
NotCode1
        lar    ar1,TempMain
        mar    *,ar1
        lacc   #0ffh
        bd     ChkNewCode0
        dmov   *                   ;last0 --> last1
        sacl   *
;************************************************************
;          DFT Subroutine
;************************************************************
DFTCall:
        lt     *,ar2               ;1..T = Cos(*)
DFTCall0
        lacc   *+,12,ar1           ;3..X(n)
        sub    *-,16               ;X(n)-Y(n-2)--> Acc
```

```
        mpy     *                   ;Cos(*) * Y(n-1) --> P
        ltd     *,ar3               ;X(n)+ Cos(*) * Y(n-1)-Y(n-2)Y(n-1)-->Y(n-2)
        lta     *,ar1               ;X(n)+2Cos(*) * Y(n-1)-Y(n-2)
        apac
        apac                        ;X(n)+2*2Cos(*)*Y(n-1)-Y(n-2)
        sach    *+,0,ar0            ;2..--> Y(n-1) Q30
        bcnd    DFTCall1,ov         ;check overflow
        banz    DFTCall0,*-,ar2     ;4..ar0(len)=?0
DFTCall1
        lar     ar0,BaseLen
        lar     ar2,PQueBuf
        mar     *,ar1
        mar     *0+,ar3
        mar     *+,ar3
        lacc    FreqCout4
        sub     One
        sacl    FreqCout4
        bcnd    DFTCall,neq
        ret
;*************************************************************
;               ENERGY Subroutine
;*************************************************************
EnergySr:
        lac     NegMax,15           ;(coef-1)/2
        add     *-,15,ar1           ;ar2=&coeff-1
        sach    TempMain
        lt      *-                  ;Y(n-2)*(coef-1)/2
        mpy     TempMain
        pac
        sach    TempMain,1
        lt      *+                  ;Y(n-1)*Y(n-2)*(coef-1)/2
        mpy     TempMain
        pac
        sach    TempMain,1
        lac     *-,15               ;ABS[Y(n-2)-Y(n-1)]/2
        sub     *,15
        abs
        sach    *
        lt      *                   ;[Y(n-2)-Y(n-1)]/2 ^2
        mpy     *
        pac                         ;[(Y(n-2)-Y(n-1))/2]^2-Y(n-1)*Y(n-2)*(coef-1)
        sub     TempMain,15
        ret
```

```
;**************************************************************
;                   Receive int subroutine
;         Processing 2 channels in single Interrupt
;**************************************************************
RecvInt:
      ldp    #TempInt
      lamm   DRR               ;DRR u_law Expanding
      sacl   TempInt
      mar    *,ar4
      and    #0ffh
      add    #uTab
      tblr   *,ar5
      lar    ar7,#3            ;4 Freq of 1st channel
RecvInt4
      lt     *-,ar4            ;4 frequency IIR
      lacc   *,12,ar6
      sub    *-,16
      mpy    *
      ltd    *
      apac
      apac
      apac
      sach   *-,0,ar7
      bcnd   RecvInt5,ov
      banz   RecvInt4,*-,ar5
RecvInt5
      adrk   #4                ;ar5+4 pointer process
      lar    ar7,#3
      lacl   TempInt           ;Get high byte data
      bsar   8
      mar    *,ar4
      adrk   #205
      add    #uTab
      tblr   *,ar5
RecvInt6
      lt     *-,ar4            ;4 frequency IIR of 2nd channel
      lacc   *,12,ar6
      sub    *-,16
      mpy    *
      ltd    *
      apac
      apac
      apac
      sach   *-,0,ar7
      bcnd   RecvInt7,ov
```

```
        banz   RecvInt6,*–,ar5
RecvInt7
        adrk   #4                  ;pointer process
        mar    *,ar4
        adrk   #205
        lacc   #0ffffh             ;output level 0 to phone
        samm   DXR
        rete
;***************************************************************
;       Other Interrupt Subroutines
;***************************************************************
;------------------ TDM transmit ISR -------------------
TTransInt:
        rete
;------------------ TDM receive ISR --------------------
TRecvInt:
        rete
;------------------ DXR transmit ISR -------------------
TransInt:
        rete
;------------------ INT1 ISR ---------------------------
Int1:
        rete
;------------------ INT2 ISR(Frame 0) ------------------
Int2:
        ldp    #FilCout
        lmmr   IMR,#D16h           ;turn on IMR Recv int
        lacc   FilCout,0           ;FilCout =?0
        bcnd   Int10,neq
        lacc   Cout0Flg            ;Cout0Flg complement
        cmpl
        sacl   Cout0Flg
        bcnd   Int11,neq
        lamm   DRR                 ;read DRR data
        lacc   #205
Int10
        sub    One                 ;FilCout – 1
        sacl   FilCout,0
        lar    ar5,#CoefBase3c     ;set Ar5 – Ar6
        lar    ar6,#F32Ch0
        lar    ar4,PQue            ;set ar4 = PQue & PQue + 1
        lacc   PQue
        add    One
        sacl   PQue
        rete
```

```
Int11:
        lmmr   IMR,#D06h           ;stop Recv Int
        lt     PQueFlg             ;caculate Filter result Block addr
        mpy    #192
        pac
        adds   PB1F11Ch31
        sacl   TempInt
        mar    *,ar5               ;Block move F11Ch31 --> BnF11Ch31
        lar    ar5,TempInt
        rpt    #191
        bldd   #F11Ch31,*+
        lar    ar5,#F11Ch31        ;Clear F11ch31 – F32Ch0
        zap
rpt #191
        sacl   *+
        lacc   PQueFlg             ;set PProcFlg(n) = -1
        adds   PPProcFlg0
        sacl   TempInt
        lar    ar5,TempInt
        zap
        sub    One
        sacl   *
        lacc   PQueFlg             ;PQueFlg+1 & #3 -->PQueFlg
        add    One
        and    Three
        sacl   PQueFlg
        lt     PQueFlg             ;caculus PQue addr
        mpy    #205*32
        pac
        adds   PB1QueCh0
        sacl   PQue
        rete
;---------------- INT3 ISR(Receive Code) -----------------
;Get Tone Format: High 8Bit         Low 8bit
;  Bit0-4 = channel    Bit0 = 0:DTMF On 1:DTMF Off
Int3:
        ldp    #TempInt
        in     TempInt,PA0         ;in data
        bit    TempInt,15
        lacc   TempInt,8           ;Get channel
        and    #0fh,16
        sach   TempInt1
        bcndd  Int21,tc
        lt     TempInt1            ;mark or unmark DtmfFlag bit
        lact   One
```

```
        bit     TempInt,11
        cmpl
        bcnd    Int22,tc
        and     DtmfFlag0
        sacl    DtmfFlag0
        rete
Int22:
        and     DtmfFlag1
        sach    DtmfFlag1
        rete
Int21:
        bit     TempInt,11
        bcnd    Int22,tc
        or      DtmfFlag0
        sacl    DtmfFlag0
        rete
Int23:
        or      DtmfFlag1
        sach    DtmfFlag1
        rete
        .end
```