Integrated Automotive Signal Processing and Audio System Using the TMS320C3x

Uday Gupta, Micheal Bychowsky, and Dr. Sen M. Kuo
Department of Electrical Engineering
Northern Illinois University
Dekalb, IL 60115

This application report consists of one of the entries in a contest, The 1995 TI DSP Solutions Challenge. The report was designed, prepared, and tested by university students who are not employees of, or otherwise associated with, Texas Instruments. The user is solely responsible for verifying this application prior to implementation or use in products or systems.

Literature Number: SPRA095
March 1997

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

List of Illustrations

ABSTRACT

An integrated acoustic signal processing system, which provides adaptive noise cancellation, acoustic echo cancellation, and adaptive-active noise control for hands-free cellular phones, is developed based on the TMS320C3x. This system provides high-quality full-duplex voice communication and reduces the acoustic noise inside the automobile passenger compartment. The integration of this system with the existing car audio system reduces the overall system cost. The Musical Interference Suppression (MIS) process is developed to cancel the interference of music while updating the coefficients of adaptive filters. The MIS filters also model the electro-acoustic paths that are required by the active noise controller.

INTRODUCTION

Why use a hands-free cellular phone system? Dialogues such as the following should give a clear picture of the inconvenience resulting from the use of existing cellular phones: “Excuse me, honey, let me make a sharp turn”, “Hold on, I need to make some turns”. This inconvenience is caused by the use of a handset in the existing cellular phone system. Drivers often have to put down the handset to respond to two-hand jobs, such as making turns, and then return to their conversations. Interruptions in phone conversations are inconvenient and costly because cellular charges are expensive. Another important concern is safety. Imagine a car driver using a cellular phone with only one hand on the steering wheel. It would seem meaningless to have an anti-lock brake system and air bag since the driver does not have both hands available to operate the steering wheel. Therefore, the hands-free cellular phone system is becoming a must for drivers who use cellular phones.

The use of a hands-free cellular phone manifests itself in the problem of acoustic echo being transmitted to the far-end talker. In addition, noise causes two unpleasant situations: speech transmission mixed with a high level of background noise and the noisy environment inside the car. As cellular phones and compact disc (CD) stereos are brought into more cars, drivers deserve to enjoy these high-tech electronic products. However, with the engine running and wind blowing, the sound quality in both systems is often degraded by background noises. Therefore, reducing background noises inside the car is highly desired.

This project designs a TMS320C3x prototype for an integrated automotive signal processing and audio system. The integrated system was designed for drivers to drive safely by using a full-duplex hands-free cellular phone without the problems of acoustic echo and ambient noise in transmission, and to reduce the ambient noise inside the car. The integrated system operates in the following modes:

- Hands-free cellular phone (HFCP) mode: The system performs both acoustic echo cancellation (AEC) [1-3] to provide full-duplex communication capability and adaptive noise cancellation (ANC) [4] to reduce background noise. In this mode, the audio is turned “off” by the integrated system.
- Adaptive-active noise control (AANC) [5-6] mode: The system performs adaptive-active noise control to reduce ambient noise inside the car. In this default operation mode, the audio system may be turned “on”.

When one mode is active, the other mode is disabled. Therefore, a single TMS320C3x digital signal processor (DSP) chip can provide the necessary computational requirements, thereby reducing system cost.

Figure 1 shows the block diagram of an integrated automotive signal processing and audio system integrated with a regular cellular phone (analog or digital) and audio system. The integration with the car audio system makes it possible to use the existing loudspeakers and power amplifiers for both the hands-free cellular phone and active noise control to further reduce the cost of the system. By integration of the existing car audio system, only two extra microphones are needed for the near-end speech pick-up in HFCP mode or residual noise pick-up in AANC mode.

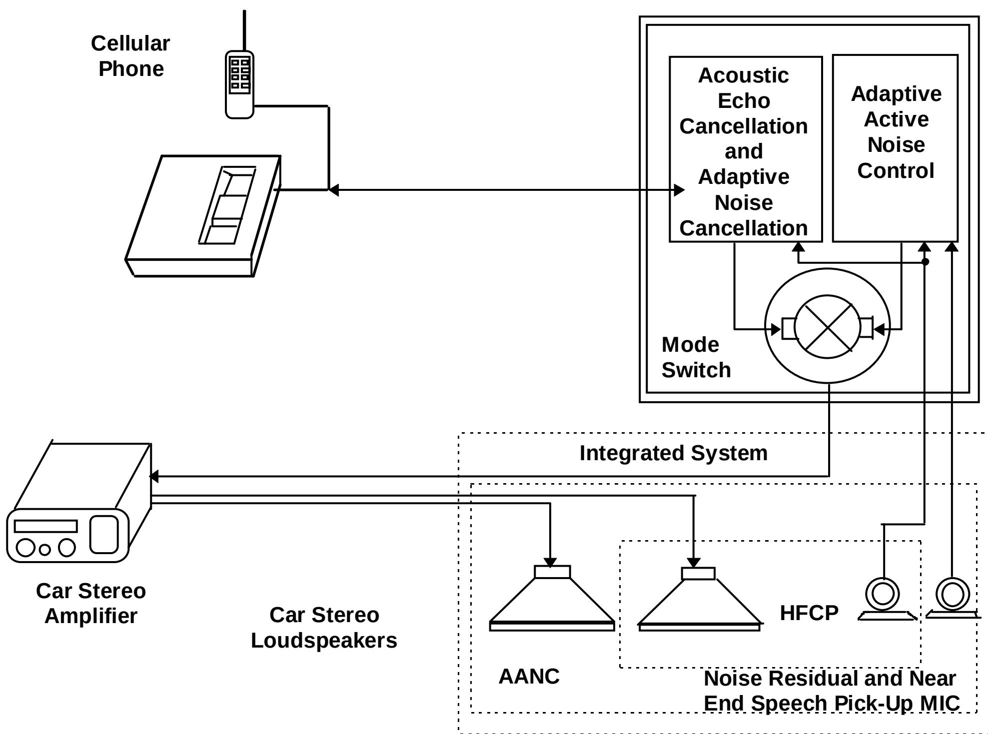

Figure 1. Integrated Automotive Signal Processing System

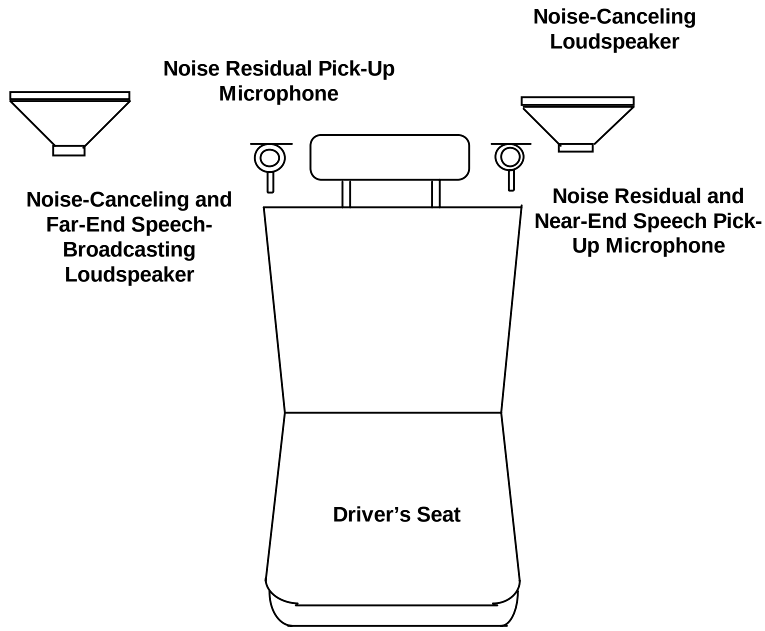

Figure 2. Configuration of Microphones and Loudspeakers for the Integrated System

The integrated system in active noise control mode is designed to acoustically cancel background noise inside the car using the existing stereo system to produce the anti-noise (as shown in Figure 2). In the prototype, two microphones are placed on both sides of the driver's headrest to pick up residual noise. By utilizing the stereo loudspeakers as the canceling loudspeakers, AANC mode can provide a noise-free zone surrounding the driver's head. This arrangement does not affect the performance of the audio system. In HFCP mode, one of the two microphones is used as the primary microphone to pick up near-end speech, and one of the loudspeakers is used to broadcast the far-end speech. In this mode, the stereo system is turned "off" automatically by the integrated system.

In subsequent sections, a hardware and software description of the integrated system is presented along with the experimental setup and results.

HARDWARE DESCRIPTION

The integrated system is based on a single TMS320C3x DSP. Figure 3 shows the block diagram of the system hardware. Spectrum Signal Processing's TMS320C30 system board was used for software development and experiments. In production, the low cost TMS320C32 can be used. The system board has two channel input/output (I/O) capabilities. Both of the channels have 16-bit analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). The integrated system requires six input and three output channels. The input channels are used for the revolutions-per-minute (RPM) signal from the car (a reference microphone was used to obtain the reference noise signal in the prototype), the received signal from the cellular base, two microphones placed at both sides of the driver's headrest, the audio signal from the stereo system, and the mode switch to determine whether the system is in HFCP mode or AANC mode. The output channels are used for the two loudspeakers and transmit signal to the cellular base. To meet I/O requirements, a four-channel I/O board consisting of two daughter modules, with Burr Brown 16-bit ADCs/DACs on it, was connected to the system board through the DSPLINK on the system board. To synchronize the operation of the system and the I/O board, the clock on the system board was disabled and daughter module A on the four-channel I/O board was used to provide the necessary clock signals for the system board and daughter module B. Therefore, the hardware is configured as shown in Table 1.

Table 1. Hardware Configuration

BOARD	CONFIGURATION
System Board	Slave
Daughter Module A	Master
Daughter Module B	Slave

There is a three-cycle delay for I/O on the four-channel board. This delay was compensated for in the software by delaying the I/O signals on the system board.

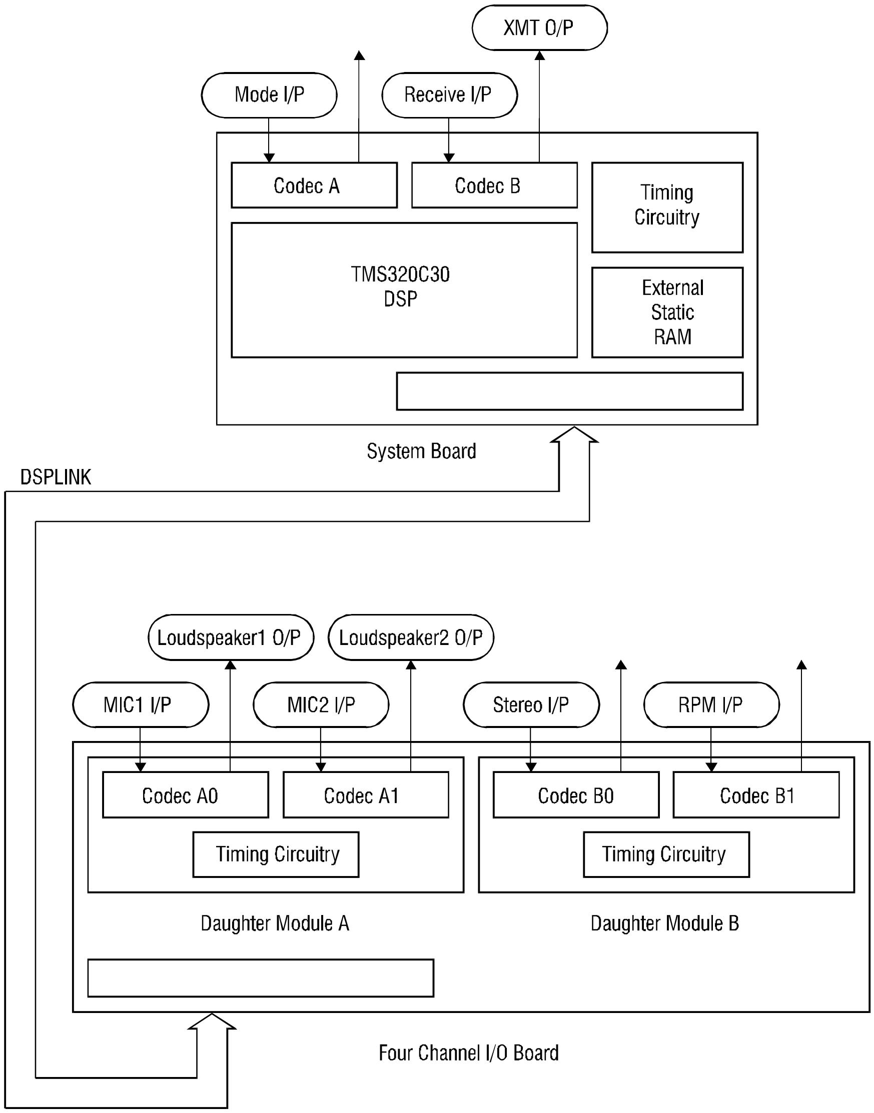

Figure 3. Block Diagram of System Hardware

The system operates at a sampling rate of 8 kHz. The cut-off frequency for the anti-aliasing and the reconstruction filters was set at 3.3 kHz. This was achieved by changing the SIP resistors on both the system board and four-channel I/O board to appropriate values. External SRAM on the system board is used for both program and data storage. Table 2 lists the memory map.

Table 2. Memory Map

ADDRESS	FUNCTION
0000 - 00CFh	Interrupt Vectors
00D0 - 01FFh	Program
2000 - FFFFh	Data

The I/O channels on the system board are mapped to the peripheral bus memory-mapped registers (808000h - 8097ffh), and I/O channels on the four-channel I/O board are mapped to the expansion bus (800000h - 801FFFh). A single external interrupt source (INT0) from the four-channel I/O is used.

SOFTWARE DESCRIPTION

The software developed for the integrated system includes the following functions:

- Mode detection
- Adaptive noise cancellation
- Acoustic echo cancellation
- Speech detection
- Adaptive-active noise control

The integrated system operates in two modes: HFCP mode or AANC mode. A select switch provides specific input for the software to decide in which mode the integrated system should work. A detailed flowchart of system software is given in Appendix A.

Hands-Free Cellular Phone Mode

In hands-free cellular phone mode, the system performs the following functions:

- Adaptive noise cancellation
- Speech detection
- Acoustic echo cancellation

Figure 4 shows the system block diagram in HFCP mode. In the figure, $d(n)$ is the near-end speech picked up by the primary microphone, and $z(n)$ is the far-end speech signal from the cellular phone. The background noise picked up by the primary microphone was canceled through adaptive noise cancellation (ANC). The reference microphone is placed close to the noise source to sense the reference noise for ANC. The ANC is a 256-tap adaptive filter that uses the normalized least mean square (LMS) algorithm [7]. In addition to the background noise, the primary microphone also picks up the far-end speech output from the loudspeaker. This acoustic echo is canceled through acoustic echo cancellation (AEC). The AEC is a 200-tap adaptive filter that uses the normalized LMS algorithm. The AEC filter models the echo path between the loudspeaker and the microphone inside the car, canceling the echo caused by the acoustic coupling between the loudspeaker and the microphone.

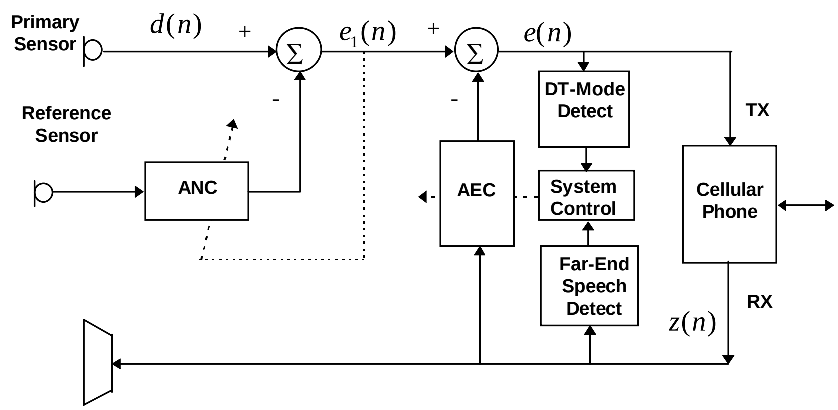

Figure 4. Block Diagram of the System in HFCP Mode

The detection of receive (RX) and double-talk (DT) modes is very important for proper operation of AEC and to avoid the corruption of AEC filter coefficients. RX mode is detected by estimating the noise floor of the far-end speech signal $z(n)$ and comparing it with the short-term power estimate to determine the presence of speech. DT mode detection is based on the echo-return loss enhancement (ERLE) factor, which is defined as the ratio of energy in the signal $e_1(n)$ before echo cancellation and energy in the signal $e(n)$ after echo cancellation. For the DT detector to work, it requires that the AEC filter provide some initial cancellation on system onset. Since the system is integrated with the car stereo system, the music signal can be used as a training signal to model the echo path for a duration of 10 seconds. Therefore, before actual conversation starts, the AEC filter has converged to the echo path. During the HFCP mode, the stereo system is turned off. The

AEC filter is updated during the RX mode only. During DT mode, the fixed filter from the previous RX mode is used to cancel the acoustic echo, and updating of the AEC filter is frozen to avoid corruption of the AEC filter coefficients.

Adaptive-Active Noise Control Mode

In AANC mode, the system performs the cancellation of undesired noise acoustically by generating anti-noise through the canceling loudspeakers. To create a three-dimensional noise-free zone, multiple adaptive filters are needed to generate the canceling noises. Two error microphones are located at both sides of the driver's headrest to pick up the residual noise for updating the AANC filter's coefficients. Since the error signals are measured by the error microphones, interference from the car audio system degrades the performance of AANC. In addition, to ensure the stability and convergence of the adaptive filters, it is necessary to compensate for the transfer functions of the error paths from the canceling loudspeakers to the error microphones by using the multiple-channel filtered-X LMS algorithm [8]. The integrated system has access to the car audio system—therefore, musical interference suppression (MIS) [9] filters were developed which converge to the corresponding error paths from the loudspeakers to the microphones to cancel the music interference to AANC. These MIS filters estimate the error paths on-line, which is very important in improving the performance of AANC.

Figure 5 shows the block diagram of the system in AANC mode, where $W_1(z)$ and $W_2(z)$ are the AANC filters, $C_{11}(z)$, $C_{12}(z)$, $C_{21}(z)$, and $C_{22}(z)$ are the MIS filters, $e_1(n)$ and $e_2(n)$ are the residual noises measured by the two microphones. As mentioned earlier, the music from the audio system becomes interference to the AANC system. The MIS system is used to cancel the music picked up by the microphones. The music-free residual noises $e_1'(n)$ and $e_2'(n)$ are used to update $W_1(z)$ and $W_2(z)$. The adaptive filters in the MIS system serve dual purposes—they not only provide musical interference suppression, but also model the error paths from the canceling loudspeakers to the microphones. These models are used in the filtered-X LMS algorithm to update the AANC filters.

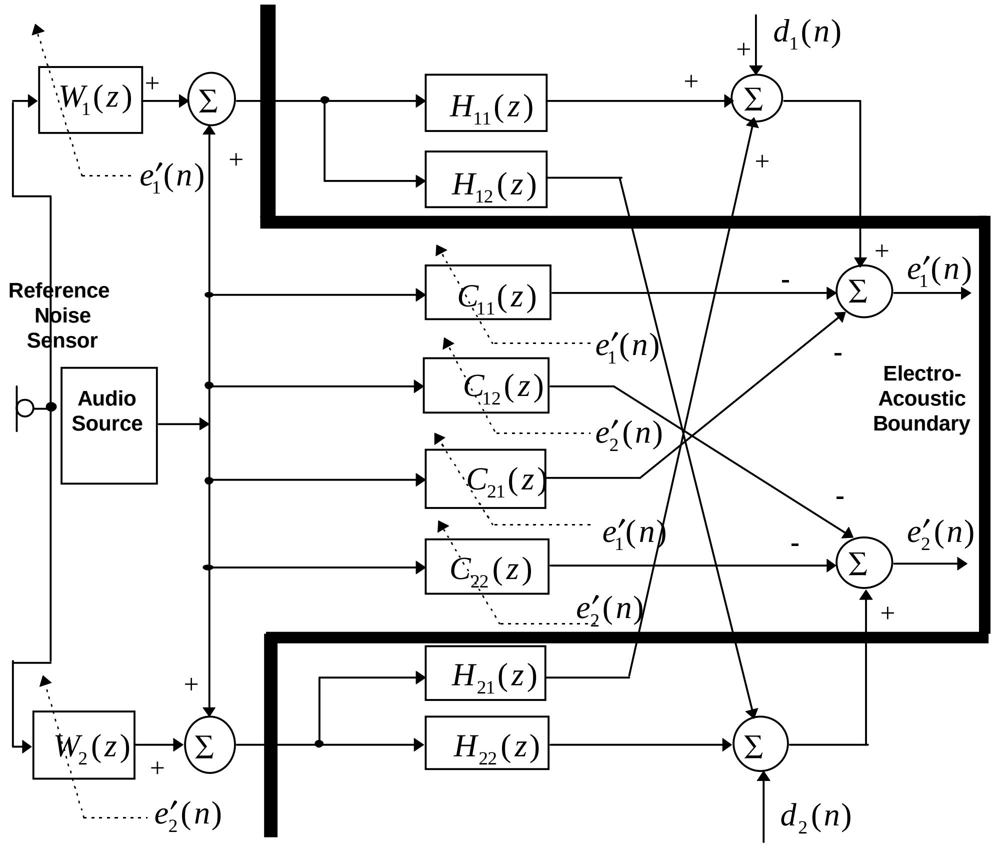

Figure 5. Block Diagram of the System in AANC Mode

Multichannel on-line modeling of error paths is a difficult task because of the acoustical coupling between the individual channels. Therefore, only two error paths from one of the loudspeakers to the two microphones can be estimated at a time. The error paths are time-varying due to the temperature variations and movement of objects. Consequently, on-line modeling of these error paths is needed. On-line modeling has to be conducted without shutting "off" one channel so that the driver and other passengers in the car can enjoy the dual-channel stereo system. This was achieved by updating only two error paths from one loudspeaker to the two microphones while doing the musical interference suppression with the other two fixed filters. The updating of a pair of error paths was done alternately for a duration of 10 seconds each. For this approach to work, it is necessary that before the complete AANC system is turned "on", the MIS filters should have converged to a close approximation of the error paths. This was achieved by doing initial on-line modeling of error paths using music with only one channel “on” for 30 seconds for each channel. This way, when both the channels are turned “on” in full AANC mode, the MIS filters would be able to cancel some interference and, at the same time, keep tracking the time-varying error paths.

The acoustic noise inside the car is mostly periodic and lies in the frequency range below 1 kHz. Therefore, the 8 kHz sampling rate requires a higher order of filter to model the error paths. To overcome this problem, all the signals used for AANC were downsampled to 2 kHz in the software. The AANC mode is implemented with AANC filters of order 64 and MIS filters of order 40. The AANC filters are updated using the normalized filtered-X LMS algorithm, and the MIS filters are updated using the normalized LMS algorithm. The order of the AANC and MIS filters was optimized according to the computation power available for processing at a sampling rate of 8 kHz.

EXPERIMENTAL SETUP AND RESULTS

The experimental setup consists of a wood structure having the shape of a car cabin where the driver’s seat is present. This wood structure is placed in an acoustic chamber to simulate the car in an open field. Two 10.0” loudspeakers (JBL 2123H) were used as audio loudspeakers for playing the music, for sending out the canceling noise during AANC mode, and for broadcasting the far-end speech during the HFCP mode. Two microphones (Shure SM98-A) are placed on both sides of headrest to pick up residual noise during the AANC mode and near-end speech during the HFCP mode. A Carvin FET 450 power amplifier was used as the audio-power amplifier. A Symmetric pre-amplifier was used to amplify the microphone-output signals. The placement of the loudspeakers, microphones, and the seat were similar to that in Figure 2. To simulate the noise source, another loudspeaker was placed in the acoustic chamber. Noise recorded inside a car was played on a tape deck and sent out through this noise source loudspeaker. The reference microphone was placed close to the noise source loudspeaker. To play the music and far-end speech, a Marantz double-cassette deck was used.

Hands-Free Cellular Phone Mode

In this mode, the right loudspeaker was used to broadcast far-end speech, and the left microphone was used to pick up near-end speech. For initial modeling, music was played on the tape deck. During actual conversation, recorded speech was played on the tape deck to generate far-end speech. A person sitting on the seat was the near-end talker. During all this time, the noise loudspeaker was "on" to simulate practical driving conditions. Figure 6 shows the time-domain performance in HFCP mode when only acoustic echo was present. The plot on the top shows the acoustic echo picked up by the primary microphone and the bottom plot shows the residual echo. Figure 7 shows the performance when only background noise was present. Figure 8 shows the performance when both background noise and acoustic echo were present. The plot on the top is the signal picked up by the primary microphone. This signal contains both background noise and acoustic echo. The plot on the bottom shows the residual noise and residual echo. The performance of AEC in this situation, though still acceptable, is not as good as when only acoustic echo was present. This is because of the interference from the residual noise present in the input to the AEC. Figure 9 shows the performance during the double-talk period. The plot on the top has a portion of near-end speech contaminated with the acoustic echo. The plot on the bottom shows the echo-free near-end speech signal. The difference in the performance of HFCP in different situations is more evident from the frequency-domain results shown in Figures 10 - 12. Figure 10 shows a reduction of about 20 dB of acoustic echo. Figure 11 shows the reduction in the dominant periodic components of background noise by 20 - 25 dB. Figure 12 shows the performance of HFCP mode when both background noise and acoustic echo are present. As mentioned earlier, the performance of AEC is degraded when both background noise and acoustic echo are present. The filter orders of 256 and 200, respectively, for ANC and AEC are found to give the desired results. A higher-order filter gives better results, however, at the cost of longer convergence time.

Adaptive-Active Noise Control Mode

In this mode, music was played from the tape deck and sent out through both loudspeakers. The recorded noise was sent out through the noise source loudspeaker. Figure 13 shows the results obtained in AANC for the left channel. Figure 13a shows the noise and music picked up by the left microphone before cancellation. Figure 13b shows the noise-free music signal picked up by the left microphone after cancellation. The signal picked up after cancellation shows about a 20 - 25 dB reduction of the most dominant periodic component of the noise. Other noise components also show some reduction but, because they are almost at the same level as the music signal, it is difficult to see any reduction in them from the figures. By sitting on the seat, one could hear the reduced noise clearly after cancellation. Similar results are plotted for the right microphone in Figure 14a and Figure 14b. The computation power available for processing limited the order of MIS and AANC filters to 40 and 64, respectively. Because of this limitation the AANC filter is not able to respond to higher order harmonics. Fortunately, the higher order harmonics have much lower power—therefore, system performance is still acceptable.

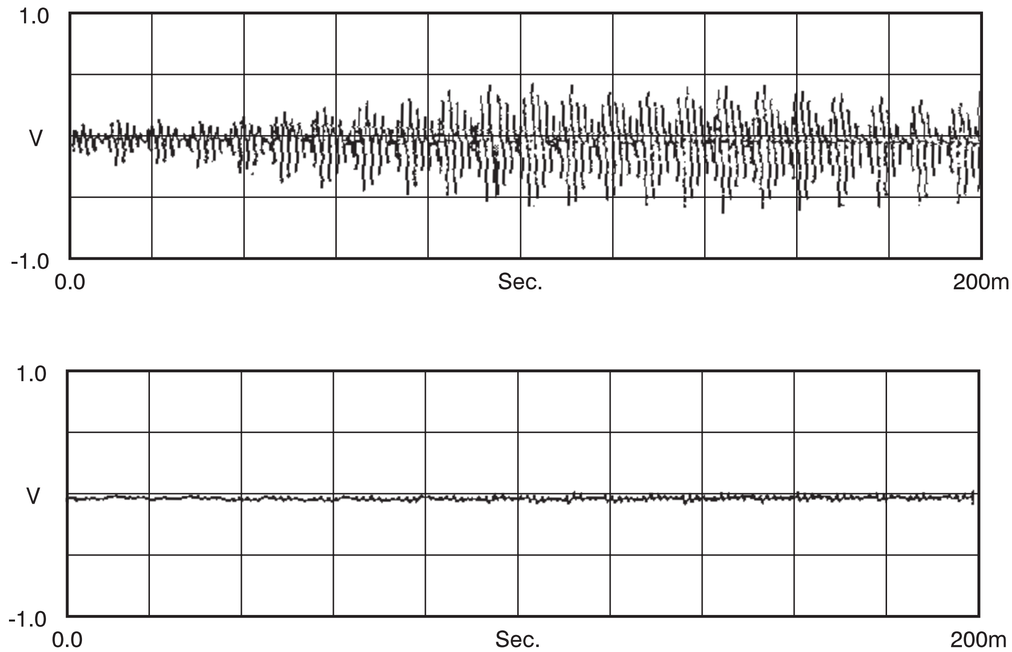

Figure 6. HFCP-Mode Performance (Only Acoustic Echo Present)

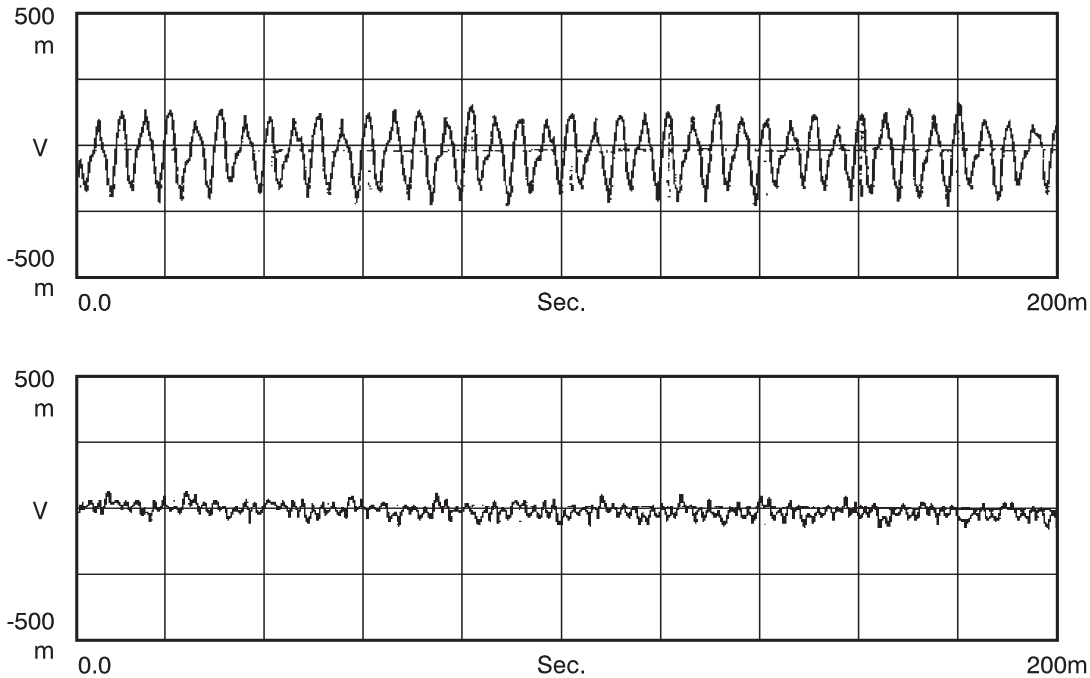

Figure 7. HFCP-Mode Performance (Only Background Noise Present)

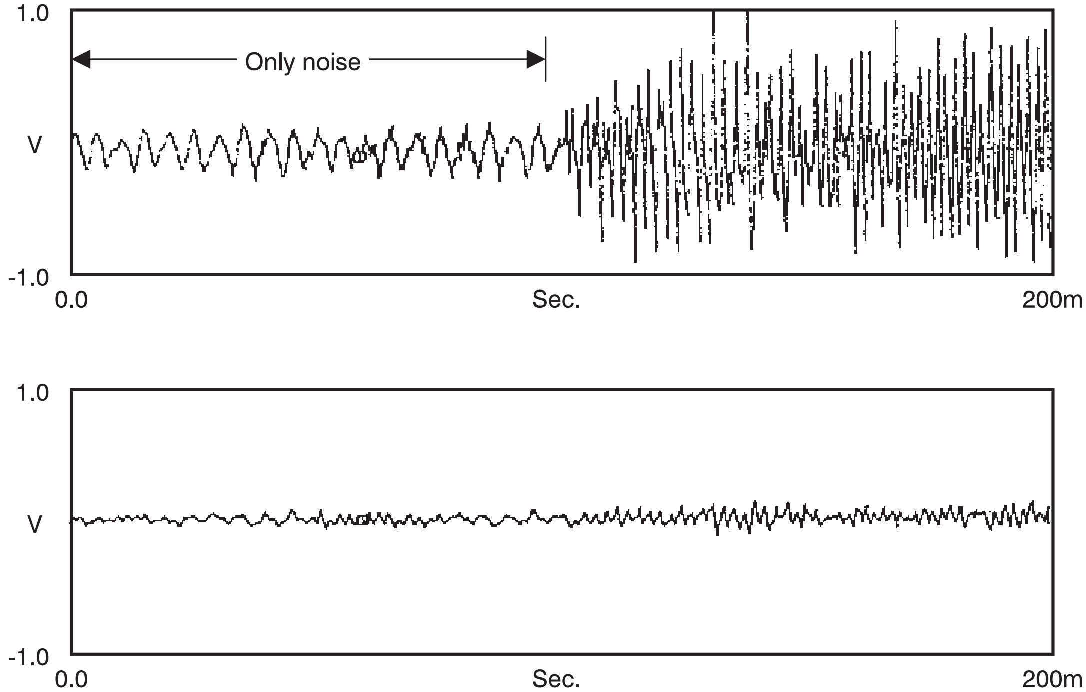

Figure 8. HFCP-Mode Performance (Acoustic Echo and Background Noise Present)

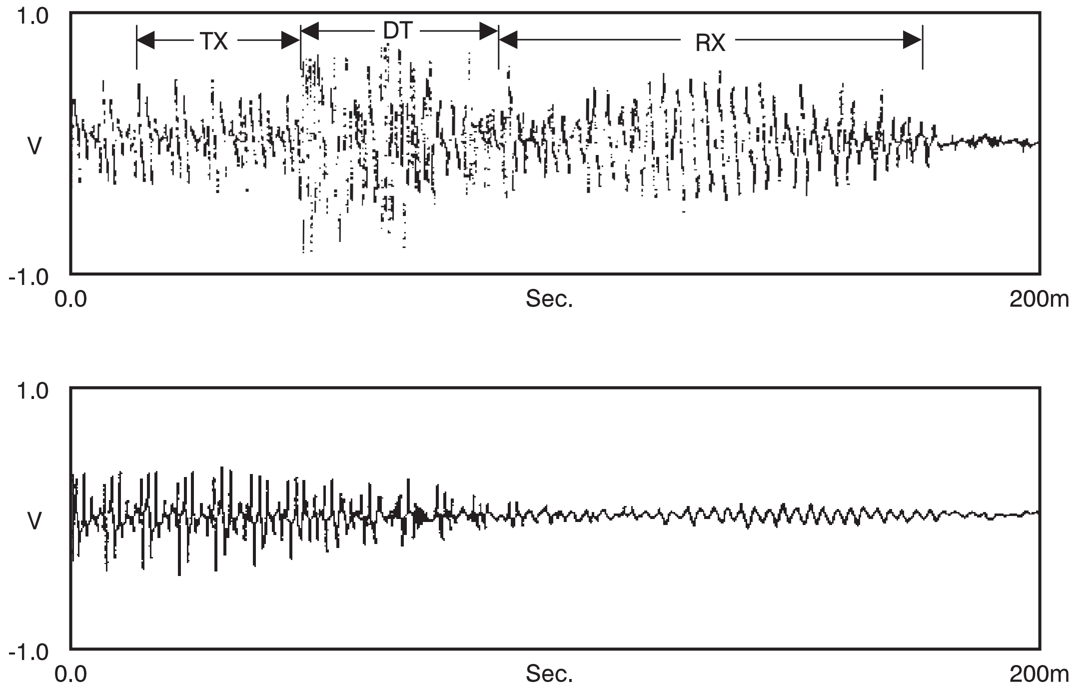

Figure 9. HFCP-Mode Performance (Double Talk)

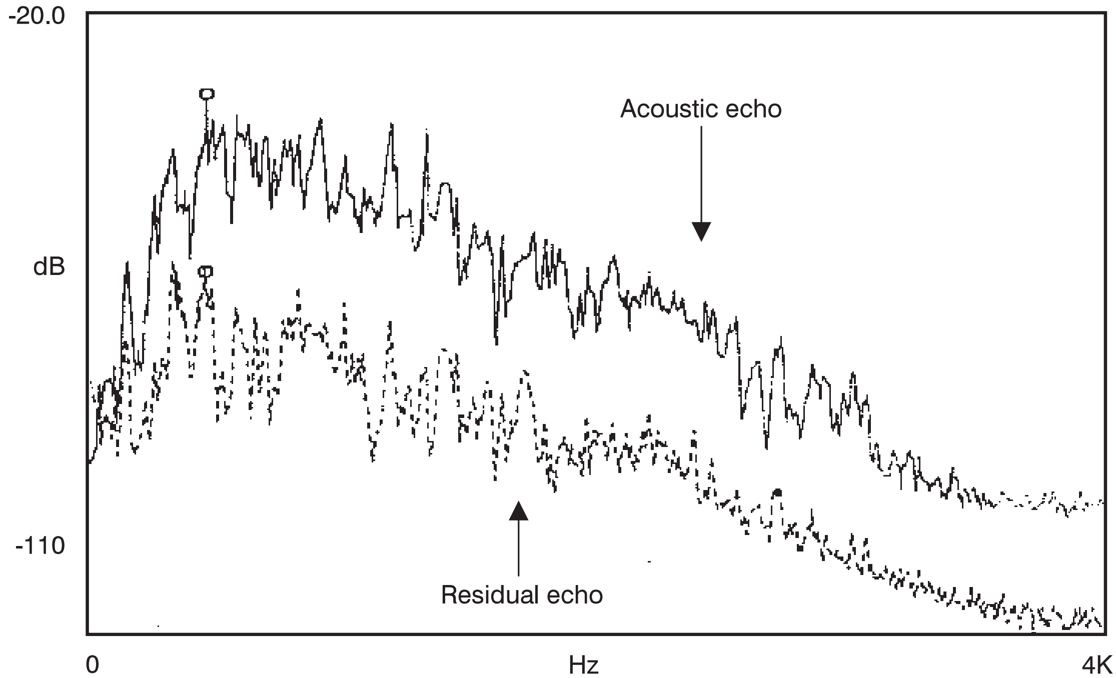

Figure 10. HFCP-Mode Performance (Only Acoustic Echo Present)

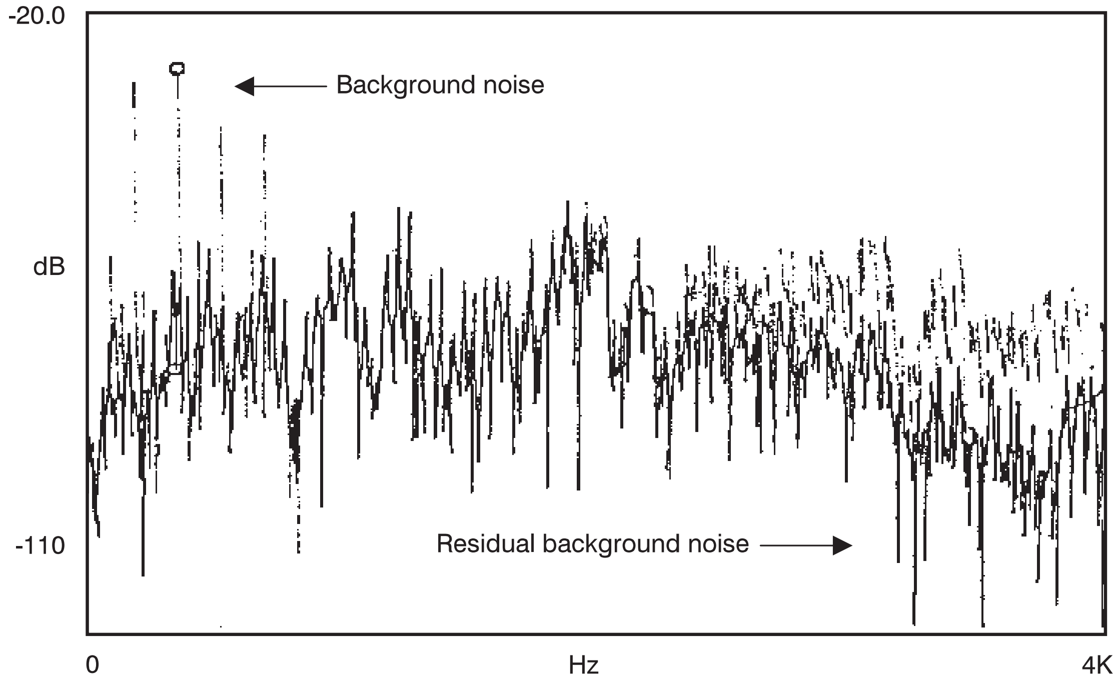

Figure 11. HFCP-Mode Performance (Only Background Noise Present)

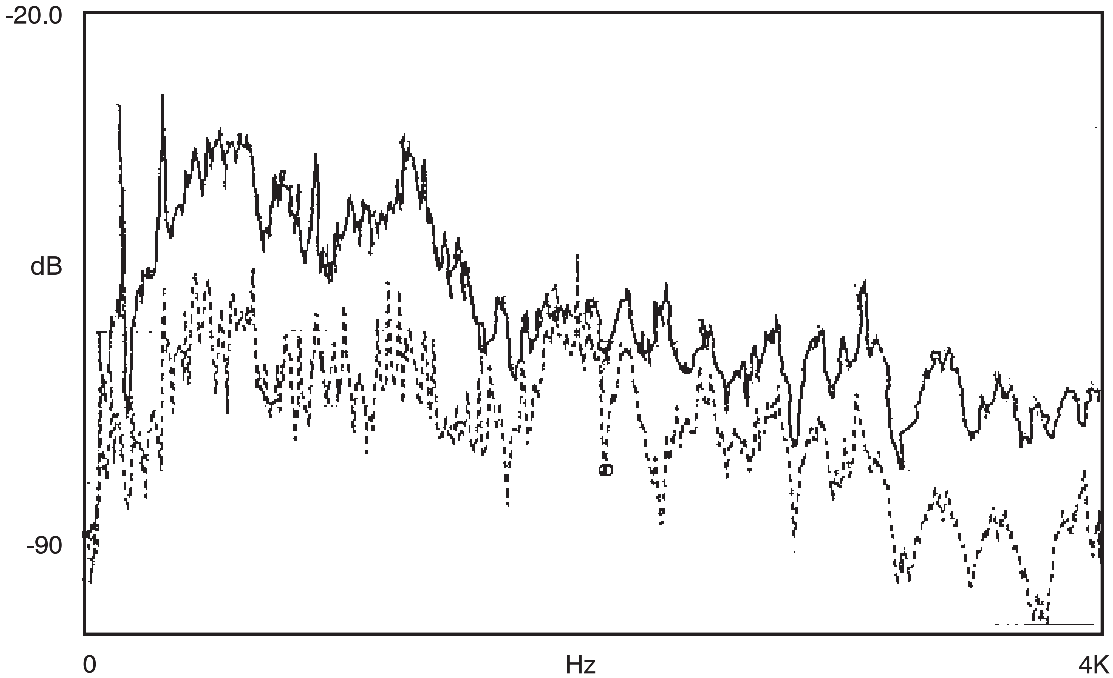

Figure 12. HFCP-Mode Performance (Acoustic Echo and Background Noise Present)

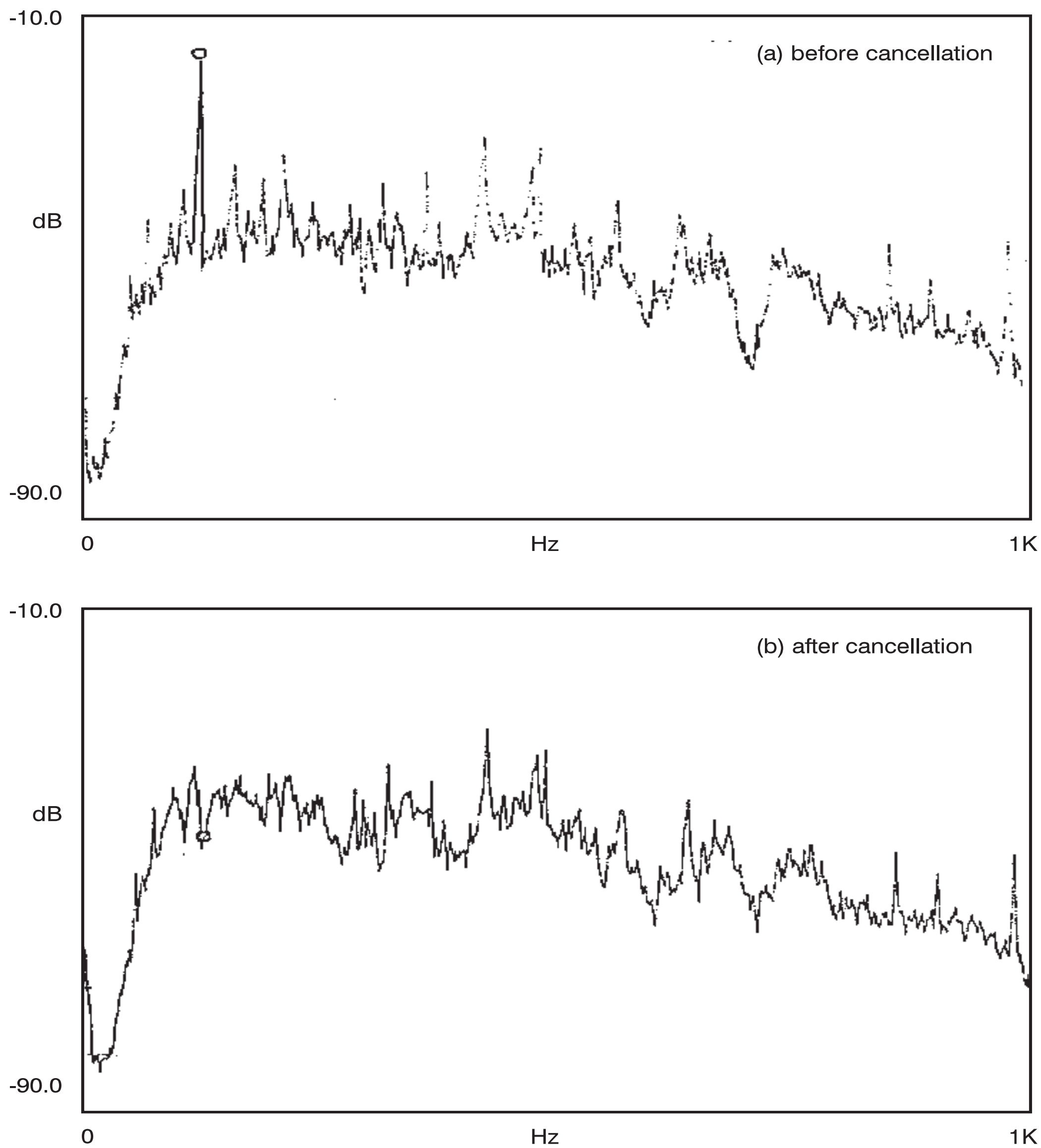

Figure 13. AANC-Mode Performance (Left Channel)

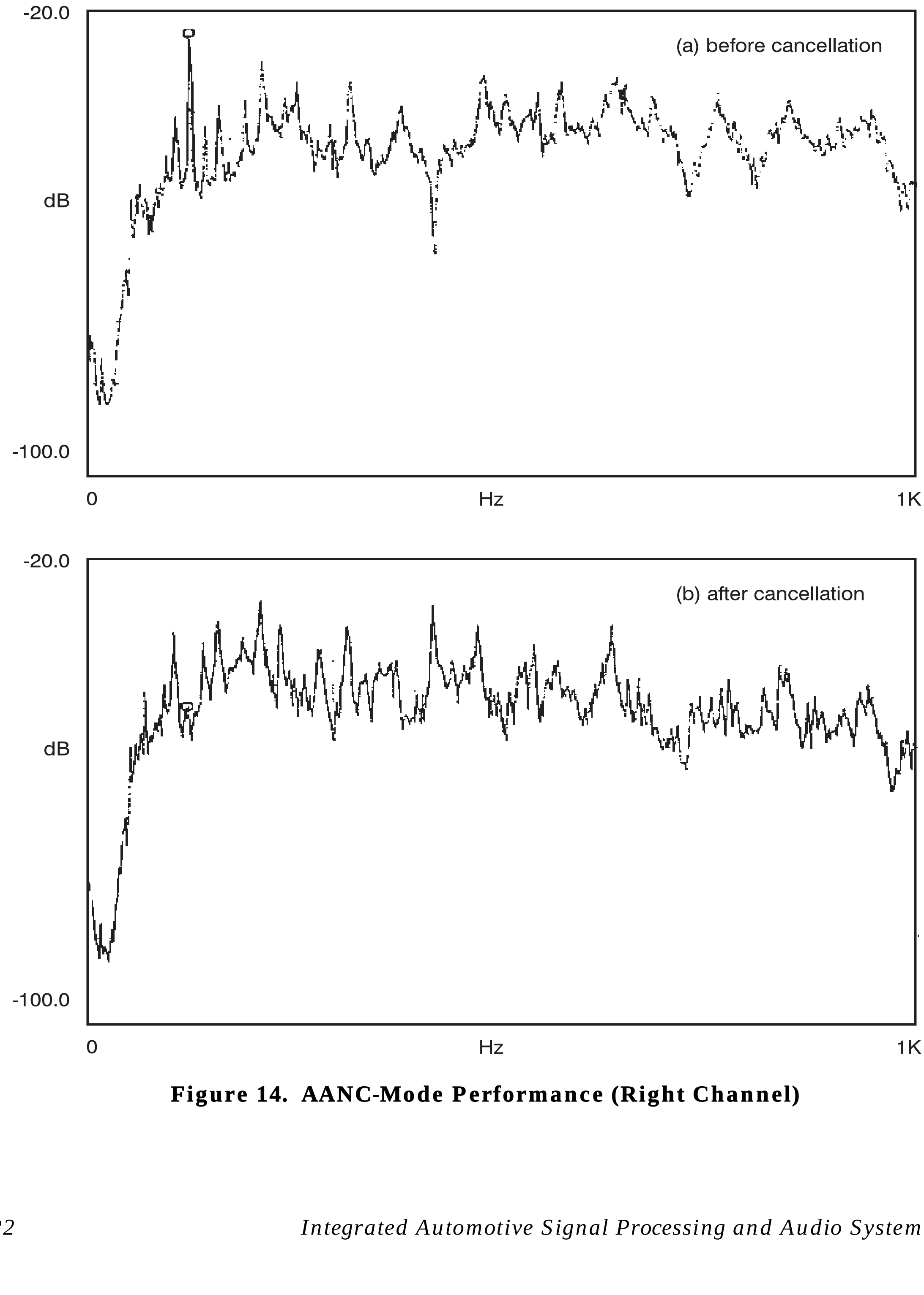

Figure 14. AANC-Mode Performance (Right Channel)

SUMMARY

The integrated automotive signal processing and audio system is designed to allow people to use a hands-free cellular phone without the problem of acoustic echo and ambient noise being transmitted to the far-end talker. The system also reduces the ambient noise inside the car to enhance the music quality from the audio system. Since the design is integrated with the existing car stereo system (i.e., the car stereo system's loudspeakers and amplifiers are used for both hands-free cellular phone and active noise control), the system cost is reduced. Furthermore, the integration of the audio system allows musical interference to be eliminated by using the MIS filters. These MIS filters not only cancel the musical interference to AANC filters, but also model on-line the error paths from canceling loudspeakers to the residual noise pick-up microphones. The music signal from the audio system is used to perform initial modeling for the echo path.

REFERENCES

1. M. M. Sondhi and W. Kellermann, "Adaptive Echo Cancellation for Speech Signal", Chapter 11 in *Advances in Speech Signal Processing,* Edited by S. Furui and M. Sondhi, Dekker, New York, 1992.
2. S. M. Kuo and J. Chen, "Multiple- Microphone Acoustic Echo Cancellation with the Partial Adaptive Processes", *Digital Signal Processing,* Vol. 3, No.1, January 1993, pp. 54–63.
3. S. M. Kuo and Z. Pan, "Adaptive Acoustic Echo Cancellation Microphone", *J. Acoust. Soc. Am.,* Vol. 93, No. 3, March 1993, pp. 1629–1636.
4. B. Widrow, et al, "Adaptive Noise Canceling: Principles and Applications", *Proc. of IEEE*, Vol. 63, No. 12, Dec. 1975, pp.1692–1716.
5. S. J. Elliott, I. M. Sothers, and P. A. Nelson, "A Multiple Error LMS Algorithm and its Application to the Active Noise Control of Sound and Vibration", *IEEE Trans. on ASAP*, Vol. ASSP-35, No. 10, Oct. 1987, pp. 1423–1434.
6. S. M. Kuo and D. R. Morgan, *Active Noise Control Systems*, Wiley, New York, 1996.
7. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
8. S. M. Kuo and B. M. Finn, "A General Multi-Channel Filtered LMS Algorithm for 3-D Active Noise Control Systems", Second Int. Con. on Recent Developments in Air and Structure Borne Sound and Vibration, Auburn AL, May 1992, pp. 345–352.
9. S. M. Kuo and B. M. Finn, "An Integrated Audio and Active Noise Control System", 1993 Int. Symp. Circuits and Systems, Chicago, IL, May 1993, pp. 2529–2532.

APPENDIX A—Flow Charts for System Software

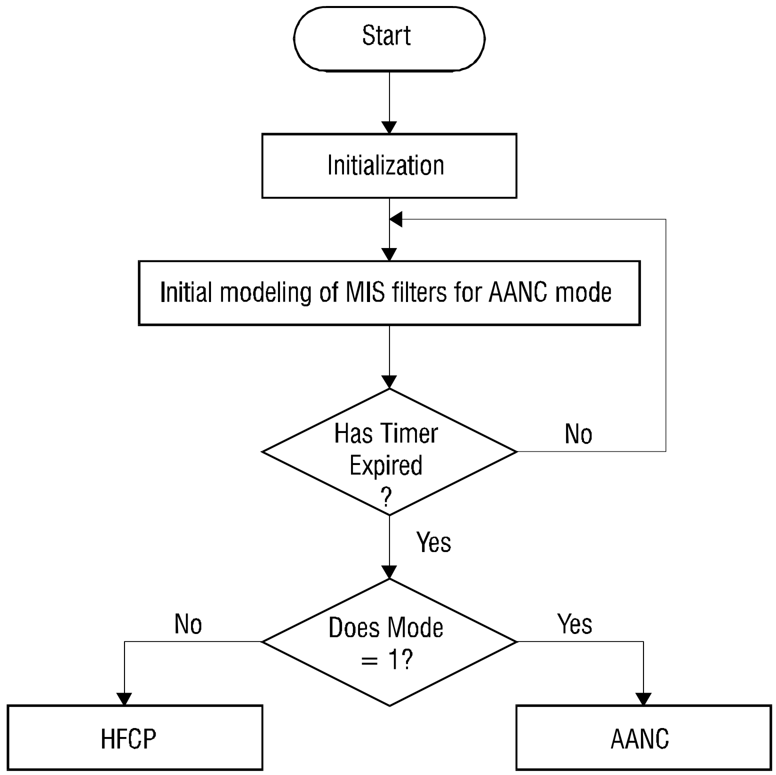

Figure 15. Flow Chart for System Initialization and Initial Modeling of MIS Filters (AANC Mode)

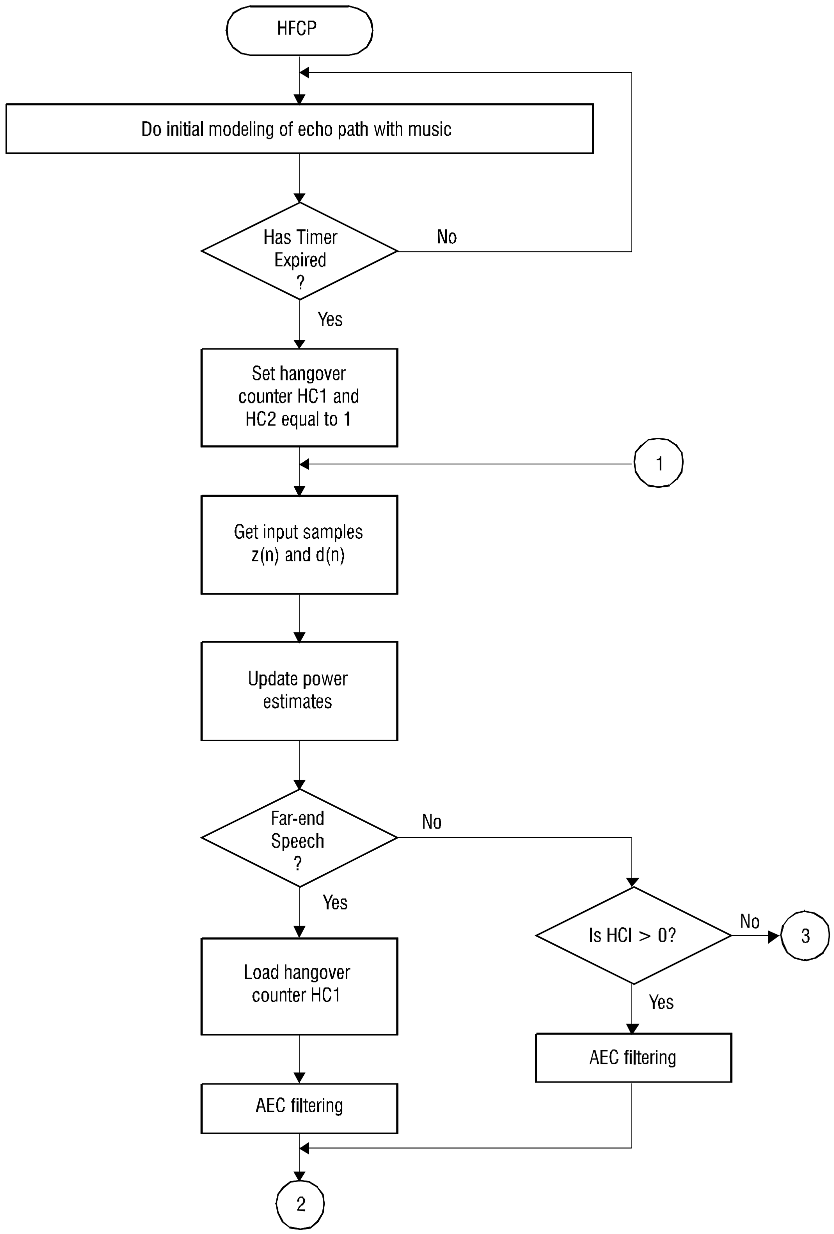

Figure 16. Flow Chart for the System Software (HFCP Mode)

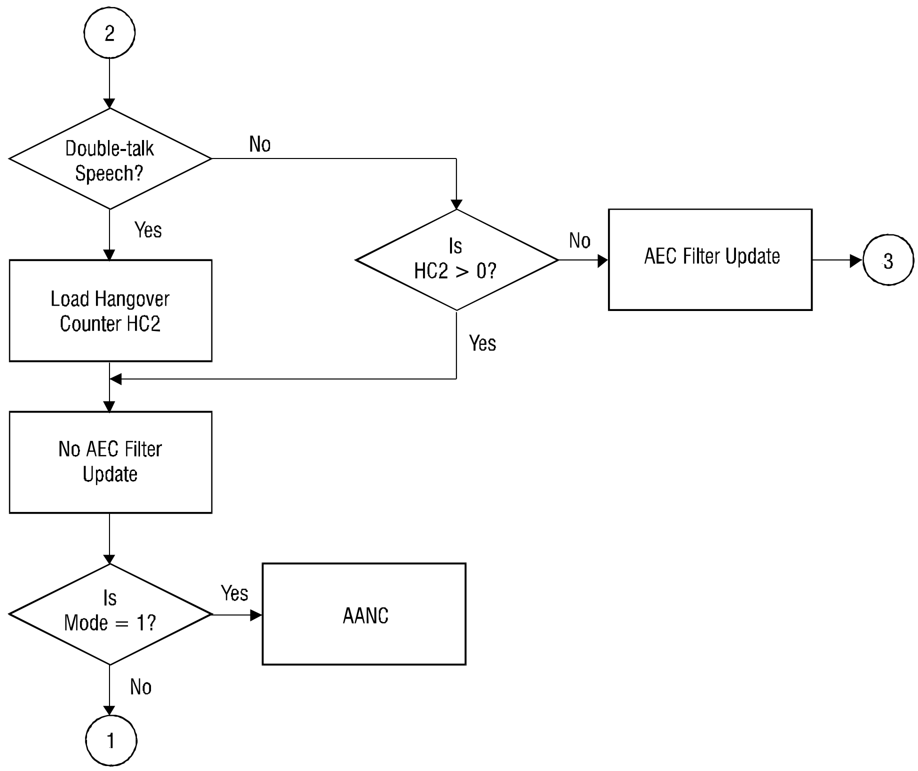

Figure 17. Flow Chart for System Software in AANC Mode (Part 1 of 2)

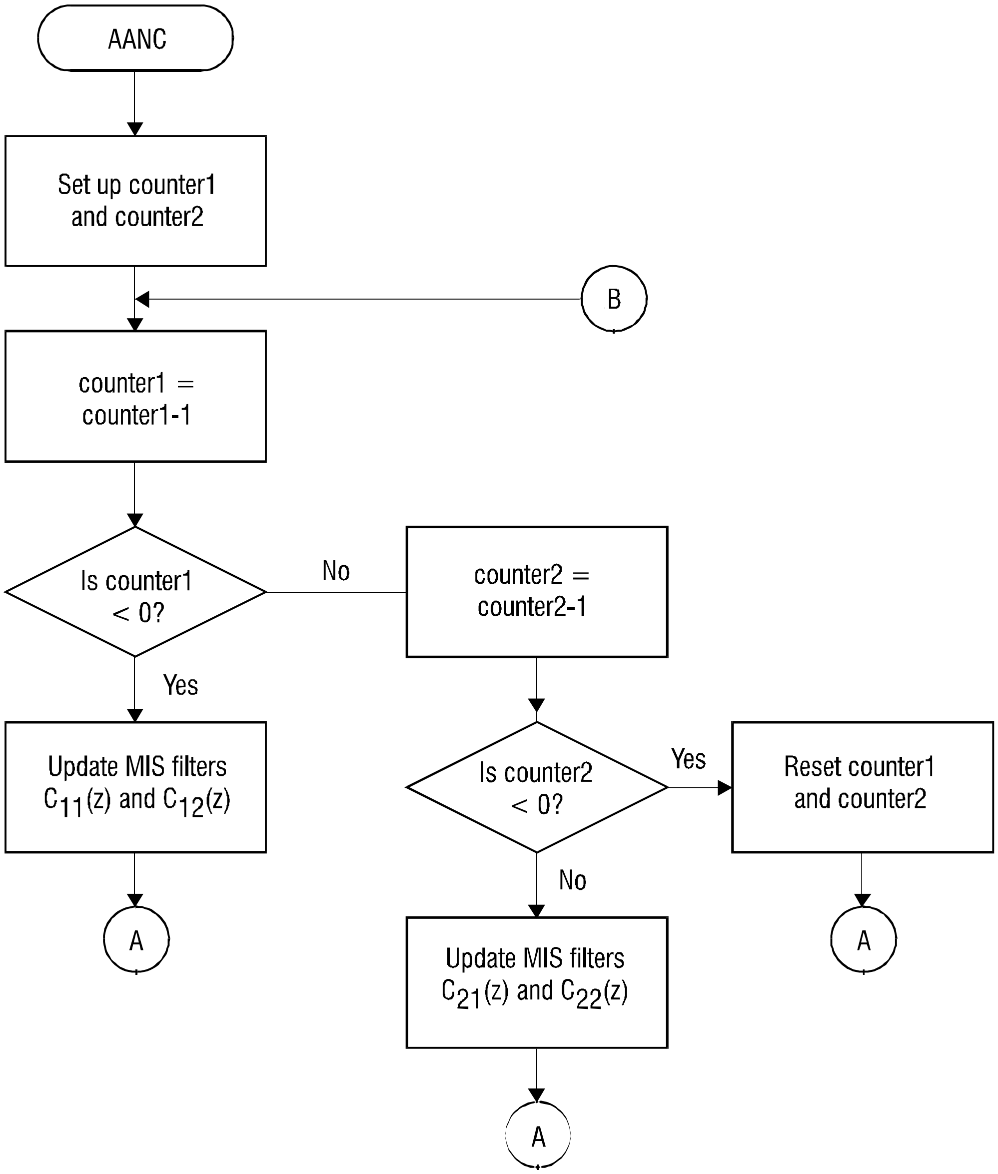

Figure 18. Flow Chart for System Software in AANC Mode (Part 2 of 2)

APPENDIX B—INTSYS.ASM Program

```
***********************************************************************
* Program:  INTSYS.ASM                                                *
*                                                                     *
* Integrated acoustic echo cancellation for hand's-free cellular      *
* phone and active noise control system                               *
*                                                                     *
***********************************************************************
* The following code will implement the software on the TMS320C30     *
* Spectrum Development System along with a 4-channel I/O board       *
*                                                                     *
* The algorithm will detect which mode the Integrated system is       *
* HFCP mode: Do acoustic echo cancellation and adaptive noise         *
*            cancellation                                             *
* AANC mode: Reduce noise inside the car                              *
*                                                                     *
***********************************************************************
*                                                                     *
* The program will follow the following flow of operations            *
*                                                                     *
*   1) Definition of all constants and buffers                        *
*   2) Initialization of TMS320C30 system board and the 4-channel     *
*      I/O board                                                      *
*   3) Clear all buffers                                              *
*   4) Do initial modeling of MIS filters for AANC mode               *
*   5) Check for the mode of the integrated system                    *
*      Mode=1   AANC mode                                             *
*      Mode=0   HFCP mode                                             *
*                                                                     *
***********************************************************************
*                                                                     *
* Following are the subroutines used by the program                   *
*                                                                     *
```

```
*    1) ISR - I/O for both the system and 4-channel board                 *
*    2) FIR - General FIR routine for parameters passed in at             *
*              subroutine header                                          *
*    3) LMS - General LMS routine for parameters passed in at             *
*              subroutine header                                          *
*    4) VSPOWER - Calculates a very short window average power            *
*    5) SPOWER  - Calculates a short window average power                 *
*    6) LPOWER  - Calculates a very long window average power             *
*    7) INVER   - Calculates the inverse of a floating point no.          *
*    8) SPOWDT1 - Calculates the short term energy in signal before echo  *
*                  cancellation                                           *
*    9) SPOWDT2 - Calculates the short term energy in signal after echo   *
*                  cancellation
*
***********************************************************************
* The FIR filter coefficients for downsampling and upsampling are         *
* stored in the file LPKA21.ASC                                           *
*                                                                         *
***********************************************************************
*                                                                         *
       .length 60               ; Define listing file parameters
       .width  132

************INTERRUPT VECTORS AT ADDRESS 0000*****************
        .sect   ".init"

RESET   .word   START           ;Setup RESET vector
INT0    .word   ISR             ;Setup IRQ0 (DSPLINK) interrupt vector
INT1    .word   START           ;Setup all other interrupts to dummy values
INT2    .word   START
INT3    .word   START
XINT0   .word   START
RINT0   .word   START
XINT1   .word   START
RINT1   .word   START
```

```
TINT0   .word   START
TINT1   .word   START
DINT    .word   START

******************DATA AT ADDRESS 2000*****************************
*                                                                           *
*       The program uses the external memory section on the Static RAM      *
*       of the system board                                                 *
*                                                                           *
***********************************************************************
        .data

PRIMCTL .word   00808064h       ; Primary bus control register address
EXPCTL  .word   00808060h       ; Expansion bus control register address
PRIMWD  .word   00000800h       ; control word defining  primary bus status
EXPWD   .word   00000000h       ; control word defining expansion bus status
CACHE   .word   00001800h       ; clear and enable cache

***********************************************************************
*       Buffer locations for the AANC mode                                  *
***********************************************************************

ADDLP   .word   00002500h       ;address of LP anti-aliasing filter coefficients
ADDMRH  .word   00002600h       ;address of music signal for right channel at 8K
ADDMLH  .word   00002700h       ;address of music signal for left channel at 8K
ADDERRH .word   00002800h       ;address of  right MIC error signal at 8K
ADDERLH .word   00002900h       ;address of  left MIC error signal at 8K
ADDRH   .word   00002A00h       ;address of reference noise signal at 8K
ADDCR   .word   00002B00h       ;address of the right channel canceling signal at 2K
ADDCL   .word   00002C00h       ;address of the left channel canceling signal at 2K
ADDY    .word   00003000h       ;address for buffer for generating cancelling signal
ADDYF   .word   00003500h       ;address of the buffer for filtered XLMS
ADDM11  .word   00003800h       ;address of the filtered X signal (RR)
ADDM12  .word   00003A00h       ;address of the filtered X signal (RL)
```

```
ADDM21  .word   00003D00h       ;address of the filtered X signal (LR)
ADDM22  .word   00004000h       ;address of the filtered X signal (LL)
ADDML   .word   00005600h       ;address of buffer for right channel input at 2k
ADDMR   .word   00006000h       ;address of buffer for right channel input at 2k
ADDC11  .word   00007000h       ;address of secondary path filter(RR)
ADDC12  .word   00007300h       ;address of secondary path filter(RL)
ADDC21  .word   00007600h       ;address of secondary path filter(LR)
ADDC22  .word   00007900h       ;address of secondary path filter(LL)
ADDW1   .word   00008000h       ;address of AANC filter W1(z)
ADDW2   .word   00009000h       ;address of AANC filter W2(z)

***********************************************************************
*       Buffer locations for HFCP mode                                *
***********************************************************************

ADDEFIL .word   0000A000h       ;address of the echopath filter
ADDFENS .word   0000B000h       ;address of reference far end speech vector
ADDW    .word   0000C000h       ;address of music signal for initial modeling
ADDR    .word   0000D000h       ;address of reference noise signal buffer
ADDENC  .word   0000E000h       ;address of adaptive noise cancellation filter

***********************************************************************
*       Delay buffers for I/O on system board                         *
***********************************************************************
DELAY1  .word   0000E500h
DELAY2  .word   0000E600h
DELAY3  .word   0000E700h

DELAY   .set    3               ;delay needed on system board I/O


***********************************************************************
*       System board analog interface initialization                  *
***********************************************************************
```

```
ADDCHA  .word   00804000h       ;address of channel A
ADDCHB  .word   00804001h       ;address of channel B

***********************************************************************
*       4-channel I/O board analog interface initialization                 *
***********************************************************************

CH0_A   .word   00800000h       ;address of channel A0
RST_A   .word   00800001h
TIM_A   .word   00800001h
INTS_A  .word   00800003h
INTM_A  .word   00800003h
CH1_A   .word   00800004h       ;address of channel A1
ANAS_A  .word   00800005h
ANAC_A  .word   00800005h
BRDIS_A .word   00800007h
RCTR_A  .word   00800007h

CH0_B   .word   00800008h       ;address of channel B0
RST_B   .word   00800009h
TIM_B   .word   00800009h
INTS_B  .word   0080000Bh
INTM_B  .word   0080000Bh
CH1_B   .word   0080000Ch       ;address of channel B1
ANAS_B  .word   0080000Dh
ANAC_B  .word   0080000Dh
BRDIS_B .word   0080000Fh
RCTR_B  .word   0080000Fh

RRD_A   .word   0000e0000h
INTD_A  .word   000010000h
ACRD_A  .word   000B20000h
TIMD_A  .word   0FA020000h
```

```
UCRD_A  .word   0A4000000h
CRD_A   .word   08DFF0000h

RRD_B   .word   000c00000h
INTD_B  .word   000000000h
ACRD_B  .word   000320000h
UCRD_B  .word   0A0000000h
CRD_B   .word   08DFF0000h

TSTMASK .word   000010000h

***********************************************************************
*       Scaling factors and constants definition  for AANC mode             *
***********************************************************************

S32767  .FLOAT  32767.0         ;scaling factor 32767
S32768  .FLOAT  3.0517578115E-5 ;scaling factor 1/32768
MU      .float  0.001           ;step size
ORDERC  .set    40              ;order of the error path modelling filter
ORDERLP .set    21              ;order of LP anti-aliasing filter
ORDERW  .set    64              ;order of the adaptive cancellation filter
BUFFER  .set    100             ;size of the buffer
DOWNSP  .set    4               ;downsampling factor
COUNT1  .float  30000.0         ;counter of initial modeling in AANC mode
COUNT2  .set    30000           ;counter for alternate updation of error path filters
NPOWL   .FLOAT  0               ;inverse of power of music signal for left channel
NPOWR   .float  0               ;inverse of power of music signal for right channel
NORMU   .float  0               ;inverse of power of reference noise signal
PREF    .float  1.0             ;initial power estimate of reference noise signal
POWERL  .float  1.0             ;initial power estimate of music signal for left
channel
POWERR  .float  1.0             ;initial power estimate of music signal for right
channel

MODE    .float  0
```

```
CHECK   .float  0.5             ;constants for checking the mode of integrated system

*********************************************************************
*       Scaling factors and constants definition  for HFCP mode             *
*********************************************************************


COUNTER .float  30000           ;counter for initial modeling of echo path
HOCNT   .set    8000            ;hangover counter for far end speech
HC      .set    4000            ;hangover counter for double talk
SPTHR   .float  0.001           ;threshold for far-end speech detection
HIEST   .float  10.0            ;highest inverse power
ECHPOW  .float  1.0             ;initial echo power

ECHOFIL .set    200             ;order of echo path modeling filter
ORDERWA .set    256             ;order of adaptive noise cancellation filter
BUFFERA .set    600             ; buffer size

MUN     .FLOAT  0.001           ;Normalised step size
BETA    .FLOAT  0.05            ;factor for power estimate update
MBETA   .FLOAT  0.95            ;1-beta
*
VSBETA  .FLOAT  3.125000E-2     ;beta for very short window of 4ms
VSOMBET .FLOAT  9.687500E-1     ;(1-beta) for very short window
VSOPOWE .FLOAT  1.0             ;old power estimate for very short window
*
SBETA   .FLOAT  7.812500E-3     ;beta for short window of 16ms
SOMBETA .FLOAT  9.920000E-1     ;(1-beta) for short window
SOPOWER .FLOAT  0.0             ;old power for short window
*
LBETA   .FLOAT  6.250000E-5     ;beta for long window of 2 secs
LOMBETA .FLOAT  9.999375E-1     ;(1-beta) for long window
LOPOWER .FLOAT  0.0             ;old power estimate for long window

SOPODT1 .FLOAT  0.0             ;initial power for the residual echo signal
```

```
SOPODT2 .FLOAT  0.0             ;initial power for signal before echo cancellation
THRESDT .FLOAT  3.0             ;ERLE threshold for double talk detection




**********************************************************************
*       Initialize the value at the address to specified values             *
**********************************************************************

REF     .word   0               ;reference noise signal at 2k
REFH    .word   0               ;reference noise signal at 8K
MUSICR  .float  0               ;right channel music input at 2K
MUSICRH .float  0               ;right channel music input at 8K
MUSICL  .float  0               ;left channel music input at 2K
MUSICLH .float  0               ;left channel music input at 8K
OUTR    .float  0               ;right channel music output at 8K/far end speech
broadcast
OUTL    .float  0               ;left channel music output at 8K
CANCR   .float  0               ;right channel canceling signal at 2K
CANCRH  .float  0               ;right channel canceling signal at 8K
CANCL   .float  0               ;left channel canceling signal at 2K
CANCLH  .float  0               ;left channel canceling signal at 8K
W11     .float  0
W12     .float  0
W21     .float  0
W22     .float  0

ERRL    .float  0               ;left MIC input at 2K
ERRLH   .float  0               ;left MIC input at 8K/speech pick up MIC
ERRR    .float  0               ;right MIC input at 2K
ERRRH   .float  0               ;right MIC input at 8K

FARENSP .float  0               ;far end speech from cellular phone
RESECHO .float  0               ;transmit signal from cellular phone
```

```
ENCOUT  .float  0               ;output of adaptive noise cancellation system

CERRR   .float  0               ;internally computed difference signal for right
channel
CERRL   .float  0               ;internally computed difference signal for left channel

RSP     .word   STACK
STACK   .space  100

**********************************************************************
*       TMS320C30 Processor Initialization                           *
*       Program section at address 00D0                              *
**********************************************************************

        .text
START:
        LDP     PRIMCTL         ;Setup data page pointer
        LDI     @RSP,SP         ;Setup the stack pointer

        LDI     @PRIMCTL,AR0    ;Setup the primary bus wait states
        LDI     @PRIMWD,R0      ;in primary bus control register
        STI     R0,*AR0

        LDI     @EXPCTL,AR0     ;Setup the expansion bus wait states
        LDI     @EXPWD,R0       ;in expansion bus control register
        STI     R0,*AR0

        LDI     @CACHE,ST       ;clear and enable cache by setting cache
                                ;control bits in the CPU control register

**********************************************************************
*       Configure and initialize 4-channel I/O board                 *
**********************************************************************
                                ;Configure site A
        LDI     @RST_A,AR0      ;Dummy read to reset
```

```
        LDI     *AR0,R0
        LDI     @RCTR_A,AR0     ;Setup the route register
        LDI     @RRD_A,R0
        STI     R0,*AR0
        LDI     @INTM_A,AR0     ;Setup the interrupt mask register
        LDI     @INTD_A,R0
        STI     R0,*AR0
        LDI     @ANAC_A,AR0     ;Setup the AMELIA ctrl register
        LDI     @ACRD_A,R0
        STI     R0,*AR0
        LDI     @TIM_A,AR0      ;Setup the timer
        LDI     @TIMD_A,R0
        STI     R0,*AR0
        LDI     @RCTR_A,AR0     ;Setup the user ctrl register
        LDI     @UCRD_A,R0
        STI     R0,*AR0

                                ;Configure site B
        LDI     @RST_B,AR0      ;Dummy read to reset
        LDI     *AR0,R0
        LDI     @RCTR_B,AR0     ;Setup the route register
        LDI     @RRD_B,R0
        STI     R0,*AR0
        LDI     @INTM_B,AR0     ;Setup the interrupt mask
        LDI     @INTD_B,R0
        STI     R0,*AR0
        LDI     @ANAC_B,AR0     ;Setup the AMELIA ctrl register
        LDI     @ACRD_B,R0
        STI     R0,*AR0
        LDI     @RCTR_B,AR0     ;Setup the user ctrl register
        LDI     @UCRD_B,R0
        STI     R0,*AR0

***********************************************************************
```

```
*       All the arrays being used are cleared i.e. they are initialized to 0*
***********************************************************************

        LDI     BUFFER-1,RC
        LDI     BUFFER,BK
        LDI     @ADDY,AR0
        LDI     @ADDYF,AR1
        LDI     @ADDMR,AR2
        LDI     @ADDML,AR3
        LDI     @ADDM11,AR4
        LDI     @ADDM12,AR5
        LDI     @ADDM21,AR6
        LDI     @ADDM22,AR7
        LDF     0.0,R0
        RPTB    CLR
        STF     R0,*AR0++(1)%
        STF     R0,*AR1++(1)%
        STF     R0,*AR2++(1)%
        STF     R0,*AR3++(1)%
        STF     R0,*AR4++(1)%
        STF     R0,*AR5++(1)%
        STF     R0,*AR6++(1)%
CLR     STF     R0,*AR7++(1)%

        LDI     BUFFER-1,RC
        LDI     BUFFER,BK
        LDI     @ADDC11,AR0
        LDI     @ADDC12,AR1
        LDI     @ADDC21,AR2
        LDI     @ADDC22,AR3
        LDI     @ADDW1,AR4
        LDI     @ADDW2,AR5
        RPTB    CLR1
        STF     R0,*AR0++(1)%
```

```
        STF     R0,*AR1++(1)%
        STF     R0,*AR2++(1)%
        STF     R0,*AR3++(1)%
        STF     R0,*AR4++(1)%
CLR1    STF     R0,*AR5++(1)%

        LDI     BUFFER-1,RC
        LDI     BUFFER,BK
        LDI     @ADDMRH,AR0
        LDI     @ADDMLH,AR1
        LDI     @ADDERRH,AR2
        LDI     @ADDERLH,AR3
        LDI     @ADDRH,AR4
        LDI     @ADDCR,AR5
        LDI     @ADDCL,AR6
        LDF     0.0,R0
        RPTB    CLR2
        STF     R0,*AR0++(1)%
        STF     R0,*AR1++(1)%
        STF     R0,*AR2++(1)%
        STF     R0,*AR3++(1)%
        STF     R0,*AR4++(1)%
        STF     R0,*AR5++(1)%
CLR2    STF     R0,*AR6++(1)%

        LDI     BUFFERA-1,RC
        LDI     BUFFERA,BK
        LDI     @ADDEFIL,AR0
        LDI     @ADDFENS,AR1
        LDI     @ADDW,AR2
        LDI     @ADDR,AR3
        LDI     @ADDENC,AR4
        LDF     0.0,R0
        RPTB    CLRA
```

```
        STF     R0,*AR0++(1)%
        STF     R0,*AR2++(1)%
        STF     R0,*AR3++(1)%
        STF     R0,*AR4++(1)%
CLRA    STF     R0,*AR1++(1)%

        OR      1h,IE           ;Enable the IRQ0
        OR      2000h,ST

        LDI     @RCTR_A,AR0     ;Setup the configuration register site A
        LDI     @CRD_A,R0
        STI     R0,*AR0

        LDI     @RCTR_B,AR0     ;Setup the configuration register site B
        LDI     @CRD_B,R0
        STI     R0,*AR0

        LDF     @COUNT1,R6      ;load counter for initial modeling in R6
        LDI     DOWNSP,R5       ;load downsample counter in R5
        LDF     0.0,R7          ;load dummy output in R7



***********************************************************************
*       Initial on-line modeling for right channel                    *
***********************************************************************

TRAIN1  IDLE

***********************************************************************
* DOWN SAMPLE THE RIGHT CHANNEL INPUT                                 *
***********************************************************************
```

```
        LDI     ORDERLP,BK      ;circular buffer length set to the order of LPF
        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDI     @ADDMRH,AR1     ;load address of right channel music signal at 8K in
AR1
        LDF     @MUSICRH,R0     ;load latest music sample in R0
        STF     R0,@OUTR        ;store latest music in OUTR to send out thru' right
speaker
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest music sample
        STI     AR1,@ADDMRH     ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@MUSICR      ;downsampled right channel input in MUSICR

***********************************************************************
* DOWN SAMPLE THE RIGHT MIC INPUT                                        *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDERRH,AR1    ;load address of right channel error at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @ERRRH,R0       ;load latest error (right) sample in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest error (right) signal
        STI     AR1,@ADDERRH    ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRR        ;downsampled right MIC input in ERRR

***********************************************************************
* DOWN SAMPLE THE LEFT MIC INPUT                                         *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDERLH,AR1    ;load address of left channel error at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
```

```
        LDF     @ERRLH,R0       ;load latest error (left) sample in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest error (left) signal
        STI     AR1,@ADDERLH    ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRL        ;downsampled left MIC input in ERRL

***********************************************************************
* DOWN SAMPLE THE REFERENCE SIGNAL INPUT                              *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDRH,AR1      ;load address of reference signal at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @REFH,R0        ;load latest reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest reference signal
        STI     AR1,@ADDRH      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@REF         ;downsampled reference signal in REF

***********************************************************************
* UPSAMPLE THE RIGHT CHANNEL CANCELLING SIGNAL                         *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDCR,AR1      ;load address of right canceling signal at 2K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @CANCR,R0       ;load the latest right canceling signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest right canceling signal
        STI     AR1,@ADDCR      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    0.25,R0         ;multiply result of FIR by scaling factor of 1/4
        STF     R0,@CANCRH      ;upsampled right canceling output in CANCRH
```

```
        LDF     0.0,R7
        STF     R7,@OUTL
        STF     R7,@CANCLH

        SUBI    1,R5            ;check for downsampling count
        BNZ     TRAIN1



***********************************************************************
* PROCESSING LOOP AT LOWER SAMPLING RATE                              *
***********************************************************************

        LDI     ORDERC,BK       ;circular buffer length set to the order of MIS filters
        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC11,AR0     ;AR0 points to error path (RR) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @MUSICR,R0      ;load the latest right music signal at 2K in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest right music signal at
2K
        STI     AR1,@ADDMR      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering

***********************************************************************
* CALCULATING THE ERROR FOR PATH R-R   AND UPDATE C11(z)              *
***********************************************************************

        LDF     @ERRR,R4        ;get downsampled right MIC error signal in R4
        SUBF    R0,R4           ;calculate the difference
        STF     R4,@CERRR       ;store the internally computed difference in CERRR
        MPYF    @MU,R4          ;multiply the difference with normalized step size

        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC11,AR0     ;AR0 points to error path (RR) estimate coefficients
```

```
        CALL    LMS             ;call subroutine to do least mean square update

        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC12,AR0     ;AR0 points to error path (RL) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        CALL    FIR             ;subroutine to do FIR filtering

***********************************************************************
* CALCULATING THE ERROR FOR PATH 1-2 AND UPDATE C12(z)                     *
***********************************************************************

        LDF     @ERRL,R4        ;get downsampled left MIC error signal in R4
        SUBF    R0,R4           ;calculate the difference
        STF     R4,@CERRL       ;store the internally computed difference in CERRL
        MPYF    @MU,R4          ;multiply the difference with normalized step size

        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC12,AR0     ;AR0 points to error path (RL) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update

***********************************************************************
* FILTERED REFERENCE SIGNAL GENERATION                                    *
***********************************************************************
        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDYF      ;address update
        LDI     @ADDC11,AR0     ;AR0 points to error path (RR) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering
        LDF     R0,R3           ;store filtered ref. signal (RR) in R3

        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
```

```
        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDC12,AR0     ;AR0 points to error path (RL) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering

***********************************************************************
* UPDATE REFERENCE SIGNAL BUFFER FOR x'11(n) and x'12(n)                *
***********************************************************************

        LDI     @ADDM11,AR1     ;load address of filtered-X signal (RR) buffer in AR1
        LDI     ORDERW,BK       ;circular buffer length set to order of AANC filters
        STF     R3,*AR1++(1)%   ;save filtered-X signal (RR) value
        STI     AR1,@ADDM11     ;address update

        LDI     @ADDM12,AR2     ;load address of filtered-X signal (RL) buffer in AR1
        STF     R0,*AR2++(1)%   ;save filtered-X signal (RL) value
        STI     AR2,@ADDM12     ;address update

***********************************************************************
* UPDATE W1(z)                                                        *
***********************************************************************
        LDF     @CERRR,R0       ;get error signal CERRR in R0
        MPYF    @MU,R0          ;multiply R0 with normalized step size
        LDI     @ADDW1,AR0      ;load address of W1 filter coefficients in AR0
        LDI     ORDERW-1,RC     ;set repeat counter
        RPTB    LMS_W1          ;repeat block up to LMS_W1
        MPYF3   *AR1++(1)%,R0,R3;calculate the update factor
        LDF     *AR0,R4         ;get the old coefficient value in R4
        SUBF    R3,R4           ;calculate new coefficient value
LMS_W1  STF     R4,*AR0++(1)    ;update the filter W1

***********************************************************************
* GENERATE RIGHT CHANNEL CANCELLING SIGNAL                            *
***********************************************************************
        LDI     ORDERW-2,RC     ;set repeat counter
```

```
        LDI     @ADDY,AR1       ;load address of reference signal buffer in AR1
        LDI     @ADDW1,AR0      ;load address of W1 filter coefficients in AR0
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDY       ;address update
        CALL    FIR             ;subroutine to do FIR filtering
        STF     R0,@CANCR       ;store right canceling signal at 2K in CANCR

        LDI     DOWNSP,R5       ;reload the downsample count in R5
        SUBF    0.25,R6         ;check for training counter
        BNZ     TRAIN1



***********************************************************************
***********************************************************************
***********************************************************************
***********************************************************************


        LDF     @COUNT1,R6      ;load the training counter in R6
        LDI     DOWNSP,R5       ;load the downsample count in R5

***********************************************************************
*       Initial modeling for left channel                             *
***********************************************************************

TRAIN2  IDLE

***********************************************************************
* DOWN SAMPLE THE LEFT CHANNEL INPUT                                  *
***********************************************************************
```

```
        LDI     ORDERLP,BK       ;circular buffer length set to the order of LPF
        LDI     ORDERLP-2,RC     ;loop counter set
        LDI     @ADDLP,AR0       ;load address of LPF coefficients in AR0
        LDI     @ADDMLH,AR1      ;load address of left channel music signal at 8K in AR1
        LDF     @MUSICLH,R0      ;load latest music sample in R0
        STF     R0,@OUTL         ;store latest music in OUTL to send out thru' left
speaker
        STF     R0,*AR1++(1)%    ;AR1 modified to point to oldest music sample
        STI     AR1,@ADDMLH      ;address updated
        CALL    FIR              ;subroutine to do FIR filtering
        MPYF    4.0,R0           ;multiply result of FIR by scaling factor of 4
        STF     R0,@MUSICL       ;downsampled left channel input in MUSICR

***********************************************************************
* DOWN SAMPLE THE RIGHT MIC INPUT                                         *
***********************************************************************

        LDI     ORDERLP-2,RC     ;loop counter set
        LDI     @ADDERRH,AR1     ;load address of right channel error at 8K in AR1
        LDI     @ADDLP,AR0       ;load address of LPF coefficients in AR0
        LDF     @ERRRH,R0        ;load latest error (right) sample in R0
        STF     R0,*AR1++(1)%    ;AR1 modified to point to oldest error (right) signal
        STI     AR1,@ADDERRH     ;address updated
        CALL    FIR              ;subroutine to do FIR filtering
        MPYF    4.0,R0           ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRR         ;downsampled right MIC input in ERRR

***********************************************************************
* DOWN SAMPLE THE LEFT MIC INPUT                                         *
***********************************************************************

        LDI     ORDERLP-2,RC     ;loop counter set
        LDI     @ADDERLH,AR1     ;load address of left channel error at 8K in AR1
        LDI     @ADDLP,AR0       ;load address of LPF coefficients in AR0
        LDF     @ERRLH,R0        ;load latest error (left) sample in R0
```

```
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest error (left) signal
        STI     AR1,@ADDERLH    ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRL        ;downsampled left MIC input in ERRL

***********************************************************************
* DOWN SAMPLE THE REFERENCE SIGNAL INPUT                                *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDRH,AR1      ;load address of reference signal at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @REFH,R0        ;load latest reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest reference signal
        STI     AR1,@ADDRH      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@REF         ;downsampled reference signal in REF

***********************************************************************
* UPSAMPLE THE LEFT CHANNEL CANCELLING SIGNAL                          *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDCL,AR1      ;load address of left canceling signal at 2K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @CANCL,R0       ;load the latest left canceling signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest left canceling signal
        STI     AR1,@ADDCL      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    0.25,R0         ;multiply result of FIR by scaling factor of 1/4
        STF     R0,@CANCLH      ;upsampled right canceling output in CANCLH
        LDF     0.0,R7
```

```
        STF     R7,@OUTR
        STF     R7,@CANCRH

        SUBI    1,R5            ;check for downsampling count
        BNZ     TRAIN2

***********************************************************************
* PROCESSING LOOP AT LOWER SAMPLING RATE                                  *
***********************************************************************

        LDI     ORDERC,BK       ;circular buffer length set to the order of MIS filters
        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC22,AR0     ;AR0 points to error path (LL) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @MUSICL,R0      ;load the latest left music signal at 2K in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest left music signal at
2K
        STI     AR1,@ADDML      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering



***********************************************************************
* CALCULATING THE ERROR FOR PATH L-L   AND UPDATE C22(z)                  *
***********************************************************************

        LDF     @ERRL,R4        ;get downsampled left MIC error signal in R4
        SUBF    R0,R4           ;calculate the difference
        STF     R4,@CERRL       ;store the internally computed difference in CERRL
        MPYF    @MU,R4          ;multiply the difference with normalized step size

        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC22,AR0     ;AR0 points to error path (LL) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update
```

```
        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        CALL    FIR             ;subroutine to do FIR filtering




***********************************************************************
* CALCULATING THE ERROR FOR PATH L-R AND UPDATE C12(z)                    *
***********************************************************************

        LDF     @ERRR,R4        ;get downsampled right MIC error signal in R4
        SUBF    R0,R4           ;calculate the difference
        STF     R4,@CERRR       ;store the internally computed difference in CERRR
        MPYF    @MU,R4          ;multiply the difference with normalized step size

        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update

***********************************************************************
* FILTERED REFERENCE SIGNAL GENERATION                                    *
***********************************************************************
        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDYF      ;address update
        LDI     @ADDC22,AR0     ;AR0 points to error path (LL) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering
        LDF     R0,R3           ;store filtered ref. signal (LL) in R3
```

```
        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering

***********************************************************************
* UPDATE REFERENCE SIGNAL BUFFER FOR x’21(n) and x’22(n)                *
***********************************************************************

        LDI     @ADDM22,AR1     ;load address of filtered-X signal (LL) buffer in AR1
        LDI     ORDERW,BK       ;circular buffer length set to order of AANC filters
        STF     R3,*AR1++(1)%   ;save filtered-X signal (LL) value
        STI     AR1,@ADDM22     ;address update

        LDI     @ADDM21,AR2     ;load address of filtered-X signal (LR) buffer in AR1
        STF     R0,*AR2++(1)%   ;save filtered-X signal (LR) value
        STI     AR2,@ADDM21     ;address update

***********************************************************************
* UPDATE W2(z)                                                        *
***********************************************************************
        LDF     @CERRL,R0       ;get error signal CERRL in R0
        MPYF    @MU,R0          ;multiply R0 with normalized step size
        LDI     @ADDW2,AR0      ;load address of W2 filter coefficients in AR0
        LDI     ORDERW-1,RC     ;set repeat counter
        RPTB    LMS_W2          ;repeat block up to LMS_W2
        MPYF3   *AR1++(1)%,R0,R3;calculate the update factor
        LDF     *AR0,R4         ;get the old coefficient value in R4
        SUBF    R3,R4           ;calculate new coefficient value
LMS_W2  STF     R4,*AR0++(1)    ;update the filter W2

***********************************************************************
* GENERATE LEFT CHANNEL CANCELLING SIGNAL                            *
```

```
*********************************************************************
        LDI     ORDERW-2,RC     ;set repeat counter
        LDI     @ADDY,AR1       ;load address of reference signal buffer in AR1
        LDI     @ADDW2,AR0      ;load address of W2 filter coefficients in AR0
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDY       ;address update
        CALL    FIR             ;subroutine to do FIR filtering
        STF     R0,@CANCL       ;store right canceling signal at 2K in CANCL

        LDI     DOWNSP,R5       ;reload the downsample count in R5
        SUBF    0.25,R6         ;check for training counter
        BNZ     TRAIN2

*********************************************************************
*********************************************************************
*********************************************************************
* AFTER INITIAL MODELLING WITH ONE CHANNEL OFF ....AT THIS TIME BOTH    *
* THE CHANNELS ARE TURNED ON                                            *
* Signals to be downsampled     1. Right channel input                  *
*                               2. Left channel input                   *
*                               3. Right MIC input                      *
*                               4. Left MIC input                       *
*                               5. Reference signal                     *
* Signals to be upsampled       1. Right channel Cancelling signal      *
*                               2. Left channel Cancelling signal       *
*********************************************************************
*********************************************************************
AANC    LDI     COUNT2,R6       ;load counter in R6 and R7 for alternate
        LDI     COUNT2,R7       ;modeling of error paths
        LDI     DOWNSP,R5       ;load the downsample count in R5
        LDF     0.0,R0
        STF     R0,@OUTR
        STF     R0,@RESECHO     ;setting outputs in HFCP mode to zero
```

```
ANC     IDLE
**********************************************************************
* DOWN SAMPLE THE RIGHT CHANNEL INPUT                                     *
**********************************************************************

        LDI     ORDERLP,BK      ;circular buffer length set to the order of LPF
        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDI     @ADDMRH,AR1     ;load address of right channel music signal at 8K in
AR1
        LDF     @MUSICRH,R0     ;load latest music sample in R0
        STF     R0,@OUTR        ;store latest music in OUTR to send out thru’ right
speaker
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest music sample
        STI     AR1,@ADDMRH     ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@MUSICR      ;downsampled right channel input in MUSICR


**********************************************************************
* DOWN SAMPLE THE LEFT CHANNEL INPUT                                      *
**********************************************************************
        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDI     @ADDMLH,AR1     ;load address of left channel music signal at 8K in AR1
        LDF     @MUSICLH,R0     ;load latest music sample in R0
        STF     R0,@OUTL        ;store latest music in OUTL to send out thru’ left
speaker
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest music sample
        STI     AR1,@ADDMLH     ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@MUSICL      ;downsampled left channel input in MUSICR
```

```
**********************************************************************
* DOWN SAMPLE THE RIGHT MIC INPUT                                    *
**********************************************************************
        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDERRH,AR1    ;load address of right channel error at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @ERRRH,R0       ;load latest error (right) sample in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest error (right) signal
        STI     AR1,@ADDERRH    ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRR        ;downsampled right MIC input in ERRR

**********************************************************************
* DOWN SAMPLE THE LEFT MIC INPUT                                     *
**********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDERLH,AR1    ;load address of left channel error at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @ERRLH,R0       ;load latest error (left) sample in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest error (left) signal
        STI     AR1,@ADDERLH    ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@ERRL        ;downsampled left MIC input in ERRL

**********************************************************************
* DOWN SAMPLE THE REFERENCE SIGNAL INPUT                             *
**********************************************************************
```

```
        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDRH,AR1      ;load address of reference signal at 8K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @REFH,R0        ;load latest reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest reference signal
        STI     AR1,@ADDRH      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    4.0,R0          ;multiply result of FIR by scaling factor of 4
        STF     R0,@REF         ;downsampled reference signal in REF


***********************************************************************
* UPSAMPLE THE RIGHT CHANNEL CANCELLING SIGNAL                          *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDCR,AR1      ;load address of right canceling signal at 2K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @CANCR,R0       ;load the latest right canceling signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest right canceling signal
        STI     AR1,@ADDCR      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    0.25,R0         ;multiply result of FIR by scaling factor of 1/4
        STF     R0,@CANCRH      ;upsampled right canceling output in CANCRH

***********************************************************************
* UPSAMPLE THE LEFT CHANNEL CANCELLING SIGNAL                          *
***********************************************************************

        LDI     ORDERLP-2,RC    ;loop counter set
        LDI     @ADDCL,AR1      ;load address of left canceling signal at 2K in AR1
        LDI     @ADDLP,AR0      ;load address of LPF coefficients in AR0
        LDF     @CANCL,R0       ;load the latest left canceling signal in R0
```

```
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest left canceling signal
        STI     AR1,@ADDCL      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering
        MPYF    0.25,R0         ;multiply result of FIR by scaling factor of 1/4
        STF     R0,@CANCLH      ;upsampled right canceling output in CANCLH

        SUBI    1,R5            ;check for downsampling count
        BNZ     ANC

***********************************************************************
* PROCESSING LOOP AT LOWER SAMPLING RATE                                  *
***********************************************************************

        LDI     ORDERC,BK       ;circular buffer length set to the order of MIS filters
        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC11,AR0     ;AR0 points to error path (RR) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @MUSICR,R0      ;load the latest right music signal at 2K in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest right music signal at
2K
        STI     AR1,@ADDMR      ;address updated
        CALL    FIR             ;subroutine to do FIR filtering

        STF     R0,@W11         ;store output of error path (RR) filter estimate in W11

        LDI     @ADDC12,AR0     ;AR0 points to error path (RL) estimate coefficients
        LDI     ORDERC-2,RC     ;set loop counter
        CALL    FIR             ;subroutine to do FIR filtering
        STF     R0,@W12         ;store output of error path (RL) filter estimate in W12

        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @MUSICL,R0      ;load the latest left music signal at 2K in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest left music signal at
```

```
2K
        STI     AR1,@ADDML       ;address updated
        CALL    FIR              ;subroutine to do FIR filtering

        STF     R0,@W21          ;store output of error path (LR) filter estimate in W21

        LDI     @ADDC22,AR0      ;AR0 points to error path (RR) estimate coefficients
        LDI     ORDERC-2,RC      ;set loop counter
        CALL    FIR              ;subroutine to do FIR filtering
        STF     R0,@W22          ;store output of error path (LL) filter estimate in W22

**********************************************************************
*       Calculate the internally computed error signals                     *
**********************************************************************

        LDF     @ERRR,R4         ;load the downsampled right MIC error input in R4
        LDF     @W11,R0
        SUBF    R0,R4
        LDF     @W21,R0
        SUBF    R0,R4            ;R4 <---- R4 - W11 - W21
        STF     R4,@CERRR        ;store R4 in CERRR to update the filters

        LDF     @ERRL,R4         ;load the downsampled right MIC error input in R4
        LDF     @W12,R0
        SUBF    R0,R4
        LDF     @W22,R0
        SUBF    R0,R4            ;R4 <---- R4 - W12 - W22
        STF     R4,@CERRL        ;load the downsampled right MIC error input in R4



**********************************************************************
*    Estimate music signal power
```

```
*
*      P^x(n) = (1 - beta) * P^x(n-1) + beta * x(n) * x(n)
**********************************************************************
*
        LDF     @MUSICR,R5
NORMR   LDF     @POWERR,R0      ; P^x(n-1) -> R0
        LDF     @MBETA,R3       ; (1 - beta) -> R3
        MPYF    R3,R0           ; (1 - beta) * P^x(n-1) -> R0
        MPYF3   R5,R5,R1        ; x(n) * x(n) -> R1
        LDF     @BETA,R2        ; beta -> R2
        MPYF    R2,R1           ; beta * x(n) * x(n) -> R1
        ADDF    R1,R0           ; R0 + R1 -> R0
        STF     R0,@POWERR      ; R0 -> P^x(n)
        CALL    INVER
        CMPF    @HIEST,R0
        BN      ENORMR
        LDF     @HIEST,R0

*
**********************************************************************
*    Invert P^x(n) and limit to get 1/max[P^x(n), Pmin]
*    (Adapted from the TMS320C30 User's Guide, 1991, p. 11-29)
**********************************************************************
ENORMR  LDF     @MUN,R5
        MPYF    R0,R5
        STF     R5,@NPOWR

        LDF     @MUSICL,R5

**********************************************************************
*    Estimate left music signal power
*
*      P^x(n) = (1 - beta) * P^x(n-1) + beta * x(n) * x(n)
**********************************************************************
```

```
*
NORML   LDF     @POWERL,R0     ; P^x(n-1) -> R0
        LDF     @MBETA,R3     ; (1 - beta) -> R3
        MPYF    R3,R0         ; (1 - beta) * P^x(n-1) -> R0
        MPYF3   R5,R5,R1      ; x(n) * x(n) -> R1
        LDF     @BETA,R2      ; beta -> R2
        MPYF    R2,R1         ; beta * x(n) * x(n) -> R1
        ADDF    R1,R0         ; R0 + R1 -> R0
        STF     R0,@POWERL     ; R0 -> P^x(n)
        CALL    INVER

        CMPF    @HIEST,R0
        BN      ENORML
        LDF     @HIEST,R0

ENORML  LDF     @MUN,R5
        MPYF    R0,R5
        STF     R5,@NPOWL

        SUBI    1,R6            ;decrement counter in R6
        BN      UP1             ;branch if -ve to update C21(z) and C22(z)

        LDF     @CERRR,R4       ;load the internally computed right error in R4
        MPYF    @NPOWR,R4          ;multiply R4 by normalized step size

***********************************************************************
*       Update MIS filter C11(z)                                          *
***********************************************************************

        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC11,AR0     ;AR0 points to error path (RR) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update

***********************************************************************
```

```
*        Update MIS filter C12(z)                                            *
**********************************************************************

        LDF     @CERRL,R4       ;load the internally computed left error in R4
        MPYF    @NPOWR,R4          ;multiply R4 by normalized step size

        LDI     @ADDMR,AR1      ;load address of right music signal(2K) buffer in AR1
        LDI     @ADDC12,AR0     ;AR0 points to error path (RL) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update
        BR      FILT            ;skip the updation of MIS filters C21(z) and C22(z)

UP1     SUBI    1,R7            ;decrement counter in R7
        BN      RELOAD

**********************************************************************
*        Update MIS filter C21(z)                                            *
**********************************************************************

        LDF     @CERRR,R4       ;load the internally computed right error in R4
        MPYF    @NPOWL,R4          ;multiply R4 by normalized step size


        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        CALL    LMS             ;call subroutine to do least mean square update

**********************************************************************
*        Update MIS filter C21(z)                                            *
**********************************************************************

        LDF     @CERRL,R4       ;load the internally computed left error in R4
        MPYF    @NPOWL,R4          ;multiply R4 by normalized step size

        LDI     @ADDML,AR1      ;load address of left music signal(2K) buffer in AR1
```

```
        LDI     @ADDC22,AR0      ;AR0 points to error path (LL) estimate coefficients
        CALL    LMS              ;call subroutine to do least mean square update
        BR      FILT             ;skip the reloading the counters

RELOAD  LDI     COUNT2,R6        ;load counter in R6 and R7 for alternate
        LDI     COUNT2,R7        ;modeling of error paths

*********************************************************************
* FILTERED REFERENCE SIGNAL GENERATION                                    *
*********************************************************************

FILT    LDI     @ADDYF,AR1       ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC      ;loop counter set
        LDF     @REF,R0          ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%    ;AR1 modified to point to oldest REF
        STI     AR1,@ADDYF       ;address update
        LDI     @ADDC11,AR0      ;AR0 points to error path (RR) estimate coefficients
        CALL    FIR              ;subroutine to do FIR filtering
        LDF     R0,R3            ;store filtered ref. signal (RR) in R3

        LDI     @ADDYF,AR1       ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC      ;loop counter set
        LDI     @ADDC12,AR0      ;AR0 points to error path (RL) estimate coefficients
        CALL    FIR              ;subroutine to do FIR filtering

*********************************************************************
* UPDATE REFERENCE SIGNAL BUFFER FOR x'11(n) and x'12(n)                *
*********************************************************************

        LDI     @ADDM11,AR1      ;load address of filtered-X signal (RR) buffer in AR1
        LDI     ORDERW,BK        ;circular buffer length set to order of AANC filters
        STF     R3,*AR1++(1)%    ;save filtered-X signal (RR) value
        STI     AR1,@ADDM11      ;address update
```

```
        LDI     @ADDM12,AR2     ;load address of filtered-X signal (RL) buffer in AR1
        STF     R0,*AR2++(1)%   ;save filtered-X signal (RL) value
        STI     AR2,@ADDM12     ;address update

        LDF     @REF,R5

**********************************************************************
*    Estimate reference noise signal power
*
*      P^x(n) = (1 - beta) * P^x(n-1) + beta * x(n) * x(n)
**********************************************************************
*
REFNOR  LDF     @PREF,R0     ; P^x(n-1) -> R0
        LDF     @MBETA,R3     ; (1 - beta) -> R3
        MPYF    R3,R0         ; (1 - beta) * P^x(n-1) -> R0
        MPYF3   R5,R5,R1      ; x(n) * x(n) -> R1
        LDF     @BETA,R2      ; beta -> R2
        MPYF    R2,R1         ; beta * x(n) * x(n) -> R1
        ADDF    R1,R0         ; R0 + R1 -> R0
        STF     R0,@PREF      ; R0 -> P^x(n)
        CALL    INVER

        CMPF    @HIEST,R0
        BN      RENORM
        LDF     @HIEST,R0

RENORM  LDF     @MUN,R5
        MPYF    R0,R5
        STF     R5,@NORMU

**********************************************************************
* UPDATE W1(z)                                                         *
**********************************************************************
```

```
        LDF     @CERRR,R0       ;load the internally computed right error in R0
        MPYF    @NORMU,R0       ;multiply R0 by normalized step size
        LDF     @CERRL,R1       ;load the internally computed left error in R1
        MPYF    @NORMU,R1       ;multiply R1 by normalized step size
        LDI     @ADDW1,AR0      ;load address of W1 filter coefficients in AR0
        LDI     ORDERW-1,RC     ;set repeat counter

        RPTB    LMS_WA          ;repeat block up to LMS_WA
        MPYF3   *AR1++(1)%,R0,R3;R3 <--- mu*e1'(n)*x'11(n)
        MPYF3   *AR2++(1)%,R1,R4;R4 <--- mu*e2'(n)*x'12(n)
        ADDF    R3,R4           ;add R3+R4
        ADDF3   R4,*AR0,R3      ;calculate the new coefficient value
LMS_WA  STF     R3,*AR0++(1)    ;update filter W1 coefficients




**********************************************************************
* GENERATE RIGHT CHANNEL CANCELLING SIGNAL                             *
**********************************************************************

        LDI     ORDERW-2,RC     ;set repeat counter
        LDI     @ADDY,AR1       ;load address of reference signal buffer in AR1
        LDI     @ADDW1,AR0      ;load address of W1 filter coefficients in AR0
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDY       ;address update
        CALL    FIR             ;subroutine to do FIR filtering
        STF     R0,@CANCR       ;store right canceling signal at 2K in CANCR

**********************************************************************
* FILTERED REFERENCE SIGNAL GENERATION                                 *
**********************************************************************
        LDI     ORDERC,BK       ;circular buffer length set to the order of MIS filters
```

```
        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC     ;loop counter set
        LDF     @REF,R0         ;load downsampled reference signal in R0
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest REF
        STI     AR1,@ADDYF      ;address update
        LDI     @ADDC22,AR0     ;AR0 points to error path (LL) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering
        LDF     R0,R3           ;store filtered ref. signal (LL) in R3

        LDI     @ADDYF,AR1      ;load address of reference signal buffer in AR1
        LDI     ORDERC-2,RC     ;loop counter set
        LDI     @ADDC21,AR0     ;AR0 points to error path (LR) estimate coefficients
        CALL    FIR             ;subroutine to do FIR filtering

***********************************************************************
* UPDATE REFERENCE SIGNAL BUFFER FOR x'21(n) and x'22(n)                *
***********************************************************************

        LDI     @ADDM22,AR1     ;load address of filtered-X signal (LL) buffer in AR1
        LDI     ORDERW,BK       ;circular buffer length set to order of AANC filters
        STF     R3,*AR1++(1)%   ;save filtered-X signal (LL) value
        STI     AR1,@ADDM22     ;address update

        LDI     @ADDM21,AR2     ;load address of filtered-X signal (LR) buffer in AR1
        STF     R0,*AR2++(1)%   ;save filtered-X signal (LR) value
        STI     AR2,@ADDM21     ;address update

***********************************************************************
* UPDATE W2(z)                                                        *
***********************************************************************

        LDF     @CERRR,R0       ;load the internally computed right error in R0
        MPYF    @NORMU,R0       ;multiply R0 by normalized step size
```

```
        LDF     @CERRL,R1       ;load the internally computed left error in R1
        MPYF    @NORMU,R1       ;multiply R1 by normalized step size
        LDI     @ADDW2,AR0      ;load address of W1 filter coefficients in AR0
        LDI     ORDERW-1,RC     ;set repeat counter

        RPTB    LMS_WB          ;repeat block up to LMS_WB
        MPYF3   *AR1++(1)%,R0,R3;R3 <--- mu*e1'(n)*x'21(n)
        MPYF3   *AR2++(1)%,R1,R4;R4 <--- mu*e2'(n)*x'22(n)
        ADDF    R3,R4           ;add R3+R4
        ADDF3   R4,*AR0,R3      ;calculate the new coefficient value
LMS_WB  STF     R3,*AR0++(1)    ;update filter W2 coefficients

***********************************************************************
* GENERATE LEFT  CHANNEL CANCELLING SIGNAL                            *
***********************************************************************

        LDI     ORDERW-2,RC     ;set repeat counter
        LDI     @ADDY,AR1       ;load address of reference signal buffer in AR1
        LDI     @ADDW2,AR0      ;load address of W1 filter coefficients in AR0
        CALL    FIR             ;subroutine to do FIR filtering
        STF     R0,@CANCL       ;store left canceling signal at 2K in CANCL

        LDF     @MODE,R0
        CMPF    @CHECK,R0       ;check for the mode
        BN      HFCP

        LDI     DOWNSP,R5       ;reload the downsample count
        BR      ANC             ;branch back for next iteration


***********************************************************************
*       HFCP mode: To do adaptive noise cancellation and acoustic echo    *
*                  cancellation                                           *
***********************************************************************
```

```
HFCP    LDF     0.0,R0
        STF     R0,@OUTR
      ||STF     R0,@OUTL        ;setting the output’s in AANC mode to zero
        STF     R0,@CANCRH
      ||STF     R0,@CANCLH
        LDF     @COUNTER,R6     ;set counter for initial modeling
        LDI     ECHOFIL,BK      ;load the size of echo path filter in BK register

NEXT    IDLE
        SUBF    0.25,R6         ;check for the initial modeling counter
        BN      NEXT1           ;if expired stop initial modeling

        LDF     @MUSICRH,R0     ;load the latest music signal in R0
        STF     R0,@OUTR        ;send out R0 through the right speaker

        LDI     @ADDW,AR1       ;load address of right music signal buffer in AR1
        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     ECHOFIL-2,RC    ;set loop counter
        STF     R0,*AR1++(1)%   ;AR1 modified to point to oldest right music signal
        STI     AR1,@ADDW       ;address update

        CALL    FIR             ;subroutine to do FIR filtering

        LDF     @ERRLH,R4       ;get the signal picked up by left MIC in R4
        SUBF    R0,R4           ;calculate the internal error

        MPYF    @MU,R4          ;multiply R4 with the normalized step size
        LDI     @ADDW,AR1       ;load address of right music signal buffer in AR1
        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     ECHOFIL-2,RC    ;set loop counter
        CALL    LMS             ;call subroutine to do least mean square update
        BR      NEXT            ;loop back
```

```
*********************************************************************
*       After initial modeling of echo path..full HFCP mode is ON now       *
*********************************************************************


NEXT1   LDF     0.0,R0
        STF     R0,@OUTR        ;initialize the variables to zero
        STF     R0,@RESECHO
        LDI     1,R6            ;set initial hangover counter for far-end speech to 1
        LDI     1,R7            ;set initial hangover counter for double talk to 1

ECHO    IDLE

        LDF     @MODE,R0
        CMPF    @CHECK,R0       ;check for the mode
        BNN     AANC



*********************************************************************
*       Adaptive noise cancellation (ANC)                                   *
*********************************************************************

        LDF     @REF,R0         ;load the reference noise signal in R0
        LDI     @ADDENC,AR0     ;load the address of ANC filter in AR0
        LDI     @ADDR,AR1       ;load the address of reference noise signal buffer in
AR1
        LDI     ORDERWA,BK      ;circular buffer length set to the order of ANC filter
        STF     R0,*AR1++(1)%   ;AR1 modified to point to the oldest reference noise
signal
        STI     AR1,@ADDR       ;address update
        LDI     ORDERW-2,RC     ;set loop counter
        CALL    FIR             ;subroutine to do FIR filtering

        LDF     @ERRLH,R4       ;load the left MIC input in R4
```

```
        SUBF    R0,R4           ;calculate the difference to remove noise
        STF     R4,@ECNOUT      ;store the noise free input in ENCOUT

        MPYF    @MU,R4          ;multiply ENCOUT with normalized step size
        LDI     @ADDR,AR1       ;load the address of reference noise signal buffer in
AR1
        LDI     @ADDENC,AR0     ;load the address of ANC filter in AR0
        LDI     ORDERW-2,RC     ;set loop counter
        CALL    LMS             ;call subroutine to do least mean square update

        LDF     @FARENSP,R0     ;load far end reference speech in R0
        STF     R0,@OUTR        ;send out far-end speech through the loudspeaker

;Call subroutines to calculate the power estimates for the three windows
        CALL    VSPOWER         ;call subroutine to calculate very short window power
estimate
        CALL    SPOWER          ;call subroutine to calculate short window power
estimate
        CALL    LPOWER          ;call subroutine to calculate long window power
estimate

        SUBF3   R2,R3,R4        ;check if SPOWER > LPOWER
        BN      WIN1            ;If not , do not swap
        LDF     R2,R3           ;change long window to short window
WIN1    SUBF    R3,R1           ;sub very short window from short window
        SUBF    @SPTHR,R1       ;compare result with speech threshold
        BN      DN              ;skip speech flag over if speech not  PRESENT

        LDI     HOCNT,R6        ;set the far-end speech hangover counter

        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     @ADDFENS,AR1    ;load the address of far-end speech signal buffer in
AR1
        LDI     ECHOFIL-2,RC    ;set loop counter
        LDF     @FARENSP,R0     ;load far end reference speech in R0
        STF     R0,*AR1++(1)%   ;AR1 points to the oldest far-end speech
```

```
        STI     AR1,@ADDFENS    ;address update
        CALL    FIR             ;call subroutine to do FIR filtering

        LDF     @ECNOUT,R1      ;load the output of ANC in R1
        SUBF    R0,R1           ;subtract the FIR output from R1 to cancel echo
        STF     R1,@RESECHO     ;store R1 in RESECHO

***********************************************************************
*       Double Talk detection                                         *
***********************************************************************
        LDF     @ECNOUT,R0      ;load the output of ANC in R0
        CALL    SPOWDT1         ;call subroutine to calculate power of R0
        LDF     R2,R3           ;store new power in R3
        LDF     @RESECHO,R0     ;load the residual echo signal in R0
        CALL    SPOWDT2         ;call subroutine to calculate power in R0
        LDF     R2,R0           ;load new power in R0
        CALL    INVERF          ;call routine to find inverse of R0
        MPYF    R3,R0           ;R0=R3/R0
        SUBF    @THRESDT,R0     ;check if R0 is greater than double talk threshold
        BNN     NT1             ;if yes branch to check for hangover counter for far-
end speech
        LDI     HC,R7           ;else load hangover counter for Double talk
        BR      ECHO            ;branch back for next iteration

NT1     SUBI    1,R7            ;decrement the hangover counter for Double talk
        BNZ     ECHO            ;if not zero branch back for next iteration
                                ;else do update of echo path filter

        LDF     @FARENSP,R5     ;load the current speech sample in R5
        LDF     @ECHOPO,R0      ;P^x(n-1) -> R0
        LDF     @MBETA,R3       ;(1 - beta) -> R3
        MPYF    R3,R0           ;(1 - beta) * P^x(n-1) -> R0
        MPYF3   R5,R5,R1        ;x(n) * x(n) -> R1
        LDF     @BETA,R2        ;beta -> R2
        MPYF    R2,R1           ;beta * x(n) * x(n) -> R1
```

```
        ADDF    R1,R0           ;R0 + R1 -> R0
        STF     R0,@ECHOPO      ;R0 -> P^x(n)
*
        CALL    INVERF          ;call subroutine to find the inverse of power
        CMPF    @HIEST,R0       ;compare 1/P^x with 1/Pmin
        BN      ENORM           ;if 1/P^x < 1/Pmin, branch to ENORM
        LDF     @HIEST,R0       ; if 1/P^x > 1/Pmin, R0 = 1/Pmin
*
***********************************************************************
*    Compute adaptive step size
*
*      mu(n) = (alpha/L) / max[P^x(n), Pmin]
***********************************************************************
*
ENORM   LDF     @MUN,R4         ;calculate the normalized step size
        MPYF    R0,R4
        LDF     @RESECHO,R0
        MPYF    R0,R4           ;multiply normalized step size with residual echo
        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     ECHOFIL-2,RC    ;set loop counter
        LDI     @ADDFENS,AR1    ;load the address of far-end speech signal buffer in
AR1
        CALL    LMS             ;call subroutine to do least mean square update

        LDI     1,R7            ;reset the double talk hangover counter

        BR      ECHO            ;branch back for next iteration

DN      SUBI    1,R6            ;decrement the hangover counter for far-end speech
        BZ      ENDFLAG         ;if zero=no far-end speech

        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     @ADDFENS,AR1    ;load the address of far-end speech signal buffer in
AR1
        LDI     ECHOFIL-2,RC    ;set loop counter
```

```
        LDF     @FARENSP,R0     ;load far end reference speech in R0
        STF     R0,*AR1++(1)%   ;AR1 points to the oldest far-end speech
        STI     AR1,@ADDFENS    ;address update
        CALL    FIR             ;call subroutine to do FIR filtering

        LDF     @ECNOUT,R1      ;load the output of ANC in R1
        SUBF    R0,R1           ;subtract the FIR output from R1 to cancel echo
        STF     R1,@RESECHO     ;store R1 in RESECHO

***********************************************************************
*       Double Talk detection                                         *
***********************************************************************
        LDF     @ECNOUT,R0      ;load the output of ANC in R0
        CALL    SPOWDT1         ;call subroutine to calculate power of R0
        LDF     R2,R3           ;store new power in R3
        LDF     @RESECHO,R0     ;load the residual echo signal in R0
        CALL    SPOWDT2         ;call subroutine to calculate power in R0
        LDF     R2,R0           ;load new power in R0
        CALL    INVERF          ;call routine to find inverse of R0
        MPYF    R3,R0           ;R0=R3/R0
        SUBF    @THRESDT,R0     ;check if R0 is greater than double talk threshold
        BNN     NT2             ;if yes branch to check for hangover counter for far-
end speech
        LDI     HC,R7           ;else load hangover counter for Double talk
        BR      ECHO            ;branch back for next iteration

NT2     SUBI    1,R7            ;decrement the hangover counter for Double talk
        BNZ     ECHO            ;if not zero branch back for next iteration
                                ;else do update of echo path filter

        LDF     @FARENSP,R5     ;load the current speech sample in R5
        LDF     @ECHOPO,R0      ;P^x(n-1) -> R0
        LDF     @MBETA,R3       ;(1 - beta) -> R3
        MPYF    R3,R0           ;(1 - beta) * P^x(n-1) -> R0
        MPYF3   R5,R5,R1        ;x(n) * x(n) -> R1
```

```
        LDF     @BETA,R2        ;beta -> R2
        MPYF    R2,R1           ;beta * x(n) * x(n) -> R1
        ADDF    R1,R0           ;R0 + R1 -> R0
        STF     R0,@ECHOPO      ;R0 -> P^x(n)
*
        CALL    INVERF          ;call subroutine to find the inverse of power
        CMPF    @HIEST,R0       ;compare 1/P^x with 1/Pmin
        BN      ENORM1          ;if 1/P^x < 1/Pmin, branch to ENORM1
        LDF     @HIEST,R0       ; if 1/P^x > 1/Pmin, R0 = 1/Pmin
*
***********************************************************************
*    Compute adaptive step size
*
*      mu(n) = (alpha/L) / max[P^x(n), Pmin]
***********************************************************************
*
ENORM1  LDF     @MUN,R4         ;calculate the normalized step size
        MPYF    R0,R4
        LDF     @RESECHO,R0
        MPYF    R0,R4           ;multiply normalized step size with residual echo
        LDI     @ADDEFIL,AR0    ;load the address of echo path filter in AR0
        LDI     ECHOFIL-2,RC    ;set loop counter
        LDI     @ADDFENS,AR1    ;load the address of far-end speech signal buffer in
AR1
        CALL    LMS             ;call subroutine to do least mean square update

        LDI     1,R7            ;reset the double talk hangover counter

        BR      ECHO            ;branch back for next iteration


ENDFLAG LDF     @ECNOUT,R0      ;if no far-end speech present
        STF     R0,@RESECHO     ;send output of ANC as transmit signal
        LDI     1,R6            ;reset hanover counter for far-end speech
        BR      ECHO            ;branch back for next iteration
```

```
*********************************************************************
*
*   Speech Power Signal Interrupt Service Routine (Very Short Window)
*
*   For speech detections - 4ms window
*
*   Parameter passed in as RO passed out as R1
*
*   Overwritten registers - R1,R4
*********************************************************************
*
VSPOWER MPYF3   R0,R0,R1        ; Square input
        MPYF    @VSBETA,R1      ; Weight input
        LDF     @VSOPOWE,R4     ; Weight old input
        MPYF    @VSOMBET,R4
        ADDF3   R1,R4,R1        ; Add together
        STF     R1,@VSOPOWE     ; Store in old power for next iteration

        RETS
*
*********************************************************************
*
*   Speech Power Signal Interrupt Service Routine (Short Window)
*
*   For speech detections - 16ms window
*
*   Parameter passed in as RO passed out as R2
*
*   Overwritten registers - R2,R4
*********************************************************************
*
SPOWER  MPYF3   R0,R0,R2        ; Square input
        MPYF    @SBETA,R2       ; Weight input
```

```
        LDF     @SOPOWER,R4     ; Weight old input
        MPYF    @SOMBETA,R4
        ADDF3   R2,R4,R2        ; Add together
        STF     R2,@SOPOWER     ; Store in old power for next iteration

        RETS
*
***********************************************************************
*
*   Power Signal Interrupt Service Routine  (Long window)
*
*   For convergence of error signal - 1 s window
*
*   Parameter passed in as RO passed out as R3
*
*   Overwritten registers - R3,R4
***********************************************************************
*
LPOWER  MPYF3   R0,R0,R3        ; Square input
        MPYF    @LBETA,R3       ; Weight input
        LDF     @LOPOWER,R4     ; Weight old input
        MPYF    @LOMBETA,R4
        ADDF3   R3,R4,R3        ; Add together
        STF     R3,@LOPOWER     ; Store in old power for next iteration

        RETS
***********************************************************************
*
SPOWDT1 MPYF3   R0,R0,R2        ; Square input
        MPYF    @VSBETA,R2      ; Weight input
        LDF     @SOPODT1,R4     ; Weight old input
        MPYF    @VSOMBET,R4
        ADDF3   R2,R4,R2        ; Add together
        STF     R2,@SOPODT1     ; Store in old power for next iteration
```

```
        RETS
*
***********************************************************************
*
SPOWDT2 MPYF3   R0,R0,R2        ; Square input
        MPYF    @VSBETA,R2      ; Weight input
        LDF     @SOPODT2,R4     ; Weight old input
        MPYF    @VSOMBET,R4
        ADDF3   R2,R4,R2        ; Add together
        STF     R2,@SOPODT2     ; Store in old power for next iteration

        RETS

***********************************************************************
INVER     PUSHF   R0            ; push floating point value of P^x
          POP     R1            ; pop value as signed integer -> R1
          ASH     -24,R1        ; right shift 24 bits to get exponent e
          NEGI    R1            ; negate exponent to get -e
          SUBI    1,R1          ; R1 = -e - 1, the exponent of x[0]
          ASH     24,R1         ; left shift 24 bits
          PUSH    R1            ; push floating point value
          POPF    R1            ; R1 = x[0] = 1.0 * 2**(-e - 1)
          MPYF3   R1,R0,R2      ; R2 = P^x * x[0]
          SUBRF   2.0,R2        ; R2 = 2.0 - P^x * x[0]
          MPYF    R2,R1         ; R1 = x[1] = x[0] * (2.0 - P^x * x[0])
          MPYF3   R1,R0,R2      ; R2 = P^x * x[1]
          SUBRF   2.0,R2        ; R2 = 2.0 - P^x * x[1]
          MPYF    R2,R1         ; R1 = x[2] = x[1] * (2.0 - P^x * x[1])
          MPYF3   R1,R0,R2      ; R2 = P^x * x[2]
          SUBRF   2.0,R2        ; R2 = 2.0 - P^x * x[2]
          MPYF    R2,R1         ; R1 = x[3] = x[2] * (2.0 - P^x * x[2])
          MPYF3   R1,R0,R2      ; R2 = P^x * x[3]
          SUBRF   2.0,R2        ; R2 = 2.0 - P^x * x[3]
```

```
          MPYF    R2,R1          ; R1 = x[4] = x[3] * (2.0 - P^x * x[3])
          RND     R1             ; This minimizes error in the LSBs
          MPYF3   R1,R0,R2       ; R2 = P^x * x[4]
          SUBRF   1.0,R2         ; R2 = 1.0 - P^x * x[4]
          MPYF    R1,R2          ; R2 = x[4] * (1.0 - P^x * x[4])
          ADDF    R2,R1          ; R1 = x[5] = x[4] * (1.0 - P^x * x[4]) + x[4]
          RND     R1,R0          ; round x[5] -> R0 = 1/P^x

***********************************************************************
*       Subroutine to do FIR filtering                                *
***********************************************************************

FIR     MPYF3   *AR0++(1)%,*AR1++(1)%,R0        ;multiplies W_A(N-1)*x(n-(N-1))
                                                ;W_A(N-1) is the N-1 th filter
                                                ;coefficient pointed to by AR0
                                                ;and x(n-(N-1)) is the oldest input
                                                ;signal data pointed by AR1
        NOP
        LDF     0.0,R2                          ;initialize R2 to 0
        RPTS    RC                              ;set up counter to perform summation
                                                ;from 0 to N-2, where N is the order of
                                                ;the filter concerned
        MPYF3   *AR0++(1)%,*AR1++(1)%,R0        ;multiple remaining [0,..,N-2] filter
                                                ;coefficients with input signal
                                                ;data x(n),.....x(n-(N-2)) and save
                                                ;result in R0
     || ADDF3   R0,R2,R2                        ;in addition to multiplication add
                                                ;previous summation into R2 thus
                                                ;performing convolution between filter
                                                ;coefficients and signal data
        ADDF3   R0,R2,R0                        ;add last product
        RETS                                    ;return to main program

***********************************************************************
```

```
*       Subroutine to perform LMS adaption                                *
***********************************************************************

LMS     MPYF3   *AR1++(1)%,R4,R1        ;x(n-(N-1))*mu*err --> R1
                                        ;multiply oldest signal data with the
                                        ;product of mu and error signal e(n)
        RPTB    ADAP                    ;repeat N-2 times where N is the order
                                        ;of filter being updated
        MPYF3   *AR1++(1)%,R4,R1        ;multiply remaining signal vector data
                                        ;with product mu*e(n) and save result in R1
      ||ADDF3   R1,*AR0,R2              ;in parallel, add the previous product to
                                        ;corresponding adaptive filter coefficient
                                        ;to get new coefficient
ADAP    STF     R2,*AR0++(1)            ;update the filter coefficients
        NOP
        ADDF3   R1,*AR0,R2              ;add last product
        STF     R2,*AR0                 ;update 0th filter coefficient
        RETS                            ;return to the main program

***********************************************************************
*       Interrupt service routine for both system board and I/O board      *
*       4 Channel I/O board                                                *
*               Channel A0:     Input :From right microphone               *
*                               Output:To right speaker                    *
*                                                                          *
*               Channel A1:     Input :From left microphone                 *
*                               Output:To left speaker                     *
*                                                                          *
*               Channel B0      Input :Stereo system input                 *
*                                                                          *
*               Channel B1      Input :Reference noise signal input        *
*                                                                          *
*       System Board                                                       *
*               Channel A       Input :Mode input                          *
```

```
*                                                                           *
*               Channel B       Input :Receive signal from cellular phone   *
*                               Output:Transmit signal to cellular phone    *
***********************************************************************

ISR:    PUSHF   R4
        PUSH    R4
        PUSHF   R5
        PUSH    R5
        PUSH    AR0
        PUSH    AR1
        LDI     @BRDIS_A,AR1    ;Load the board status
        LDI     *AR1,R1
        TSTB    @TSTMASK,R1     ;Test to see if interrupt is from site A
        BNZ     ISR_END         ;Branch if not
        LDI     @INTS_A,AR1
        LDI     *AR1,R1

        LDI     @CH0_A,AR1      ;load the address of channel A0 in AR1
        LDI     *AR1,R5         ;store value from A/D in R5
        ASH     -16,R5          ;shift by 16 bits
        FLOAT   R5,R5           ;convert to floating point format
        MPYF    @S32768,R5      ;scale the input
        STF     R5,@ERRRH       ;store the right MIC input

        LDF     @OUTR,R5        ;get right speaker output in R5
        LDF     @CANCRH,R4      ;get right channel cancelling signal in R4
        ADDF    R4,R5           ;add R4 and R5
        MPYF    @S32767,R5      ;scale R5 by 32767
        FIX     R5,R5           ;convert R5 to integer value
        LSH     16,R5           ;shift left by 16 bits
        STI     R5,*AR1         ;output signal to secondary source

        LDI     @CH1_A,AR1      ;load address of channel A1 in AR1
```

```
        LDI     *AR1,R5         ;store value from A/D in R5
        ASH     -16,R5          ;shift by 16 bits
        FLOAT   R5              ;convert to floating point format
        MPYF    @S32768,R5      ;scale the input
        STF     R5,@ERRLH       ;store the left MIC input

        LDF     @OUTL,R5        ;get left speaker output in R5
        LDF     @CANCLH,R4      ;get left channel cancelling signal in R4
        ADDF    R4,R5           ;add R4 and R5
        MPYF    @S32767,R5      ;scale R5 by 32767
        FIX     R5,R5           ;convert R5 to integer value
        LSH     16,R5           ;shift left by 16 bits
        STI     R5,*AR1         ;output signal to secondary source

        LDI     @CH0_B,AR1      ;load address of channel B0 in AR1
        LDI     *AR1,R5         ;store value from A/D in R5
        ASH     -16,R5          ;shift by 16 bits
        FLOAT   R5              ;convert to floating point format
        MPYF    @S32768,R5      ;scale the input
        STF     R5,@MUSICRH
        STF     R5,@MUSICLH     ;store the stereo system input

        LDI     @CH1_B,AR1      ;load address of channel B1 in AR1
        LDI     *AR1,R5         ;store value from A/D in R5
        ASH     -16,R5          ;shift by 16 bits
        FLOAT   R5              ;convert to floating point format
        MPYF    @S32768,R5      ;scale the input
        STF     R5,@REFH        ;store the reference noise signal

***********************************************************************
*       I/O on the system board                                       *
***********************************************************************
        LDI     @ADDCHA,AR1     ;load address of channel A in AR1
        LDI     *AR1,R5         ;store value from A/D in R5
```

```
ASH     -16,R5          ;shift by 16 bits
FLOAT   R5              ;convert to floating point format
MPYF    @S32768,R5      ;scale the input

LDI     @DELAY1,AR0     ;load address of delay buffer 1 in AR0
LDI     DELAY,BK        ;circular buffer size set to delay
LDF     *AR0,R4         ;load the delayed input in R4
STF     R4,@MODE        ;store the delayed input
STF     R5,*AR0++(1)%   ;store the current input at the address pointed by AR0
STI     AR0,@DELAY1     ;update address

LDI     @ADDCHB,AR1     ;load address of channel B in AR1
LDI     *AR1,R5         ;store value from A/D in R5
ASH     -16,R5          ;shift by 16 bits
FLOAT   R5              ;convert to floating point format
MPYF    @S32768,R5      ;scale the input

LDI     @DELAY2,AR0     ;load address of delay buffer 2 in AR0
LDF     *AR0,R4         ;load the delayed input in R4
STF     R4,@FARENSP     ;store the delayed input
STF     R5,*AR0++(1)%   ;store the current input at the address pointed by AR0
STI     AR0,@DELAY2     ;update address

LDI     @DELAY3,AR0     ;load address of delay buffer 3 in AR0
LDF     *AR0,R5         ;load the delayed output in R5
LDF     @RESECHO,R4     ;load the current output in R4
STF     R4,*AR0++(1)%   ;store the current output at the address point by AR0
STI     AR0,@DELAY3     ;address update

MPYF    @S32767,R5      ;scale R5 by 32767
FIX     R5,R5           ;convert R5 to integer value
LSH     16,R5           ;shift left by 16 bits
STI     R5,*AR1         ;output signal to transmit output
```

```
ISR_END:
        AND     0FFFEh,IF       ;Clear the C30's interrupt register
        POP     AR1
        POP     AR0
        POP     R5
        POPF    R5
        POP     R4
        POPF    R4
        RETI                    ;return from the interrupt service routine

        .end
```