Implementing a Single Channel Active Adaptive Noise Canceller with the TMS320C50 DSP Starter Kit

APPLICATION REPORT: SPRA285

Authors: Stéphane Boucher
Martin Bouchard
Andre L'esperance
Bruno Paillard
Department of Electrical and Computer Engineering
Faculty of Applied Sciences
University of Sherbrooke, Sherbrooke, Quebec

Digital Signal Processing Solutions
November 1997

TEXAS INSTRUMENTS

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 application 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Implementing a Single Channel Active Adaptive Noise Canceller with the TMS320C50 DSP Starter Kit

Abstract

This application report shows how to implement a single channel active adaptive noise canceller with the low cost Texas Instruments (TI™) TMS320C50 (‘C50) digital signal processor (DSP) starter kit (DSK). Once the needed equipment is obtained, building a functional active noise canceller is easy with the help of this application report. The canceller uses a feedback control topology and is designed to cancel periodic noise.

This application report supports the text with equations, schematics, and block diagrams.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

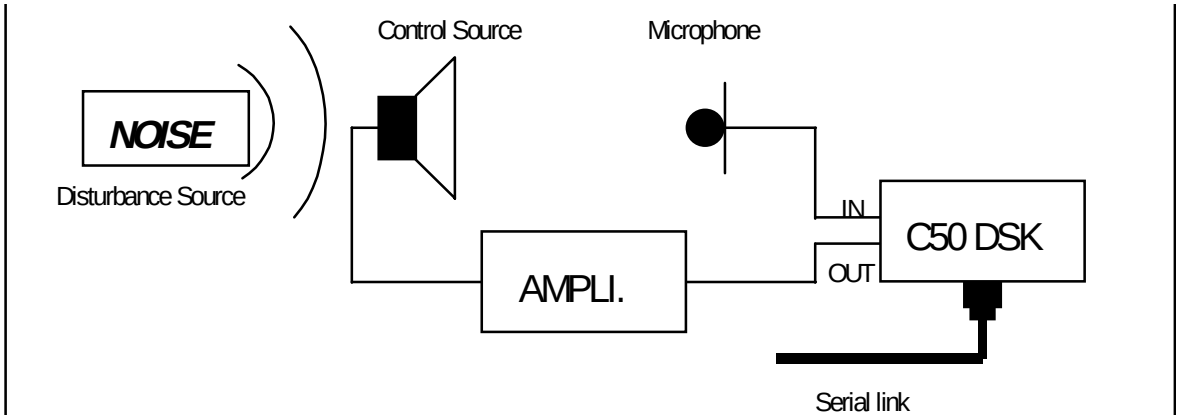
Basic Principles of Active Control

Active noise control works on the principle of destructive interference between the noise wave generated by a primary source (the one to be canceled) and the interference wave, generated by the control source (usually a loudspeaker). It is the control and the generation of this acoustic interference wave that is processed by the DSP. The DSP performs the real-time calculations necessary to generate the control signal (generally a FIR filtering process) and optimize the control filter (usually a filtered-X LMS algorithm).

Description of the Controller

The controller is a feedback type controller, physically similar to the one presented by Olson and May in 1953[1]. The choice of a feedback configuration leads to a very simple system composed of a few low-cost components (see Figure 1).

Figure 1. Single Channel Active Adaptive Noise Canceller



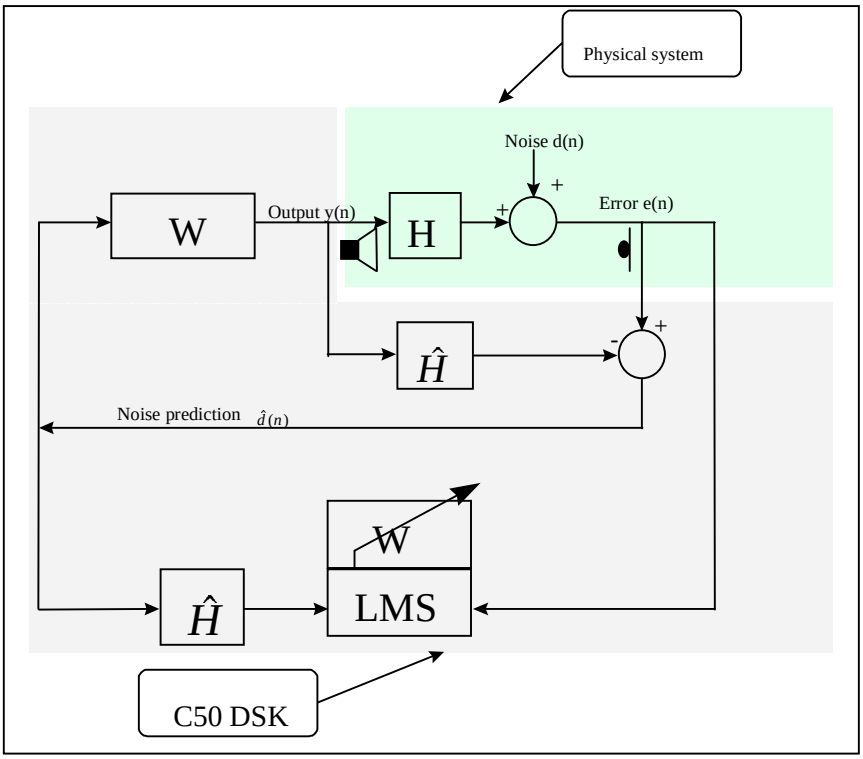
The goal of the system is to create a quiet zone around the microphone. The dimension of the quiet zone depends on the frequency of the noise to be attenuated and is larger for a lower frequency. The approximate dimension of the quiet zone is one tenth of a wavelength[2].

This system can also achieve *global* control if the control loudspeaker can generate a sound field that matches the spatial distribution of the sound field created by the noise source.

For example, if the noise source is a loudspeaker generating a 100 Hz pure tone, global control can be achieved by positioning the loudspeaker of the controller near (compared to the wavelength) the source of noise. This is possible because the secondary source can then match the spatial distribution of the primary source, both being point (or almost) sources.

As described in the previous section, *Basic Principles of Active Control*, the problem to be solved is the generation of acoustic interference to reduce noise at the microphone. Figure 2 shows the proposed solution, in which a feedback controller uses a feedforward approach.[3]

Figure 2. Feedback Controller Using a Feedforward Approach



In Figure 2, *H* represents the plant transfer function; or the impulse response between the output signal sent to the loudspeaker, *y(n),* and the error signal measured at the microphone, *e(n)*. A 256-coefficient FIR model of H ($\hat{H}$ as shown in Figure 2) estimates the contribution of the control signal at the microphone. This estimate is digitally subtracted from the microphone signal, yielding the signal $\hat{d}$ *(n),* which represents the unwanted signal alone (the contribution from the control speaker has been subtracted).

The output of the secondary source (the interference wave) is a linear combination of $\hat{d}$ (n), the estimate of the noise. The adaptive filter *W* is the control filter and has a transversal structure. Its 16 weights are adapted with the filtered-X LMS algorithm[4]. This filter has two purposes:

- Predict the value of the perturbation signal some time in the future, from the present and past sample values. The prediction time corresponds roughly to the propagation delay between the loudspeaker and the microphone.

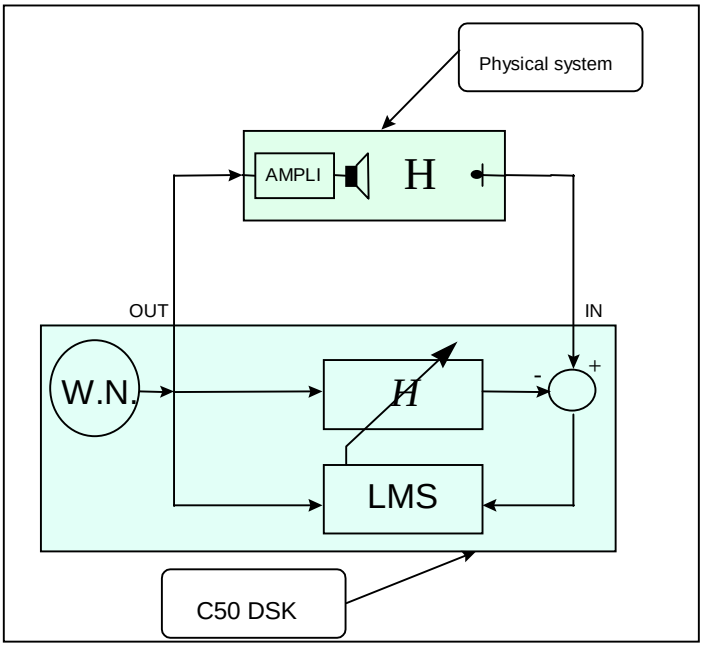
- Generate from the predicted value the control sample to send to the control speaker. This generation takes into account the transfer function between the speaker and the microphone.

Actually, both filters are combined into a single control filter *W*.

This description shows one limit of feedback controllers: to achieve full control, the disturbance signal has to be predictable. Therefore, feedback controllers work well on narrow band and periodic signals, but give a poor performance on a white random signal.

Figure 3 shows the identification process that must be performed prior to the control itself to estimate the transfer function of the plant $\hat{H}$.

Figure 3. Identification Process of the Plant



During the identification, a white noise generated by the ‘C50 is driven to the speaker for a few seconds. The microphone signal is monitored during this excitation, and a LMS algorithm identifies *H* from the input and output signals.

After this phase, the control algorithm of Figure 2 is engaged, and its filtered-X LMS algorithm adapts *W* to minimize the residual energy of the error microphone. Within a few seconds, the energy drops at the location of the microphone.

The filtered-X LMS continues adaptation so that the controller can track variations in the frequencies of the signals being canceled, or in the spectral content (relative amplitudes of the tones).

Number of Taps for $\hat{H}$ and *W*

The number of taps needed to identify the plant depends on the sampling frequency chosen and the length of the impulse response of the plant. The plant impulse response is the speaker-to-microphone impulse response. It is affected by the acoustic reverberation of the room; therefore, the more reverberant the room, the longer the impulse response. If the canceller is used in a room with a long impulse response, and at a higher sampling frequency than the default one (600 Hz), more taps may be needed for $\hat{H}$. But, for a sampling frequency of 600 Hz, the 256 taps should be enough for most rooms.

For the control filter *W*, 16 is the number of taps used in the code. This should be enough to control up to 8 tones. A minimum of two coefficients per tone is needed to control the tones properly.

Hardware Considerations

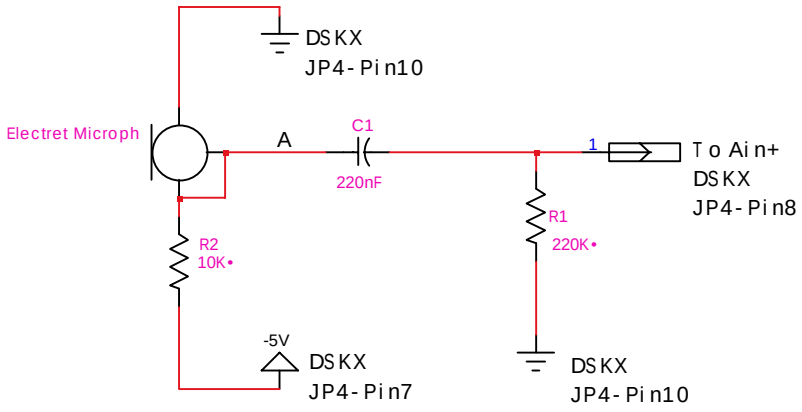
Speaker-Microphone Distance

For the control to work well, the distance between the microphone and the speaker should not be changed once the identification has been done. The distance used by the authors is approximately 30 cm, or 12 inches.

Input Stage

The canceller hardware is easy to set up and needs little explanation, except for the input stage if an electret microphone is used. The input stage is composed of an electret microphone, two resistors, and a capacitor. The schematic is shown in Figure 4.

Figure 4. Input Stage



A 2-wire connection (amplifier configuration) is used on the microphone to have a high sensitivity. The polarizing resistor value (R2) is a compromise between an uneven polarization (point A on the schematic is not polarized at -2.5 V) and a high sensitivity. It can be adjusted depending on the required sensitivity and the particular model of microphone used.

The polarization is negative (derived from the –5 V power supply of the DSK board) instead of the more common positive because the –5 V is the only regulated power supply available from a connector on the DSK board. The disadvantage is that the casing of the microphone, connected to the negative lead on most microphones, will not connect to ground in this configuration. This can make the microphone more sensitive to capacitive coupled noise, although we did not observe this problem with the system.

The common values of the capacitor C1 and resistor R1 provide a high-pass cutoff of 3 Hz. This is quite low for most applications but can be adjusted, if needed.

Signal Levels

Some signal levels must be specified for the canceller to work well with default values of variables in the ‘C50 code (for an example of the ’C50 code, see the Appendix).

- The noise signal, without control, should lead to a 50 to 200 mV peak signal returned by the microphone. If necessary, R2 can be adjusted or a preamplifier can be added in the circuit to follow this requirement.
- For an amplifier input voltage equal to 3 V peak, the maximum AIC (analog interface circuit) output, the amplifier gain should be set to obtain at least a 200 mV peak signal returned by the microphone.

  In other words, the amplifier should be able to drive the speaker to generate the same sound pressure at the microphone as the noise itself without control.

DSK Implementation

All the controller code is running on the TMS320C50 DSP. Once the code is assembled, it can be loaded with the DSK debugger included with the package[5]. The DSP program can be divided into two principal sections:

- Main
- Interrupt

The main section executes all the non-real-time processing. The different tasks described in Figure 5 execute sequentially.

The interrupt section handles all the real-time processing, including identification, control, and optimization processes. It executes on a sample-by-sample basis, under interruption from the AIC. All these processes execute in three phases, as shown Figure 6.

Figure 5. Main Section

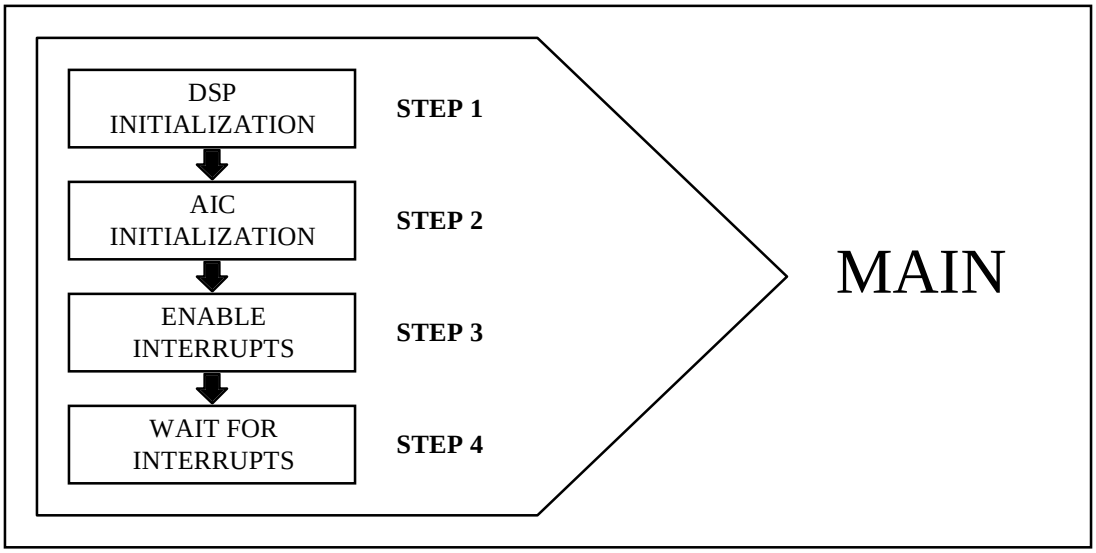
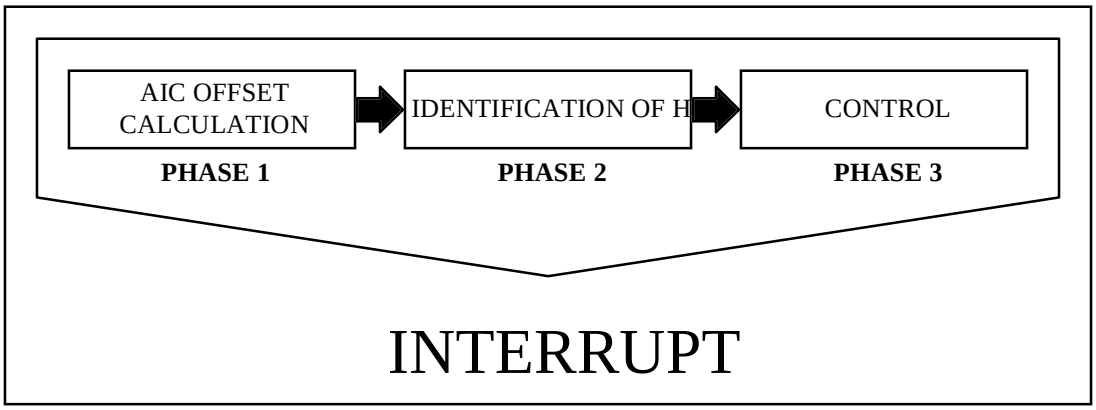


Figure 6. Interrupt Section



Main Routines

’C50 Initialization

The first step is to initialize the DSP (for information on ’C50 initialization, see the TI *TMS320C50 User’s Guide*).[6]

AIC Initialization

The TI TLC32040 analog interface circuit (AIC) is installed on the DSK board.[5] Because the description of its initialization routine is beyond the scope of this document, we will only explain how to obtain the desired sampling frequency by modifying some global variables in the code.

First, since the AIC is initialized to run in synchronous mode, only the global variables TA, TB, and VPRD are used to configure the sampling rate and the antialiasing filters. TA and TB are AIC registers, which adjust the sampling frequency of the AIC and cut-off frequency of its switched capacitor filters. VPRD adjusts the frequency of the timer of the ’C50, whose output drives the master clock (Mc) of the AIC.

The value of TB adjusts the ratio between the sampling frequency and the cutoff frequency of the low-pass filter. By adjusting TB to 32, we decide that the cut-off frequency (-70 dB) is approximately half the sampling frequency.

This value should not be changed unless you want to modify the ratio between the sampling frequency and the cut-off frequency, although increasing the cut-off frequency will bring some aliasing. Then, only TA and VPRD should be adjusted to modify the sampling frequency. The sampling frequency (F_s) is given by:

$$TA = \frac{M_c}{2 \times SCF} = \frac{10^6}{2 \times TB \times F_S} = \frac{10^6}{2 \times 32 \times F_S}$$

If the result for TA is smaller than 31, the maximum value allowed for TA, the PRD register of the ’C50, can stay at 1 (the default VPRD value that gives a Master clock frequency of 10 MHz). Otherwise, TA can be fixed to the maximum value 31 and a new VPRD can be calculated:

$$VPRD = INT\left(\frac{10^6}{TA \times TB \times F_S} - 1\right) = INT\left(\frac{10^6}{31 \times 32 \times F_S} - 1\right)$$

The default values in the code for VPRD (16), TA (31) and TB (32) provide a sampling frequency of 600 Hz, a low-pass cut-off frequency (-70 dB) of 300 Hz and a high-pass cut-off frequency of approximately 20 Hz. With these settings, the control should work properly for periodic signals of frequency between about 100 Hz and 250 Hz.

Enable Interrupts

This step enables the receive interrupt from the serial port of the ‘C50. When this interrupt is active, since the AIC operates in synchronous mode, a 16 bit sample value has just been transmitted to the AIC and another one received from the AIC.

Wait for Interrupts

An infinite *nop* loop starts. The ’C50 waits for serial port interrupts.

Interrupt Routine

The interrupt routine is divided into three phases executed sequentially just after the main program has enabled interrupts.

The first phase (phase 1) calculates the input offset of the AIC.

The second phase (phase 2) identifies the transfer function between the control source and the microphone.

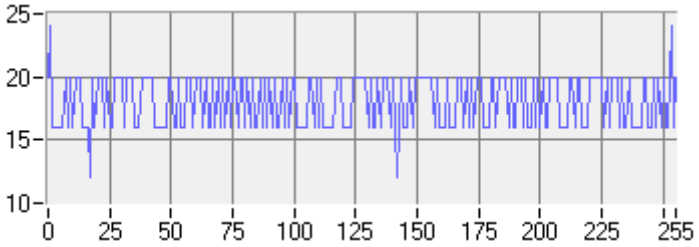
The third phase (phase 3) is the control itself.

AIC Input Offset Calculation

If you observe the signal returned by the AIC to the DSP when its input is connected to GND, you will notice that the average of the signal is not zero but rather equal to a certain offset (Figure 6). This offset should be subtracted from any sample coming in from the AIC. This is why the first thing that is done after enabling the AIC interrupts is to average this offset on 64 input samples.

Since the AIC returns a couple of irrelevant values immediately after configuration, it is necessary to let about 200 samples go before we add the 64 samples to average. This offset calculation process takes place in phase 1 of the interrupt routine.

Figure 7. Signal Returned by the AIC with Input Connected to GND



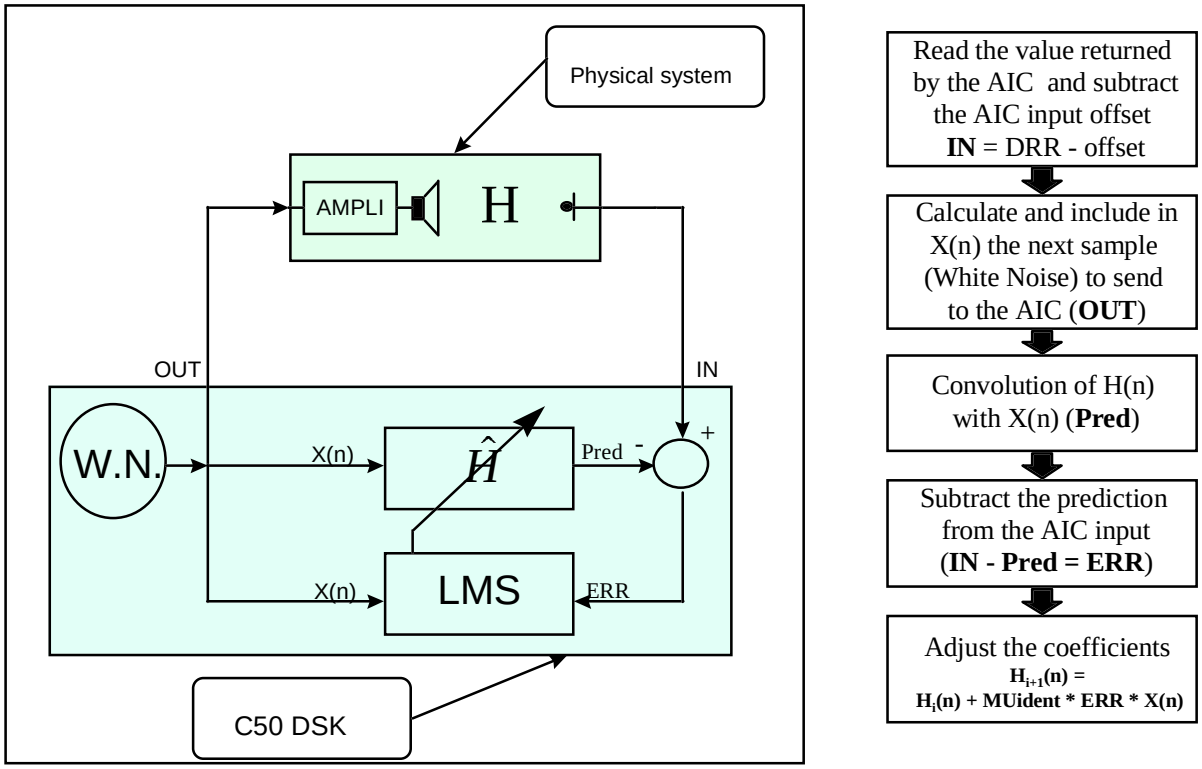
Identification of *H*

Once the DSP and the AIC are initialized and the AIC offset is calculated, the identification process of *H* begins. The goal of this process is to adjust the 256 coefficients of the FIR filter $\hat{H}$ to minimize the error signal *ERR* in Figure 8. The adaptive algorithm responsible for adjusting the coefficients is a simple but effective LMS.

For a good identification performance, the unwanted noise should not be present during this process. The convergence of the identification process can be measured by the ratio, in dB, of the expectation of the energy of the input signal (IN in Figure 8) over the expectation of the energy of the error (ERR in Figure 8):

$$20\log\frac{E(IN^2)}{E(ERR^2)}$$

Figure 8. Steps of the Identification Phase, Executed for ITER Iterations



To fully characterize the plant, a white noise is used as input to the system. The adaptation of the coefficients takes place over *ITER* iterations; that is, until the counter *counter* reaches the *ITER* value. This global variable should be set depending on the precision of the identification needed.

A number of iterations equal to *N* times the filter length improves the performance of the identification by a factor of *N* (lowers the amplitude of the error signal *ERR* by a factor of *N*). For example, 2560 iterations (10 filter lengths) should lead to a convergence of 20 dB while 25600 iterations (100 filter lengths) should lead to a convergence of 40dB. This is true if the identification gain constant μ (the variable MUident) is optimally adjusted, and only until the identification noise *ERR* reaches the system noise.

For the default value of MUident (15000) to lead to a good identification, the level returned by the microphone during identification phase should be around 100 mV (peak to peak).

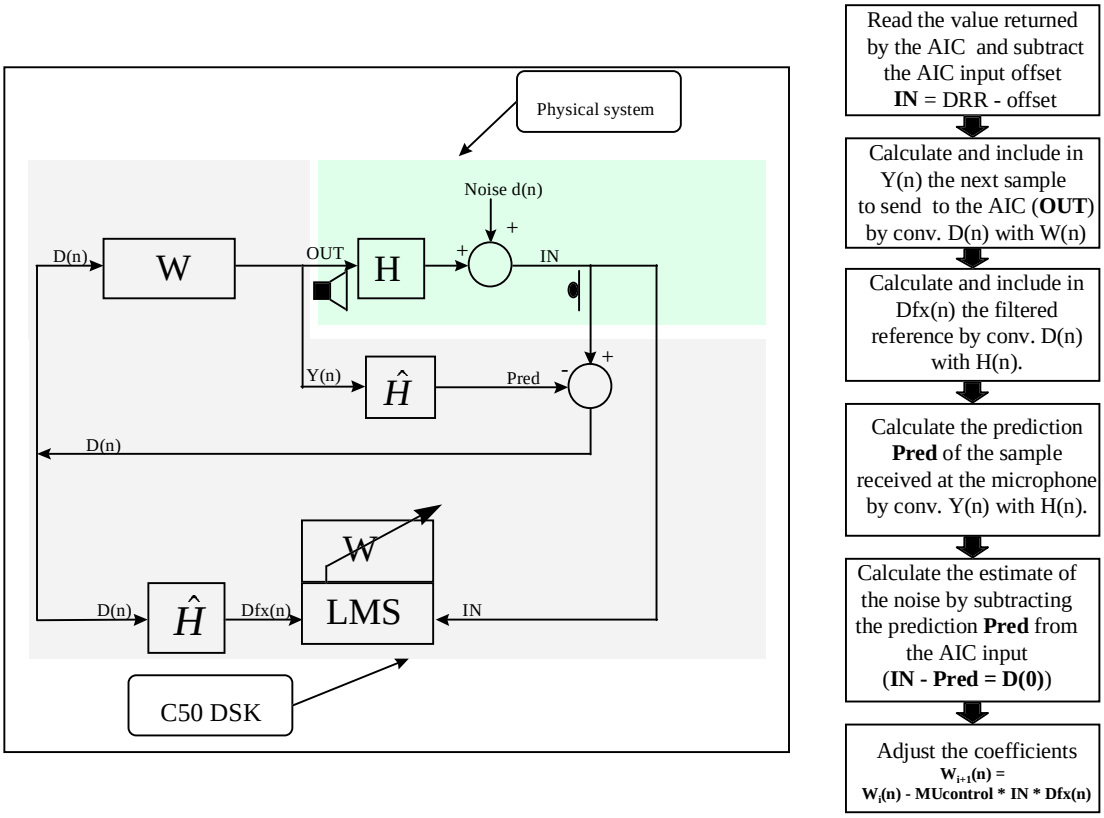
For a signal returned by the microphone *N* times smaller/higher than 100 mV, the MUident value should be multiplied/divided by N^2.

The identification process takes place in phase 2 of the interrupt routine.

Control

After completing *ITER* iterations (interrupts) in the identification phase, the control phase starts. This phase is the last one of the process and executes until the DSP is reset. The goal of this phase is to cancel the noise at the microphone by adapting the coefficients of the control filter *W* (see Figure 9). The adaptation process is continuous to track changes in the noise signal characteristics.

Figure 9. Steps of the Control Phase, Executed After the Identification and Until a Reset of the DSP



For optimal control, the control gain constant (the variable MUControl) needs to be adjusted. The default value (6000) should lead to a good control performance if the level of the disturbance signal returned by the microphone is around 100 mV (peak to peak) with no control.

If your microphone is returning a higher amplitude signal, you may have to lower the gain constant and vice versa. If you cannot modify the gain constant any more (you have reached the maximum or minimum value), you may have to modify the normalization of some calculations in the program.

The parameter MUControl adjusts the step size of the filtered-X LMS algorithm. Giving it a large value will lead to a faster convergence, up to a point where the algorithm will diverge. At this point, full range random noise will be sent to the speaker, which can be quite annoying.

A good way to obtain the best value for MUControl is to find by trial and error the smallest value making the algorithm diverge. Dividing by two will yield the value that makes the algorithm converge the most rapidly. Dividing by two again will yield a value making the algorithm converge half as fast but to a 3 dB lower residual noise.

Two arrays are used for the control filter (*W*). One is a 16 bit array (every tap of the filter is on 16 bits); the other is a 32 bit array. Both contain the 16 taps of the filter, one of which has more resolution.

The 32 bit version of the filter is used for the update of the coefficients; the 16 bit version is used for the convolutions. This allows a better resolution of the coefficients for the adaptation process.

The 16 bit array is updated by keeping only the 16 MSBs (most significant bits) of each tap of the 32 bit array. The 32 bit array is useful in the sense that it allows smaller values coming in (IN) to have an influence on the control filter coefficients.

This process takes place in phase 3 of the interrupt routine.

Performance

With proper signal levels and good speakers and microphone, you should achieve a 40 dB reduction in noise at the microphone, for a pure tone disturbance. With suggested values, the reduction is obtained within seconds of the start of the control phase.

The sampling rate can be increased to 19200 Hz (the maximum of the AIC), although it is not useful for acoustic noise cancellation because the size of the silent zone would be very small. This would allow a control to be effective on pure tones up to 9000 Hz.

Appendix A TMS320C50 Assembler Code

```
*******************************************************************************
*       TMS320C50 ASSEMBLER CODE FOR A SINGLE CHANNEL ACTIVE ADAPTIVE
*       NOISE CANCELLER
*******************************************************************************

         .mmregs

        .data    ;2000h, beginning of a data page
*******************************************************************************
***     The following global variables are for the serial port interrupt
***     routine and should be on the same data page in order to avoid constant
***     reloading of the "dp".
*******************************************************************************
ITER            .word   10240    ; Number of iterations for the Identification of H phase.
MUident .word   15000            ; Gain constant for the Identification of H.
MUControl       .word   6000             ; Gain constant for the Control optimization.
noislvl .word   2000             ; The level of the bipolar random noise in the
                                         ; Identification of H phase.
IN              .word   0
total           .word   0
counter .word   0
offset          .word   0
PHASE           .word   1               ;Beginning phase.
OUT             .word   0
ONE             .word   1
ERR             .word   0
Pred            .word   0
ERRxMU          .word   0
INxMU           .word   0
*******************************************************************************

****************************
*** AIC global variables ***
****************************
TA              .word   31               ; Registers of the AIC
RA              .word   31
TB              .word   32
RB              .word   32
AIC_CTR         .word   29h
VPRD            .word   16               ; PRD register of the C50
****************************

***************************************
*** Random Noise Generator variables ***
***************************************
seed            .word   07e6dh           ; seed for random noise generator
TEMP            .word   0
***************************************

**************
*** ARRAYS ***
**************
Wprog           .set    1c00h            ;W(15) address in program memory (necessary for MACD
instruction)
Wdata           .set    1c00h            ;W(15) address in data memory
Wdata32 .set    1c10h           ;W(15) address for 32 bit coefficients

Hprog           .set    1c30h            ;H(255) address in program memory (necessary for MACD
instruction)
Hdata           .set    1c30h            ;H(255) address in data memory

Y0              .set    1d30h            ;Y(0) address
Y255            .set    1e2fh            ;Y(255) address

Dfx             .set    1e30h            ;Dfx(-1) address
Dfx0            .set    1e31h            ;Dfx(0) address
Dfx15           .set    1e40h            ;Dfx(15) address
```

```
X0              .set    300h            ;X(0) address
X255            .set    3ffh            ;X(255) address
X256            .set    400h            ;X(256) address
                                        ;D(n) uses the same memory as X(n).
D0              .set    300h            ;D(0) address
D16             .set    30fh            ;D(16) address
D255            .set    3ffh            ;D(255) address

        .sect   "COEFF"         ;Reset of vectors W(n), H(n), Y(n) and Dfx(n).
        .bes    9232                    ;(1e41h-1c00h)words * 16bits = 9232 bits

        .sect   "REF"                   ;Reset of X(n) and D(n)
        .bes    4096


*********************************
***   Set up the ISR vectors
*********************************
        .sect   "VECTORS"
        .space 2 * 16
int2:           B       COMM

        .space 4 * 16
rint:           B        RINT           ;0A; Serial prot receive interrupt RINT.
xint:           B        TRANSMIT       ;0C; Serial port transmit interrupt XINT.



*******************************************************************************
*******************************************************************************
***
***                     MAIN
***
***                     1- DSP INITIALIZATION
***                     2- AIC INITIALIZATION
***                     3- ENABLE INTERRUPTS
***                     4- WAIT FOR INTERRUPTS
***
*******************************************************************************
*******************************************************************************

        .text

                SETC    INTM
        call    Init_DSP        ; DSP INITIALIZATION
        CALL    AICINIT  ; AIC INITIALIZATION
        call    Enab_INT        ; ENABLE INTERRUPTS


WAIT            nop                     ; WAIT FOR INTERRUPTS

                B       WAIT

***** End of main ************************

****************************************
***     TMS320C50 INITIALIZATION      ***
****************************************

Init_DSP        LDP     #0
        OPL     #0834h,PMST
        LACC    #0
        SAMM    CWSR
        SAMM    PDWSR
        setc    OVM              ; OVM = 1
        SPM     0                ; PM = 0
        setc    SXM
                setc    CNF             ; CNF = 1
```

```
        RET

***** End of DSP INITIALIZATION ***********

****************************************
***     AIC INITIALIZATION
****************************************

AICINIT:        SPLK    #022h,IMR       ; Using XINT syn TX & RX

        SPLK    #20h,TCR        ; To generate 10 MHz from Tout.
        lmmr    PRD,VPRD
        MAR     *,AR0
        LACC    #0008h          ; Non continuous mode
        SACL    SPC             ; FSX as input
                LACC    #00c8h          ; 16 bit words
        SACL    SPC
        LACC    #080h           ; Pulse AIC reset by setting it low
        SACH    DXR
        SACL    GREG
        LAR     AR0,#0FFFFh
        RPT     #10000          ; and taking it high after 10000 cycles
        LACC    *,0,AR0         ; (.5ms at 50ns)
        SACH    GREG            ; Restore GREG to 0000

        lacc    #2448h          ; In case the AIC is by error waiting for a
        sacl    DXR             ; secondary communication, we send the default
        splk    #20h,IFR        ; TA and RA values (18h).  The AIC returns in primary
         ; communication state after.  To Solve a problem on the serial
         ; port of some C50

        ;------------------------
        LDP     #TA
        SETC    SXM
        LACC    TA,9            ; Initialized TA and RA register
        ADD     RA,2
        CALL    AIC_2ND
        ;------------------------
        LDP     #TB
        LACC    TB,9            ; Initialized TB and RB register
        ADD     RB,2
        ADD     #02h
        CALL    AIC_2ND
        ;------------------------
        LDP     #AIC_CTR
        LACC    AIC_CTR,2       ; Initialized control register
        ADD     #03h
        CALL    AIC_2ND

        RET

AIC_2ND:        LDP     #0
        sach    DXR

intaic0 bit     IFR,10
        bcnd    intaic0,NTC
                splk    #20h,IFR

        ADD     #6h,15          ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        sach    DXR


intaic1 bit     IFR,10
                bcnd    intaic1,NTC
        splk    #20h,IFR

                SACL    DXR
```

```
intaic2 bit     IFR,10
        bcnd    intaic2,NTC
        splk    #20h,IFR

        lacl    #0
                SACL    DXR

intaic3 bit     IFR,10
        bcnd    intaic3,NTC
                splk    #20h,IFR

        RET

***** End of AIC INITIALIZATION ***********


****************************************
***     ENABLE INTERRUPTS              ***
****************************************

Enab_INT        SPLK    #012h,IMR       ; enable RINT and INT2
        CLRC    INTM            ; enable int.

        RET

***** End of ENABLE INTERRUPTS ************



*******************************************************************************
*******************************************************************************
***
***                           INTERRUPT ROUTINES
***
*******************************************************************************
*******************************************************************************

*******************************************************************************
***     C50 serial port receive interrupt routine (RINT)
*******************************************************************************

RINT:           ldp     #OUT            ; Send the value in "OUT" to
        lacl    OUT             ; the Data Transmit Register (DXR)
        samm    DXR

        cpl     #01,PHASE
        bcnd    Phase1,TC       ; AIC OFFSET CALCULATION
        cpl     #02,PHASE
                bcnd    Phase2,TC       ; IDENTIFICATION OF H
        cpl     #03,PHASE
        bcnd    Phase3,TC       ; CONTROL


************************************************
***     AIC OFFSET CALCULATION ---> PHASE 1
************************************************

Phase1          lacl    #0              ; Send 0 to the serial port transmit register
        sacl    OUT             ; while calculating the offset.
        lacl    #1
        add     counter
        sacl    counter
        lamm    DRR
        sacl    IN
        lacl    #200            ; Let go the first 200 samples in order for the
        NEG                     ; AIC to stabilize.
        add     counter
        bcnd    END_IN,LT
```

```
        lacl    IN
        add     total           ; Add 64 samples in "total"
        sacl    total
        cpl     #263,counter
        bcnd    END_IN,NTC
        lacc    total
        bsar    6               ; Divide total per 64 to calculate the offset average.
        sacl    offset
        lacl    #0
        sacl    counter
        sacl    total
        lacl    #2              ; Switch to the Identification Phase (phase #2)
        sacl    PHASE
        B       END_IN

*** End of AIC Offset Calculation phase ***


*************************************************************************
***     IDENTIFICATION OF H PHASE ---> PHASE 2
***
***     Global Variables to configure:
***     ITER: Number of iterations for the Identification of H phase.
***     MUident: Gain constant for the Identification of H.
***     noislvl: The level of the bipolar random noise.
***
*************************************************************************


Phase2
        ;Read the value returned by the AIC and subtract the AIC input offset: IN = DRR -
offset

        lamm    DRR             ; Read the value returned by the AIC
        sub     offset          ; Subtract AIC input offset
                sacl    IN


        ; Calculate and include in X(n) the next sample (White Noise) to send to the AIC
(OUT)

                LACC    seed,1  ; Random noise generator
                XOR     seed
                SACL    TEMP,2
                XOR     TEMP
                AND     #8000h
                ADD     seed,16
                SACH    seed,1  ; Reduce the output by at least 1/8
                LACC    seed,11         ; to prevent the overflow
        ROL             ; Depending of the sign of the random output,
        lacc    noislvl ; + or - noislvl is sent to the AIC (bipolar random noise).
        and     #0fffch ; Force the two LSBs to 0 in order for the AIC to see the
        bcnd    POS,NC          ; value as a data and not a command.
        neg

POS             sacl    OUT             ; Store in OUT the next interrupt DXR sample (the next
value
                                        ; to send to the serial port transmit register).
                lar     ar3,#X0 ; Include in X(n) the next sample to send.
        mar     *,ar3
        sacl    *



        ; Convolution of H(n) with X(n) (Pred)

        zpr
        lacc    #1,15
```

```
        mar     *,ar3
        lar     ar3,#X255
        FIR     rpt     #255
        macd    Hprog,*-
        apac
        sach    Pred

        ; Subtract the prediction from the AIC input (IN - Pred = ERR)

                neg
        lar     ar3,#IN
        add     *,16
        sach    ERR

        ; Adjust the coefficients Hi+1(n) = Hi(n) + MUident * ERR * X(n)

                lt      ERR             ; ERR * MUident
        mpy     MUident
        pac
        add     ONE,13          ; Rounding
        sach    ERRxMU,2

                lacc    #254            ; Update the 256 Coefficients.
        samm    BRCR

                lar     ar2,#Hdata
        lar     ar3,#X256
        lt      ERRxMU
        mpy     *-,ar2

                rptb    LOOP-1
ADAPT           zalr    *,ar3
                mpya    *-,ar2
        sach    *+              ;Store the new H coefficient.
LOOP            zalr    *
        apac
        sach    *+              ;Store the last new H coefficient.

        lacl    #1
        add     counter
        sacl    counter

INCR            lacl    ITER
        samm    DBMR
        cpl     counter
        bcnd    END_IN,NTC
        lacl    #0
        sacl    OUT
        sacl    counter
        mar     *,ar3
        lar     ar3,#X0
        rpt     #256
        sacl    *+,0
        lacl    #3
        sacl    PHASE

        B       END_IN

**** End of Identification of H  phase ****



*************************************************************************
***     CONTROL PHASE ---> PHASE 3                                    ***
***     Global Variables to configure:                                    ***
***     MUControl: Gain constant for the Control.                     ***
***                                                                   ***
*************************************************************************
```

```
Phase3
        ; Read the value returned by the AIC and subtract the AIC input offset: IN = DRR -
offset

        lamm    DRR             ; Read the value returned by the AIC
        sub     offset          ; Subtract AIC input offset
        sacl    IN

        ; Calculate and include in Y(n) the next sample to send to the AIC (OUT)
        ; by the convolution of D(n) with W(n)

debut           zpr
        lacc    #1,12           ; Rounding
        mar     *,ar3
        lar     ar3,#D16
FIRC            rpt     #15             ; Convolution
        mac     Wprog,*-
        apac
        and     #0fffch,13      ; Force the two LSBs of the next sample to send to 0
        sach    OUT,3
        lar     ar3,#Y0  ; Include the next sample to send in Y(n)
        sach    *,3

        ; Calculate and include in Dfx(n) the filtered reference
        ; by the convolution of D(n) with H(n).

        zpr
        lacc    #1,11           ; Rounding
        lar     ar3,#D255
FIRC1           rpt     #255            ; Convolution
        macd    Hprog,*-
        apac
        lar     ar3,#Dfx        ;Include the filtered reference in Dfx(n)
        sach    *,4
        lar     ar3,#Dfx15      ; Shift by one the filered reference vector.
        mar     *-,ar3          ; Since this vector is never used in a convolution,
        rpt     #14             ; it has to be shifted manually. (last in, last out)
        dmov    *-
                dmov    *

        ; Calculate the prediction "Pred" of the sample received at the microphone
        ; by the convolution of Y(n) with H(n).

        zpr
        lacc    #1,15           ; Rounding
        lar     ar3,#Y255       ; The normalization should be here the same as in the
FIRC2           rpt     #255            ; convolution (FIR) in the IDENTIFICATION OF H phase.
        macd    Hprog,*-        ; Convolution.
        apac
        sach    Pred

        ; Calculate the estimate of the noise by subtracting the prediction "Pred" from
        ; the AIC input: D(0) = IN - Pred

                neg
        lar     ar3,#IN
        add     *,16
        lar     ar3,#D0
        sach    *

        ; Adjust the coefficients: Wi+1(n) = Wi(n) - MUControl * IN * Dfx(n)

                lt      IN              ; IN * MUControl
        mpy     MUControl
        pac
        add     ONE,8           ; Rounding
        sach    INxMU,7
```

```
               lacl    #14
        samm    BRCR
        lar     ar2,#Wdata32    ; 32 bit Coefficients of W
        lar     ar3,#Dfx15      ; Filtered Reference
        lar     ar4,#Wdata      ; 16 bit Coefficients of W
        lt      INxMU
        mpy     *-,ar2
        spm     3               ; Causes transfers from the product register to the ALU
                                ; to be shifted to the right six places (divided per 64).
                                ; Since the result of "MUControl * IN * Dfx(n)" is divided
by 64,
                               ; "IN * MUControl" can take values 64 times higher. This
allow
                               ; non-zero results for IN values 64 times lower.

               rptb    LOO-1
ADAPTC         lacc    *+,16           ;Load the acc with a 32 bit coefficient of W.
        or      *-,ar3
        mpys    *-,ar4
        sach    *+,ar2          ;Store a new W coefficient into the 16 bit vector of W.
        sach    *+              ;Store a new W coefficient into the 32 bit vector of W.
        sacl    *+              ;
LOO             lacc    *+,16
        or      *-,ar4
        spac
        spm     0
        sach    *+,ar2          ;Store the last new W coefficient into the 16 bit vector of W.
        sach    *+              ;Store the last new W coefficient into the 32 bit vector of W.
        sacl    *+              ;
        B       END_IN

********* End of Control phase **********

END_IN          RETE

********** End of RINT routine **********


************************************************
***     Other Interrupt routines              ***
************************************************
COMM:           RETE
TRANSMIT:       RETE



        .end

/* Linker command file for DSK code   */
-o anc.out   /* Specify output file   */
-m anc.map   /* Generate map file     */

mxb.obj      /* Object file to link   */

MEMORY
{
      PAGE 0 : VECT:   origin = 00802h, length = 003dh
      PAGE 0 : PROG:   origin = 00a00h, length = 1200h
      PAGE 0 : UPG :   origin = 01c00h, length = 0300h
      PAGE 1 : DAT :   origin = 02000h, length = 0c00h
      PAGE 1 : B1  :   origin = 00300h, length = 0100h

}

SECTIONS
{
        .text   :       {} > PROG       PAGE 0
        VECTORS:        {} > VECT       PAGE 0
        COEFF   :       {} > UPG        PAGE 0
```

```
        .data   :       {} > DAT        PAGE 1
        REF     :       {} > B1 PAGE 1
}
```

References

[1] H.F. Olson and E. G. May, ”Electronic Sound Absorber,” *Journal of the Acoustical Society of America 25*, pp. 1130-1136, 1953.
[2] S.J. Elliott, P. Joseph, A.J. Bullmore and P.A. Nelson, ”Active Cancellation at a Point in a Pure Tone Diffuse Sound Field,” *Journal of Sound and Vibration 120*, pp. 183-189, 1988a.
[3] S.J. Elliott, T.J. Sutton, B. Rafaely and M. Johnson, ”Design of Feedback Controllers Using a Feedforward Approach,” *Proc. of Active 95*, pp.863-874, 1995.
[4] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.
[5] *TMS320C5x DSP Starter Kit User’s Guide*, Texas Instruments, pp. B-3 to B-34, 1994.
[6] *TMS320C50 User’s Guide*, Texas Instruments, p. 7-2, 1993.