TEXAS INSTRUMENTS

Implementation of Echo Control for ITU G.165/DECT on TMS320C62xx Processors

Application Report

August 1999

Wireless ASP Products

SPRA576

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products 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, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Contents

1 Overview 1

2 G.165 and DECT Echo Control 2

3 Telecom Levels in Echo Cancellers 4

3.1 Echo Cancellation Algorithm 4

3.1.1 Echo Prediction 4

3.1.2 Predictor Update 6

3.1.3 Echo Suppressor 7

3.1.4 Nonlinear Processor 8

3.1.5 Modem Answer Tone Detection 8

3.2 Performance Limitations for Echo Cancellation 9

3.2.1 Linear Signal Noise 10

3.2.2 Codec Companding Distortion 10

3.2.3 Convergence Noise 10

3.2.4 Leakage Noise 10

4 Echo Canceller Performance 11

5 Description 13

5.1 RAM Requirements 13

5.2 ROM Requirements 13

5.3 MIPs Requirements 14

6 Software Routines 15

6.1 C-Code and C-Callable Subroutines 15

6.1.1 TestEchoC6x.c 15

6.1.2 C External Functions 17

6.2 Echo Control Assembler Code 19

6.2.1 Echoc6xxx.asm 19

6.2.2 Echoc6xxx.asm16 20

6.2.3 Echoc6xxx.asm32 21

6.2.4 ModemTone.asm 22

6.2.5 ToneGenerate.asm 22

7 References 23

List of Figures

1 G.165 Echo Control on Both Ends of International Link 2
2 DECT Echo Control Software Block 3
3 Modem Answer Tone Response Curves 9

List of Tables

1 Digital Codec Levels 4
2 Normalized Digital Codec Levels 4
3 Echo Canceller Leak Rates 7
4 4-ms Echo Tail Using 16-Bit Filter Weights (1/2 Second) 11
5 8-ms Echo Tail Using 16-Bit Filter Weights (1/2 Second) 11
6 16-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second) 11
7 32-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second) 12
8 64-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second) 12
9 32-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second) 12
10 64-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second) 12
11 Calculating Memory Requirements 13
12 Worked Example Memory Requirements 13
13 Program Memory Requirements 13
14 CPU Cycle Requirements 14
15 CPU Cycles/MIPs Requirements 14
16 Global Echo Variables 15
17 ChannelEcho Variables 16
18 ChannelModem Variables 16
19 ToneGen Variables 17
20 16-Bit Cancellation z0 Word Aligned Pipeline 20
21 16-Bit Cancellation z0 Not Word Aligned Pipeline 20
22 16-Bit Update Pipeline 21
23 32-Bit Cancellation Pipeline 21
24 32-Bit Update Pipeline 22

Implementation of Echo Control for ITU G.165/DECT on TMS320C62xx Processors

ABSTRACT

This application note describes the implementation of multichannel echo control on a TMS320C62xx processor core that can cancel echoes with a 4–32-ms delay. This software is based on the leaky normalized least mean square (LMS) filter with either 16- or 32-bit weighting coefficients.

1 Overview

The 4-wire to 2-wire hybrid used in analog telephone transmission introduces an echo into the received part. When this echo is less than a few milliseconds, it provides the telephone user with assurance that the telephone is working. As the echo extends beyond a few milliseconds, however, it becomes distracting and causes the user to slow his speech. With the installation of more digital echoless equipment into the telephone network, the distance between the user and the hybrid increases, resulting in an increase of the echo delay. It is therefore necessary to cancel the long delay echo and to provide a short delay comfort echo so that users hear the echo they are accustomed to.

By changing some of the update parameters, you can easily adapt the application software described here to handle longer echoes.

All code is C-callable optimized assembler code, and includes subroutines for both the International Telecommunications Union (ITU) G.165 recommendation and the Digital European Cordless Telecommunication (DECT) approaches to echo control.

2 ITU G.165 and DECT Echo Control Recommendations

Both of these recommendations use basically the same specification for the electrical line echo canceller; the difference between them is in the handling of the residual signal left after cancellation and the reintroduction of the comfort noise. In ITU recommendation G.165, residual echo is removed by a nonlinear processor (NLP), which basically mutes all signals below a given threshold, as shown in Figure 1. In the DECT recommendation, residual echo is suppressed using an echo suppressor that suppresses far-end speech by 9–12 dB when the near-end user is talking, as shown in Figure 2. In the ITU G.165 recommendation, comfort noise is typically introduced in the same echo controller that cancels the data, whereas the DECT recommendation takes a system view and introduces the comfort noise for its partner echo control at the other end of the radio frequency (RF) link. The DECT approach gives shorter echoes for further cancellers, but requires a common control strategy for both ends of the link, whereas the ITU G.165 approach allows both ends to have different strategies. Examples of ITU G.165 and DECT echo controls are shown in Figure 1 and Figure 2, respectively. In both figures, the echo control blocks are shown in bold boxes, other DSP functions are shown in nonbold boxes, and external blocks are denoted by boxes with dashed lines. The A/D/A and hybrid are replaced by integrated services digital network (ISDN)-style links in some systems. Sidetone (ST) is sometimes referred to as comfort noise.

While the removal of a distant echo and replacement by a nearer echo is good for the human perception of speech, the distortion introduced can upset the modem and its built-in controllers. For this reason, modems and other equipment that require no external echo control transmit a disabling tone at the start of a call. This tone is detailed in ITU recommendations V.25 and ITU G.164. This application code also contains routines to detect this tone, though the use of this information to control the actual routines is the responsibility of the end user because this control varies from application to application.

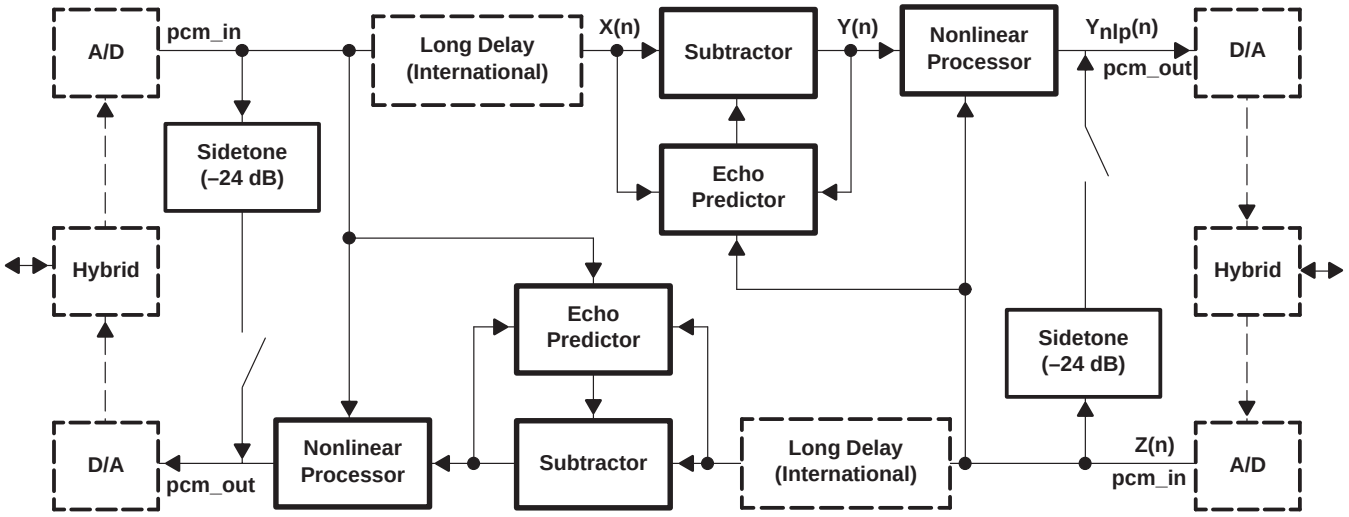

Figure 1. ITU G.165—Echo Control on Both Ends of International Link

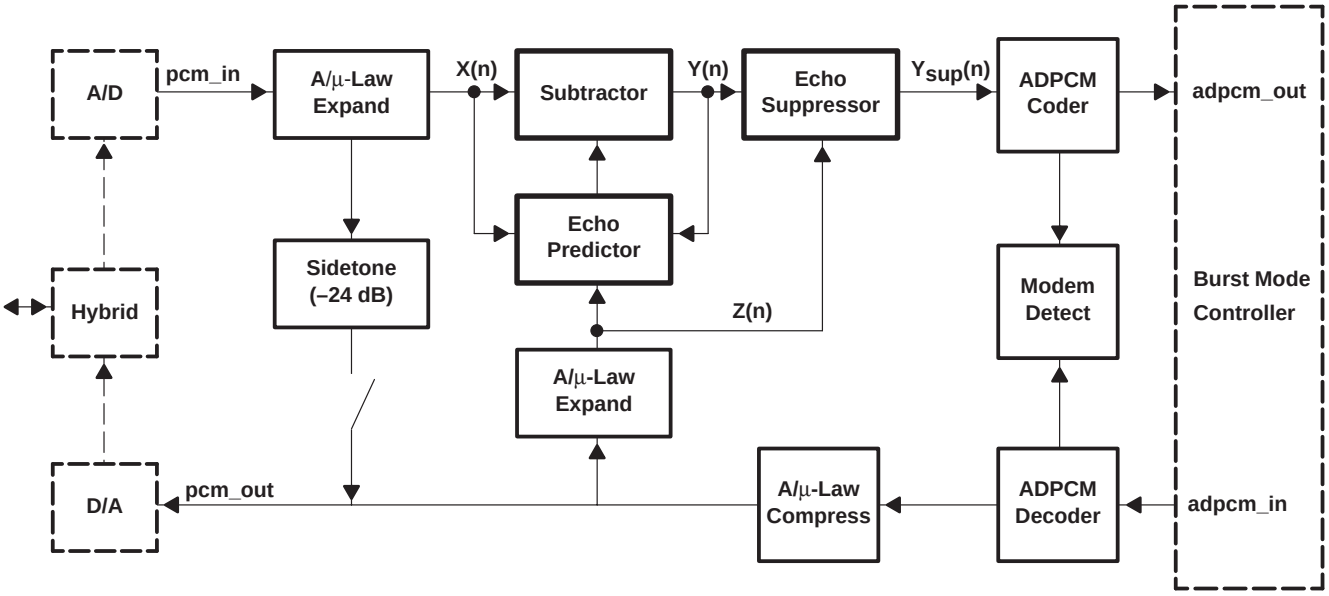

Figure 2. DECT Echo Control Software Block

3 Telecom Levels in Echo Cancellers

Voice data is represented in telecommunications by discrete digital levels that can be linear or logarithmic coded. Most world communications are implemented using logarithmic code that complies with either the A-Law or μ-Law standard specified in ITU recommendation G.711. For voice coding, these values must be linearized to produce linear digital code because different algorithms linearize to different resolutions. The digital equivalents defined in ITU G.711 for A-Law or μ-Law are shown in Table 1.

Table 1. Digital Codec Levels

LAW	MIN	MAX
A-Law	–4032	4032
μ-Law	–8031	8031

To provide uniform levels into voice coders like ITU G.726, A-Law levels are multiplied by 2 to produce 14-bit 2s complement or Q13 numbers, as shown in Table 2.

The 0-dBm point is defined as the root mean square (rms) level of a sine wave 3 dB below the clipping level; therefore, the mean amplitude of the 0-dB point is half the maximum level.

Table 2. Normalized Digital Codec Levels

LAW	MIN	MAX	0-dBm POINT
A-Law*2	–8064	8064	4032
μ-Law	–8031	8031	4016
Linear	–8192	8191	4095

3.1 Echo Cancellation Algorithm

This section describes and illustrates the calculations used for the echo predictor, the echo suppressor, the nonlinear processor alternative to echo suppression, and modem answer tone detection.

3.1.1 Echo Prediction

The general algorithm of a transversal finite impulse response (FIR) predictor is shown in Equation 1 and Equation 2.

$$Yp(n) = \sum_{k=0}^{31} W_k(n)Z(n - k) \quad (1)$$

where:

$W_k(n)$ is the weighting factor for each FIR (k) parameter at time (n)

Z(n) is the transmitted input signal at time (n)

Yp(n) is the predicted echo on the received signal at time (n)

$$Y(n) = X(n) - Yp(n) \quad (2)$$

where:

X(n) is the received echo signal at time (n)

Y(n) is the echo-cancelled received signal at time (n)

Using this predicted signal, the predicted echo is subtracted or canceled from the return path, as shown in Equation 2, to produce the predicted error or echo-cancelled signal. When there is no signal from the near end, this signal is the error in the echo predictor. When there is a signal from the near end, this signal represents the near-end signal with the echo from the far end canceled out. To properly update the predictor, it is necessary to know if there is a signal from the near end. The algorithm, therefore, contains a number of conditions that must be met before the predictor is updated.

1. The signal being sent from the far end must be above a certain threshold; that is, there must be speech at the far end for there to be an echo capable of being canceled.
2. The signal being received from the near end must be significantly less than the signal being sent from the far end; that is, the predictor must not be updated during double talk.
3. Continuous jitter of the predictor variables can cause noise/distortion to the reception of the near-end signal; if the noise of the echo is less than the noise caused by the update of the predictor, do not update the predictor.

The near-end, far-end and, echo-cancelled signal levels are continuously monitored using the infinite impulse response filters with a short time constant so that zero-crossovers are not rejected as not being speech. The filter rapidly detects short silences in the speech. The equations for both these filters are shown in Equation 3. These infinite impulse response (IIR) filters use a simple single-pole decay that can be implemented using the only shifts without the need for multiplication.

$$Signal_{Level}(n) = Signal_{Level}(n-1) + \left[\frac{|Signal_{Input}(n)| - Signal_{Level}(n-1)}{32}\right] \quad (3)$$

By comparing the results of this equation for Z(n), a suitable threshold for speech transmission can be determined so that the predictor is updated only when the level exceeds this threshold. Double talk is detected by comparing the signal on both these filters; when the Z(n) filter is sufficiently greater than the Y(n) filter, the predictor can safely be updated. Alternatively, the scaling factor for the update can be based on the relative strengths of the two signals, but this can cause problems if the echo signal is too strong when the algorithm is initially reset.

3.1.2 Predictor Update

To update the predictor, it is necessary to change the W_k parameters of the predictor. The faster these parameters are changed, the faster the filter converges onto the actual echo. If these parameters are changed too rapidly, a significant distortion is introduced to the signal received from the lines. This distortion can cause so much feedback that the filter can become unstable. It is therefore necessary to reach a compromise between the update rate and the distortion introduced to the signal. Based on the calculations in Equation 1 and 2, the error in the predicted signal can be shown in Equation 4.

$$Y(n) = X(n) - \sum_{k=0}^{31} W_k(n) * Z(n - k) \tag{4}$$

Although this subject is beyond the scope of this document, the update for this predictor can be performed using the LMS algorithm, as shown in Equation 5.

$$W_k(n + 1) = W_k(n) + u * Y(n) * Z(n - k) \tag{5}$$

Because the delay to the echo being canceled is not a rapidly changing variable, it is not necessary in this application to maximize the convergence rate of the echo canceller. The u constant can be made relatively small to generate an echo canceller that produces a low level of distortion. The update rate of this filter is proportional to both the input signal level and the echo signal level, and the filter tends to converge faster for high signal levels. This rapid convergence can be counteracted by normalizing the update rate to the level of these input signals. The echo is proportional to the input signal so the normalization can be to the input signal $Z(n)^2$. Do not place too much emphasis on individual samples because when there are near zero crossings, the error can be large compared to the signal level, and the signal from the X(n) input is delayed by the echo. To avoid these problems, the predictor update uses the mean input signal level for the Z(n) variable rather than the instantaneous signal level. When the normalization constant given in Equation 6 is substituted in Equation 4, this results in Equation 7. Equation 7 also shows the actual constant used to set the convergence stability compromise.

$$\text{Normalization} = 2^{\text{INT}(2*\text{LOG}_2(Z_{\text{Level}}))} \tag{6}$$

$$W_k(n + 1) = W_k(n) + \left\{ \frac{2^{\text{INT}(2*\text{LOG}_2(Z_{\text{Level}}(n)))} * Y(n) * Z(n - k)}{512} \right\} \tag{7}$$

To stabilize the filter during tones and prevent overflows, a leak of 1/m is introduced into the update. The value of this leak depends on the level of cancellation achieved. The signal level before and after cancellation is measured using Equation 3, and the function shown in Equation 8 determines the approximate amplitudes.

$$\text{INT}(\text{LOG}_2(\text{Signal}_{\text{level}})) \tag{8}$$

The difference between these two logs provides the cancellation in 6-dB steps. This is translated to a leak rate according to the values shown in Table 3.

Table 3. Echo Canceller Leak Rates

MEASURED CANCELLATION	LEAK (16-BIT WEIGHTS)	LEAK (32-BIT WEIGHTS)
>36 Db		...
>24 dB	1/32768	
30...36 dB		1/65536
24...30 dB		1/32768
18...24 dB	1/16384	1/16384
12...18 dB	1/8192	1/8192
6...12 dB	1/4096	1/4096
0...6 dB	1/2048	1/2048
–6...0 dB	1/1024	1/1024
...	...	...

This leak rate then becomes Equation 9.

$$W_k(n+1) = \left(\frac{W_k(n) * (m-1)}{m}\right) + \left[\frac{2^{INT(2*LOG_2(Z_{Level}(n)))} * Y(n) * Z(n-k))}{512}\right] \quad (9)$$

In terms of the actual inputs, the overall equation for the update of the predictor coefficients is shown in Equation 10.

$$W_k(n+1) = \left(\frac{W_k(n) * (m-1)}{m}\right) + \left[\frac{2^{INT(2*LOG_2(Z_{Level}(n)))} * \left(X(n) - \sum_{l=0}^{31} W_l(n) * Z(n-l)\right) * Z(n-k)}{512}\right] \quad (10)$$

Overall, this predictor converges onto the echo in about 250 μs, nearly cancelling the echo down to the floor level of the system. To further reduce the echo, an echo suppressor (DECT) or a nonlinear processor (ITU G.165) is required.

3.1.3 Echo Suppressor

The echo suppressor can be programmed to give various echo suppressions from 0 dB to 24 dB. The programmed echo suppression is soft switched in and out with an attach time of 5 ms and a decay time of 2.5 ms. If full suppression is reached, there is a hangover time of 70 ms after the end of speech before the echo suppressor is switched out.

In the echo suppressor, speech is detected using a simple 4th-order FIR on the absolute values of the speech being sent to the line. The calculation for this filter is shown in Equation 11.

$$\begin{aligned} &\text{If } \Sigma_{k=0}^{7}|Z(n-k)| > \text{Supp\_Thresh} \\ &\text{And } \Sigma_{k=24}^{31}|Z(n-k)| > \text{Supp\_Thresh} \\ &\text{Then Suppressor}_{\text{Charge}}(n) = \text{Attack\_Constant} \\ &\text{Else Suppressor}_{\text{Charge}}(n) = \text{Decay\_Constant} \end{aligned} \quad (11)$$

The $\text{Suppress}_{\text{Change}}$ is used to drive an asymmetric 2nd-order IIR filter to generate the actual suppress factor. This IIR is subject to a hangover time of 70 ms, which is initiated when echo suppression has reached the maximum value. There is no hangover if echo suppression does not get fully turned on. The calculation for the IIR is shown in Equation 12. The time constant for this filter is altered between the attach and the decay by varying the constant, k1.

$$\text{Suppress}(n) = \text{Suppressor}_{\text{Charge}}(n) + k1 * \text{Suppress}(n-1) + k2 * \text{Suppress}(n-2) \quad (12)$$

After filtering, suppress has a range of 0 < Suppress < Supp_Thresh, where Supp_Thresh is the fractional part of the signal to be suppressed. This value is then subtracted from 1, and the result multiplied by L_{RES} to give the suppressed output. The calculation for this suppression function is shown in Equation 13.

$$Y_{SUP}(n) = Y(n) * [1 - \text{Suppress}(n)] \quad (13)$$

3.1.4 Nonlinear Processor

As an alternative to echo suppression, the echo-cancelled signal can be further reduced by a nonlinear processor; in this case, the output is muted when the cancelled signal falls below a given threshold, as shown in Equation 14.

$$\begin{aligned} &\text{if } Z_{\text{level}}(n) < \text{Threshold} \\ &\text{then } Y_{NLP}(n) = 0 \\ &\text{else } Y_{NLP}(n) = Y(n) \end{aligned} \quad (14)$$

3.1.5 Modem Answer Tone Detection

In addition to cancellation, it is necessary to detect the 2100-Hz modem answer tone in applications where modem data may be used instead of voice data. This is more common in ITU G.165-type echo controllers than in DECT-type echo controllers. However, when DECT echo controllers are used in radio local loop applications, modem answer tone detection is also needed. Answer tone detection is performed using two basic 2nd-order IIR filters to measure the tone energy and the output of band energy, respectively. Phase reversals of the tone are detected as short negative blips in the IIR energy of the detected tone. A phase reversal will give zero energy out of an IIR because the predicted signal is the exact opposite of the real signal, causing cancellation. The basic IIR calculation is Equation 15.

$$y_n = x_n * b_0 + x_{n-1} * b_1 + x_{n-2} * b_2 + y_{n-1} * a_1 + y_{n-2} * a_2 \quad (15)$$

The performance graphs of the band-pass and band-stop filters are shown in Figure 3.

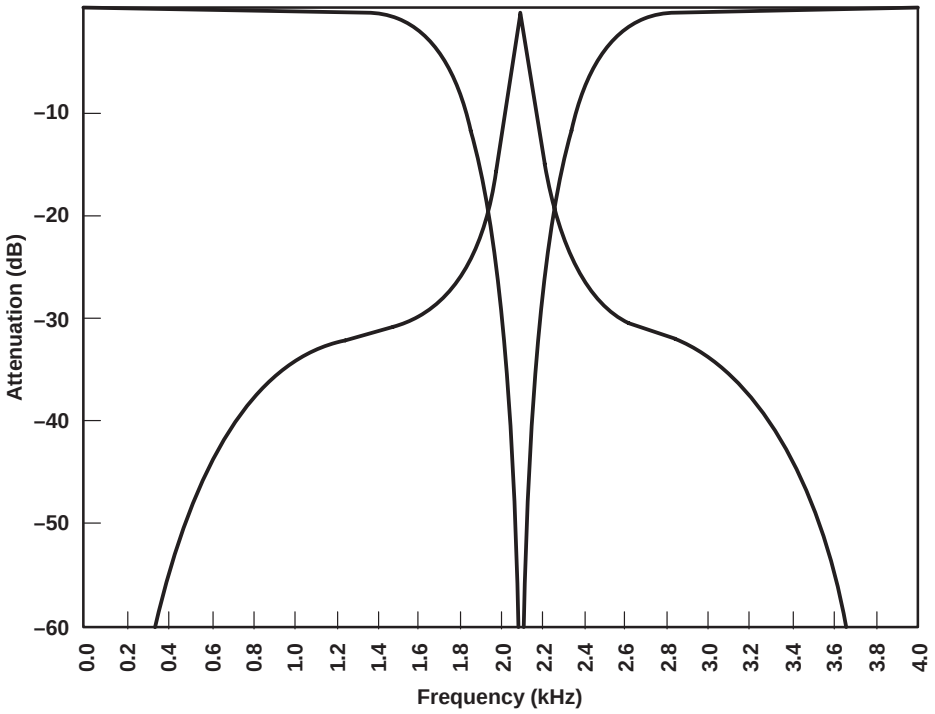

Figure 3. Modem Answer Tone Response Curves

The output level of these filters is measured using a 1st-order IIR filter of the absolute signal. The calculations for both of these filters are shown in Equation 16. These IIR filters use a simple single-pole decay that can be implemented using only shifts without the need for multiplication.

$$\text{Signal}_{\text{Level}}(n) = \text{Signal}_{\text{Level}}(n-1) + \left[\frac{|\text{Signal}_{\text{Input}}(n)| - \text{Signal}_{\text{Level}}(n-1)}{16}\right] \quad (16)$$

By comparing the signal level in the pass filter with a threshold and the two filters, the tone can be detected. Further detection is then based on whether the presence of the tone is sufficient for the samples to be valid.

3.2 Performance Limitations for Echo Cancellation

The performance of the echo canceller can be limited by any of these four factors:

- Linear signal noise
- Code-decode (codec) companding distortion
- Convergence noise
- Leakage noise

3.2.1 Linear Signal Noise

Linear signal noise comes from the ability to represent the output signal within the echo canceller. This is a resolution limit, and as such is 1/2 the LSB * the 0-dB level or 20*log (4095*2) = 78.2 dBm. Achieving this result with linear data provides a good indication of the integrity of a simulated echo canceller, but for real systems, one of the remaining three factors usually dominates.

3.2.2 Codec Companding Distortion

The codec companding distortion is the noise specified for an ideal codec at 33 dB in the ITU G.711 recommendation. For a true codec, ITU G.712 specifies 31 dB. This limits the ability of an echo canceller to cancel to 31–33 dB codec (depending on the quality of the codec used), and this is fundamental to all telephone systems.

3.2.3 Convergence Noise

This is the noise introduced into the output signal by the changes to the weighting parameters. This noise can be limited by either the accuracy of the error signal or the accuracy of the weighting signal. In practice with 16-bit weighting factors, this noise is limited by the minimum error needed to change the weights by one, and is Error x Length of the Predictor Affect on the Predictor. With 32-bit weights, the minimum error is that of the input signal resolution; therefore, both the error and the effect are reduced, providing a significant performance improvement.

Generally, if the echo tail is more than 8 ms, this noise exceeds the codec companding distortion. This is the limiting factor in choosing to use single- or double-precision weighting factors.

3.2.4 Leakage Noise

Certain tones cause an echo canceller to become unstable; this is counteracted by adding a leakage factor to the canceller. This leakage factor also reduces the effective length of the canceller by reducing the convergence noise of near-zero values. Although it adds noise to the larger values overall, this distortion is provided by the length x leakage. For an 8-ms canceller, the length is 64 samples, and the minimum leakage that can be obtained is 3/32768. The leakage degradation is therefore 54 dB. When artificial hybrids are cancelled with delay values of zero, leakage noise dominates instead of convergence noise.

4 Echo Canceller Performance

The echo canceller algorithm has been simulated using the TMS320C62xx simulator using tests similar to those described in the ITU G.165 recommendation for a variety of echo delays, signal levels, and echo types. These experimental results are summarized in Table 4 through Table 10.

Table 4. 4-ms Echo Tail Using 16-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.8 dBm	–49.0 dB	–56.5 dBm	–45.7 dB	–59.2 dBm	–44.6 dB	–64.1 dBm	–43.8 dB	–69.3 dBm
–6.8 dBm	–49.1 dB	–62.6 dBm	–46.1 dB	–65.6 dBm	–44.9 dB	–70.4 dBm	–42.0 dB	–73.6 dBm
–12.9 dBm	–48.9 dB	–68.4 dBm	–46.1 dB	–71.6 dBm	–43.4 dB	–74.9 dBm	–38.2 dB	–75.8 dBm
–18.9 dBm	–48.2 dB	–73.8 dBm	–44.7 dB	–76.3 dBm	–39.0 dB	–76.5 dBm	–32.0 dB	–75.6 dBm
–24.9 dBm	–46.3 dB	–77.9 dBm	–40.9 dB	–78.5 dBm	–34.1 dB	–77.7 dBm	–27.3 dB	–76.8 dBm
–30.9 dBm	–42.4 dB	–79.9 dBm	–34.3 dB	–77.8 dBm	–27.0 dB	–76.6 dBm	–21.9 dB	–77.5 dBm

Table 5. 8-ms Echo Tail Using 16-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.8 dBm	–48.6 dB	–56.1 dBm	–45.9 dB	–59.4 dBm	–44.1 dB	–63.6 dBm	–42.4 dB	–68.0 dBm
–6.8 dBm	–48.7 dB	–62.2 dBm	–45.7 dB	–65.2 dBm	–44.0 dB	–69.5 dBm	–42.5 dB	–74.0 dBm
–12.9 dBm	–47.9 dB	–67.4 dBm	–46.5 dB	–72.0 dBm	–43.3 dB	–74.9 dBm	–37.9 dB	–75.5 dBm
–18.9 dBm	–48.6 dB	–74.2 dBm	–44.6 dB	–76.2 dBm	–39.0 dB	–76.6 dBm	–32.0 dB	–75.6 dBm
–24.9 dBm	–46.9 dB	–78.5 dBm	–40.5 dB	–78.1 dBm	–33.8 dB	–77.4 dBm	–26.7 dB	–76.3 dBm
–30.9 dBm	–42.1 dB	–79.7 dBm	–34.4 dB	–78.0 dBm	–27.5 dB	–77.1 dBm	–21.7 dB	–77.3 dBm

Table 6. 16-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.8 dBm	–69.1 dB	–76.6 dBm	–64.5 dB	–78.0 dBm	–60.1 dB	–79.7 dBm	–49.9 dB	–75.5 dBm
–6.8 dBm	–64.7 dB	–78.2 dBm	–61.9 dB	–81.4 dBm	–51.1 dB	–76.7 dBm	–47.0 dB	–78.5 dBm
–12.9 dBm	–59.3 dB	–78.9 dBm	–50.4 dB	–76.0 dBm	–47.0 dB	–78.5 dBm	–40.7 dB	–78.3 dBm
–18.9 dBm	–51.1 dB	–76.7 dBm	–47.4 dB	–79.0 dBm	–39.8 dB	–77.4 dBm	–33.9 dB	–77.5 dBm
–24.9 dBm	–48.3 dB	–79.9 dBm	–41.1 dB	–78.7 dBm	–33.3 dB	–76.9 dBm	–26.9 dB	–76.5 dBm
–30.9 dBm	–41.4 dB	–78.9 dBm	–32.8 dB	–76.4 dBm	–28.6 dB	–78.2 dBm	–25.5 dB	–81.1 dBm

Table 7. 32-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.8 dBm	–46.8 dB	–54.3 dBm	–46.7 dB	–60.2 dBm	–46.2 dB	–65.8 dBm	–44.5 dB	–70.0 dBm
–6.8 dBm	–46.8 dB	–60.3 dBm	–46.8 dB	–66.4 dBm	–44.2 dB	–69.8 dBm	–41.2 dB	–72.8 dBm
–12.9 dBm	–45.5 dB	–65.0 dBm	–44.1 dB	–69.7 dBm	–41.4 dB	–72.9 dBm	–38.1 dB	–75.7 dBm
–18.9 dBm	–42.8 dB	–68.4 dBm	–40.7 dB	–72.3 dBm	–37.1 dB	–74.7 dBm	–29.8 dB	–73.3 dBm
–24.9 dBm	–40.9 dB	–72.5 dBm	–36.7 dB	–74.3 dBm	–37.1 dB	–75.2 dBm	–25.8 dB	–75.3 dBm
–30.9 dBm	–26.9 dB	–64.5 dBm	–27.1 dB	–70.7 dBm	–24.0 dB	–73.6 dBm	–20.6 dB	–76.2 dBm

Table 8. 64-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.8 dBm	–18.4 dB	–25.9 dBm	–18.4 dB	–31.9 dBm	–18.4 dB	–37.9 dBm	–18.4 dB	–43.9 dBm
–6.8 dBm	–18.4 dB	–31.9 dBm	–18.4 dB	–37.9 dBm	–18.4 dB	–43.9 dBm	–18.4 dB	–49.9 dBm
–12.9 dBm	–18.3 dB	–37.8 dBm	–18.2 dB	–43.7 dBm	–18.2 dB	–49.8 dBm	–18.0 dB	–55.6 dBm
–18.9 dBm	–18.3 dB	–43.8 dBm	–18.2 dB	–49.8 dBm	–18.2 dB	–55.8 dBm	–17.8 dB	–61.4 dBm

For long echo cancellations, echo cancellers can become unstable if their update rate and, hence, convergence is too fast. The 32-bit cancellers converge to a residual of –76...–78 dB. For comparison, the cancellation of the 32- and 64-ms cancellers is shown in Table 9 and Table 10.

Table 9. 32-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.4 dBm	–70.5 dB	–76.1 dBm	–66.6 dB	–78.3 dBm	–61.3 dB	–78.9 dBm	–52.4 dB	–76.1 dBm
–5.6 dBm	–66.9 dB	–78.5 dBm	–60.6 dB	–78.3 dBm	–54.5 dB	–78.2 dBm	–47.8 dB	–77.5 dBm
–11.7 dBm	–58.4 dB	–76.1 dBm	–52.3 dB	–76.0 dBm	–48.5 dB	–78.2 dBm	–42.5 dB	–78.2 dBm
–17.7 dBm	–54.5 dB	–78.2 dBm	–51.5 dB	–81.2 dBm	–40.9 dB	–76.7 dBm	–39.0 dB	–80.8 dBm
–23.7 dBm	–50.6 dB	–80.3 dBm	–40.2 dB	–75.9 dBm	–37.4 dB	–79.2 dBm	–29.9 dB	–77.8 dBm
–29.7 dBm	–41.5 dB	–77.2 dBm	–38.4 dB	–80.2 dBm	–28.8 dB	–76.6 dBm	–24.2 dB	–78.1 dBm

Table 10. 64-ms Echo Tail Using 32-Bit Filter Weights (1/2 Second)

	–6-dB ECHO		–12-dB ECHO		–18-dB ECHO		–24-dB ECHO	
LINE OUT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT	CANCELLA-TION	RESULT
–0.4 dBm	–33.0 dB	–38.6 dBm	–33.0 dB	–44.7 dBm	–33.0 dB	–50.7 dBm	–33.1 dB	–56.8 dBm
–5.6 dBm	–33.0 dB	–44.7 dBm	–33.0 dB	–50.7 dBm	–33.1 dB	–56.8 dBm	–32.5 dB	–62.3 dBm
–11.7 dBm	–32.3 dB	–50.0 dBm	–32.4 dB	–56.1 dBm	–32.3 dB	–62.0 dBm	–31.2 dB	–67.0 dBm
–17.7 dBm	–32.0 dB	–55.7 dBm	–32.0 dB	–61.8 dBm	–31.9 dB	–67.6 dBm	–30.6 dB	–72.4 dBm

5 Description

The TI echo control code for the TMS320C62xx processor (’C62xx) meets either the ITU G.165 or DECT echo control standards, and is designed for implementing multichannel echo controllers.

5.1 RAM Requirements

The memory requirements for the echo control software depend on the number of channels, the echo tail length, and the accuracy of the weighting filters. Table 11 shows how to calculate memory requirements. The circular buffer must be an integral power of 2, due to the way circular buffers are implemented on the ’C62xx processor.

Table 11. Calculating Memory Requirements

USE	BYTES (16-BIT WEIGHTS)	BYTES (32-BIT WEIGHTS)
Global variables	8	8
Channel variables n	40n	40n
Circular buffers 2^z	$2n* 2^z$	$2n* 2^z$
Weighting filter w	2n*w+4	2n*w+4
Modem answer tone detect	28n	28n

Table 12 shows some worked examples for common echo canceller configurations without modem answer tone detection.

Table 12. Worked Example Memory Requirements

Canceller			PARAMETERS			MEMORY	
TIME	CHANNELS	WEIGHTS	n	z	w	BYTES EQUATION	BYTES TOTAL
4 ms	1	16	1	5	30	$8+40*1+2*2^5+2*1*30+4$	176
8 ms	1	16	1	6	62	$8+40*1+2*2^6+2*1*62+4$	304
16 ms	1	32	1	7	126	$8+40*1+2*2^7+4*1*126+4$	816
32 ms	1	32	1	8	254	$8+40*1*2*2^8+4*1*254+4$	1584
4 ms	32	16	32	5	30	$8+40*32+2*2^5+2*32*30+4$	5632
8 ms	32	16	32	6	62	$8+40*32+2*2^6+2*32*62+4$	9728
16 ms	32	32	32	7	126	$8+40*32+2+2^7+4*32*126+4$	25864
32 ms	32	32	32	8	254	$8+40*32+2*2^8+4*32*254+4$	50440

5.2 ROM Requirements

The program size for 16-bit and 32-bit weights is different; however, the program size does not vary with the number of channels implemented. Table 13 specifies program memory requirements.

Table 13. Program Memory Requirements

USE	BYTES (16-BIT WEIGHTS)	BYTES (32-BIT WEIGHTS)
Echo control program	2568	2152
Modem answer detect program	464	464
Example C	2960	2960
Total	5992	5576

5.3 MIPS Requirements

The millions of instructions per second (MIPS) required for echo control depends on the length of the tail being cancelled and on whether or not the canceller is being updated. Typically, a canceller needs to be updated only during the initial stages of a call; once converged, the canceller can be frozen and the update stage avoided. This process can be used to increase the number of channels being cancelled. The 16-bit echo canceller incorporates a filter inversion process to eliminate rounding errors in the update process. This means that even when not being updated, this filter inversion process must be implemented. Thus, an alternative update routine, No_Update, that requires less cycles should be called instead. With the 32-bit canceller, the rounding error is at a much lower level and can be safely ignored.

This code was developed using a pre-release version 1.1 of the simulator, from which the benchmarks in Table 14 were obtained.

Table 14. CPU Cycle Requirements

SUBROUTINE	FORMULA
Echo cancel 16-bit weights	30+n/2
Echo cancel 32-bit weights	34+n
Echo update 16-bit weights	28+n
Echo no update 16-bit weights	15+n/2
Echo update 16-bit weights	28+n
Echo update 32-bit weights	20+3*n/2
Echo suppress	35/38
Echo nonlinear process	14
Modem answer tone detect	26

Table 15 shows some worked examples for common echo canceller configurations without modem answer tone detection.

Table 15. CPU Cycles/MIPS Requirements

USE	CHANNELS	DURING UPDATE			AFTER UPDATE		
		FORMULA	CYCLES	MHz	FORMULA	CYCLES	MHz
16-bit weights 4 ms	1	96+3*n/2	144	1.2	83+n	115	.9
16-bit weights 8 ms	1	96+3*n/2	192	1.5	83+n	147	1.2
32-bit weights 16 ms	1	92+5*n/2	412	3.3	72+n	200	1.6
32-bit weights 32 ms	1	92+5*n/2	732	5.9	72+n	328	2.6
32-bit weights 64 ms	1	92+5*n/2	1372	11.0	72+n	584	4.7
16-bit weights 4 ms	32	96+3*n/2	4608	37	83+n	3680	29
16-bit weights 8 ms	32	96+3*n/2	6144	49	83+n	4704	38
32-bit weights 16 ms	32	92+5*n/2	13184	105	72+n	6400	51
32-bit weights 32 ms	32	92+5*n/2	23424	187	72+n	10496	84
32-bit weights 64 ms	32	92+5*n/2	43904	351	72+n	18688	150

6 Software Routines

While all of the ITU G.165 and DECT echo control code is written in ’C62xx assembler language, the routines are written within the guidelines for ’C6x C-callable routines, and can be used with the TI C compiler. For use with other compilers or assembler code, it may be necessary to check which registers are parent protected and which are child protected. The TI C compiler uses parent protection for registers a0...9 and b0...9 with child protection for registers a10...15 and b10...15. This means that registers a0...9 and b0...9 are not protected within the assembler routines. In parent protection, registers are saved by the calling routine before the subroutine call. In child protection, registers are saved by the child routine after the subroutine call.

6.1 C Code and C-Callable Subroutines

All of the main ITU G.165 and DECT echo control subroutines are C-callable. When necessary, these routines disable interrupts to protect multiple assignment code. The maximum time for which interrupts are disabled depends on the echo delay and version chosen. The test vector example program and the C functions are explained in the following sections.

6.1.1 TestEchoC6x.c

This file provides an example of how to call the various echo control assembler routines from a C environment. This program was used to generate the echo control test results. All benchmark figures were obtained by compiling this program with the -k -as -g -o2 -pe options with timing measured over the subroutine call. Lines of the form: label =*(* short *)address are defined addresses in the simulator where test vector or input control can be found. The use of local variables gives significant improvement in the MIPS achieved because they allow the assembler to optimize them. The four typedef variables provide C-context to the data structures used within the assembly modules. The #define of ToneChannels, specifies how many channels of modem answer tone detection channels are implemented. Due to the indeterminate size of the echo control channels (different echo tail lengths), these channels cannot be assigned by C (without significantly increasing the size of the code) and should be assigned in the assembler modules instead.

6.1.1.1 GlobalEcho

Table 16 defines variables that affect the operation of all channels.

Table 16. GlobalEcho Variables

VARIABLE	DESCRIPTION
Back_Samp_Rate	The rate at which the DECT echo suppressor updates its estimate of the background noise. Defaults every 180 samples. Not used with a nonlinear processor.
EC_EchoGain	Defines the maximum echo that can be cancelled as a fraction n/256. Default = 128 (–6 dB)
ES_Marg	The echo suppressor defines how much of the signal must be over the background to be treated as voice rather than noise as a fraction n/16384. Default 24576 (3.52 dB); for the nonlinear processor this is the absolute signal level for muting, and should be changed to 500 (mute at LSB only left).
EC_DeadTime	Defines how many digital delays are left uncanceled before the canceller. This is used to reduce MIPS requirements where there are known digital echoless delays between the canceller and the start of the echo. Default = 0.

6.1.1.2 ChannelEcho

Table 17 defines variables that are local to each channel of the echo controller.

Table 17. ChannelEcho Variables

VARIABLE	DESCRIPTION
ES_BackCount	Counter for updating background noise
ES_BackThres	Current estimate of background noise * ES_marg
ES_FIRold1/2	Current signal level for echo suppressor filter
ES_k1/2	Internal IIR variable for current echo suppression
ES_attack	Rate at which suppressor switches on
ES_decay	Rate at which suppressor switches off
ES_threshold	Level to which echo suppressor suppresses
ES_Back	Current estimate of background noise
ES_Hangover	Time after end of speech that suppressor continues to suppress in samples
Spare	Unused (data alignment)
Avg_Line_Out	Time average of Z signal, reference input (*512)
Avg_Echo_In	Time average of X signal, input to canceller (*512)
Avg_Echo_Out	Time average of Y signal, output from canceller (*512)
LineOut	Circular buffer pointer for reference signal
W[]	Unbound array of data weights for predictor (requires -pe option in C compiler)

6.1.1.3 ChannelModem

Table 18 defines variables that are local to each channel of the modem answer tone detection. These variables are order optimized for zero data page faults.

Table 18. ChannelModem Variables

VARIABLE	DESCRIPTION
pass_w1	Band-pass filter delayed products, one sample delay
stop_w1	Band-stop filter delayed products, one sample delay
pass_w2	Band-pass filter delayed products, two sample delay
stop_w2	Band-pass filter delayed products, two sample delay
pass_energy	Energy in band-pass filter
stop_energy	Energy in band-stop filter
time_out	Time a valid tone has been received for (zero no tone, max. 16383)
end_tone	Flag to indicate a possible phase reversal

6.1.1.4 ToneGen

Table 19 defines variables that are local to each channel of the tone generation routine (used for test purposes only, not part of main echo control code). These variables are order optimized for zero data page faults.

Table 19. ToneGen Variables

VARIABLE	DESCRIPTION
f1_a1	Frequency 1, resonator constant for frequency generation initialized to 32768*cos(360*Frequency/8000)
f1_sr2	Frequency 1, output sample delayed 2 samples, initialized to r*sin(360*Frequency/8000), where r is output peak amplitude
f2_a1	Frequency 2, resonator constant for frequency generation initialized to 32768*cos(360*Frequency/8000)
f2_sr2	Frequency 2, output sample delayed 2 samples, initialized to r*sin(360*Frequency/8000), where r is output peak amplitude
f1_sr1	Frequency 1, output sample delayed 1 samples, initialized to 0
f2_sr1	Frequency 2, output sample delayed 1 samples, initialized to 0

6.1.2 C External Functions

With the exception of Init_Echo(), all of these external C-callable assembler functions have the same structure, requiring the same basic parameters:

- The input signal with the echo
- The reference signal
- A pointer to the channel-specific variables
- A pointer to the global variables
- They all return the improved echo signal.

The parameter format is (short Sin, short Rin, ChannelEcho *channel, GlobalEcho *global, short *out1, short *out2).

6.1.2.1 Init Echo()

This function requires no parameters, but returns the number of channels that the assembly code was assembled for (see Section 6.2, *Echo Control Assembler Code*). It resets all the channels defined at assembler time in the echo control routine. It is defined in echo_c6xxx.asm.

6.1.2.2 Echo_Cancel (short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global)

This function performs the time-critical part of the echo cancellation process. It is defined in echo_c6xxx.asm. Interrupts are disabled during this routine.

Sin	Signal to cancel
Rin	Reference signal for canceller
*Channel	Pointer to local channel data
*Global	Pointer to global channel data
Returns	Canceled signal

6.1.2.3 Echo_Update (short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global)

This function performs the non-time-critical part of the echo cancellation process. It still should be called once per channel per loop, but it can be called later in the loop because its output does not affect the overall output. This may reduce the group delay. It is defined in echo_c6xxx.asm. Interrupts are disabled during this routine.

Sin	Canceled signal
Rin	Reference signal for canceller
*Channel	Pointer to local channel data
*Global	Pointer to global channel data
Returns	Canceled signal

6.1.2.4 No_Update (ChannelEcho *Channel)

This function replaces Echo_Update for the 16-bit code only. When Update is not required for the 32-bit code, no action is required due to negation of the filter to remove dc offsets, which are insignificant in the 32-bit code. It is defined in echo_c6xxx.asm16. Interrupts are disabled during this routine.

*Channel	Pointer to local channel data

6.1.2.5 Echo_Suppress (short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global)

This function performs the echo suppression process for DECT. It is defined in echo_c6xxx.asm. Interrupts are disabled during this routine. It returns the suppressed value.

Sin	Canceled signal
Rin	Reference signal for canceller
*Channel	Pointer to local channel data
*Global	Pointer to global channel data
Returns	Canceled and suppressed signal

6.1.2.6 Echo_NLP (short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global)

This function performs the echo nonlinear process for ITU G165. It is defined in echo_c6xxx.asm. Interrupts are enabled during this routine. It returns the nonlinear processed value.

Sin	Canceled signal
Rin	Reference signal for canceller
*Channel	Pointer to local channel data
*Global	Pointer to global channel data
Returns	Canceled and nonlinear processed signal

6.1.2.7 ModToneReset (ChannelModem *Channel, short Channels)

This function resets one or more modem answer tone detection channels. The address of the channel or first address of an array of channels is passed along with the number of channels, and if multiple channels are to be reset, they must be memory contiguous. It does not return a value. It is defined in ModemTone.asm. Interrupts are enabled during this routine.

*Channel	Pointer to local channel
Channels	Number of channels to reset

6.1.2.8 ModTone (short sin, ChannelModem *Channel)

This function performs the modem answer tone detection. It is defined in ModemTone.asm. Interrupts are enabled during this routine. It returns the status of the tone.

Sin	Signal to detect modem tone on	
*Channel	pointer to local channel data	
Returns	0	No tone present
	2	Tone present without phase reversals (or no phase reversals yet)
	3	Tone present with phase reversals

6.1.2.9 InitTone (short f1_a1,short f1_sr2,short f2_a1,short f2_sr2,ToneGen *Tone)

This function is not part of the echo control code, but it generates test data tones for the modem answer tone detection. It initializes the data structure.

6.1.2.10 InitDTMF (short Digit, oneGen *Tone)

This function is not part of the echo control code, but it programs the tone generator to generate dual-tone multifrequency (DTMF) tones.

6.1.2.11 ToneGenerate (ToneGen *Tone)

This function is not part of the echo control code, but it generates test data for the modem answer tone detection. It generates a sample from the data structure.

6.2 Echo Control Assembler Code

All of the echo control core functions are written in highly optimized assembly code to give very efficient performance in terms of MIPS. Several of the functions contain multiple assignment code and disable interrupts; these routines also enable interrupts at the end. Some routines temporarily change the AMR register to make register a5 a circular address pointer. The functions contained in each file are described in this section.

6.2.1 Echoc6xxx.asm

This module reserves the data RAM for all the channels defined by the .equ statement near the top of the file by the variable *channels*, and the length of the echo tail by the .equ statement for the variable *order*. Valid values for *order* are 5, 6, 7, and 8, generating 4-, 8-, 16-, and 32-ms tail echo cancellers, respectively. In principle, the *order* could be increased to values higher than 7, though the value in the update routine would need to be reduced, resulting in a corresponding increase in convergence time to maintain stability within the algorithm.

This module defines the C-callable functions Init_Echo, Echo_Cancel, Echo_Update, Echo_Supress, and Echo_NLP and the C-referenceable variables ChannelTable and EchoGlobal. These variables can be used by C to alter the behavior of the algorithm and the G726_reset, which resets one or all of the channels.

6.2.2 Echoc6xxx.asm16

This module must be included via a .copy or .include directive, from a master assembler program in which the variables *order*, z*order*, *worder*, and *AMRVal*, have been predefined. It contains the code that defines the data structures, reset code, and echo cancellation functions that perform the 16-bit weighting factor versions of the echo canceller. The pipelines for the cancellation and update are shown in Table 20 through Table 22. All of the main loops are coded to have zero memory stalls.

Table 20. 16-Bit Cancellation Z0 Word-Aligned Pipeline

1	2	3(1)	4(2)	5(1)	6(2)	7(1)	8(2)	9(1)	10(2)
LDW Z0,Z1						MPYLH Z0,W0		ADD sum0,2	
	LDW W0,W1					MPYHL Z1,W1		ADD sum1,3	
	LDW W2,W3						MPYLH Z2,W2		ADD sum0,2
		LDW Z2,Z3					MPYHL Z3,W3		ADD sum1,3
								B Loop	Dec Loop

Table 21. 16-Bit Cancellation Z0 Not Word-Aligned Pipeline

1	2	3(1)	4(2)	5(1)	6(2)	7(1)	8(2)	9(1)	10(2)	11(1)
LDW Z–1,Z0		LDW Z3,Z4				MPY Z0,W0		ADD sum0,2		
	LDW W0,W1						MPYH Z1,W1		ADD sum1,3	
	LDW W2,W3					MV W2,W3	MPY Z2,W2		ADD sum0,2	
		LDW Z1,Z2						MPYH Z3,W3		ADD sum1,3
								B Loop	Dec Loop	

NOTE: LDW Z–1,Z0 and LDW Z3,Z4 are really the same instruction.

Table 22. 16-Bit Update Pipeline

1	2	3(1)	4(2)	5(1)	6(2)	7(1)	8(2)	9(1)	10(2)	11(1)	12(2)	13(1)	14(2)	15(1)
	LDH *5++,9					MPY LH 9,2.7		SUB 7,8,7	SHR 7,8,0		OR a0,b0,b1	MV b1,a3		STW a3,*a6++
LDW *b6++,b4					MPY SU b4,2,8									
					MPY HSLU			SUB	CLR					
	LDH					MPY LH								
							B Loop	Dec Loop						

NOTE: A6 saves in old B6 and is 28 RAM locations behind to avoid memory stalls.

6.2.3 Echoc6xxx.asm32

This module must be included via a .copy or .include directive from a master assembler program in which the variables *order*, *zorder*, *worder*, and *AMRVal* are predefined. It contains the code that defines the data structures, reset code, and echo cancellation functions that perform the 32-bit weighting factor versions of the echo canceller. The pipelines for the cancellation and update are shown in Table 23 and Table 24. All of the main loops are coded to have zero memory stalls.

Table 23. 32-Bit Cancellation Pipeline

1	2	3(1)	4(2)	5(1)	6(2)	7(1)	8(2)	9(1)	10(2)	11(1)
		KDH *b5,a11					MPYSU a11,b9,a8		SHR a8,16,a3	ADD a3,a7,a7
	LDW *b10,b9						MPYLH a11,b9,b8		ADD b8,b7,b7	
LDH *a5,b11						MPYSU b11,a9,a8		SHR a8,16,a3	ADD a3,a7,a7	
	LDW *a10,a9					MPYLH b11,a9,b8		ADD b8,b7,b7		
						B Loop	Dec Loop			

Table 24. 32-Bit Update Pipeline

1	2	3	4(1)	5(2)	6(3)	7(1)	8(2)	9(3)	10(2)	11(2)	12(3)
						SHR 7,2,8					
	LDW *6,7						SUB 7,8,8		ADD 8,9,9		STW 9,*6
						MPYHL 3,A1,0		ADD 0,7,9			
LDH *5,3					MPYUS 3,A1,3		SHR 3,16,7				
LDH					MPYUS		SHR				
						MPYHL		ADD			
	LDW						SUB		ADD		STW
						SHR					
								Dec Loop			
								B Loop			

6.2.4 ModemTone.asm

This module defines the C-callable functions ModToneReset and ModTone. The data memory for these routines should be reserved by the C source code because it is only referenced indirectly. There are no user-defined .equ statements in these routines.

6.2.5 ToneGenerate.asm

This module defines the C-callable functions InitTone, InitDTMF, and ToneGenerate. These functions are used to generate test data only and are not part of the echo control code, although they could have other uses. The data memory for these routines should be reserved by the C source code because it is only referenced indirectly. There are no user-defined .equ statements in these routines.

7 References

1. ITU Recommendation G.164

   General characteristics of international telephone connections and international telephone circuits. Echo cancellers
2. ITU Recommendation G.165

   General characteristics of international telephone connections and international telephone circuits. Echo cancellers
3. ITU Recommendation G.711

   Pulse code modulation (PCM) of voice frequencies
4. ITU Recommendation G.712

   Transmission performance characteristics of pulse code modulation
5. ITU Recommendation V.25

   Data communication over the telephone network. Automatic answering equipment and/or parallel automatic calling equipment on the general switched telephone network, including procedures for disabling echo control devices for both manually and automatically established calls.
6. European Telecommunications Standards Institute (ETSI) 300 175-*
7. Digital Enhanced Cordless Telecommunications (DECT)
8. *TMS320C62xx CPU and Instruction Set,* literature number SPRU189D
9. *TMS320C62xx Programmer's Guide,* literature number SPRU198C