Implementation of Echo Control for G165 / DECT on Texas Instruments TMS320C62xx processors

Application Report

Last updated 5-May-98

TEXAS INSTRUMENTS

Overview

The 4-Wire to 2-Wire Hybrid used in analogue telephone transmission introduces an echo into the received part, when this echo is less than a few milliseconds it provides the telephone user with “comfort” that is telephone is indeed working, however as the echo extends beyond a few milliseconds it becomes very distracting to the user and causes them to slow down their speech. With the installation of more digital echo-less equipment into the telephone network the distance between the user and the hybrid is increasing; this leads to a consequential increase to the echo delay. It is therefore becoming more necessary to both cancel the long delay echo and also to provide a short delay “comfort” echo so that the users hear the echo they are used to, or in the case of multiple echo cancellers, there is an echo for echo cancellers nearer the users to cancel.

This application note describes an efficient implementation of multi-channel echo-controllers using the TMS320C62xx processor core, that can cancel echoes of 4-32mS delay. The software is based on the leaky normalized LMS (Least Mean Square) filter with either 16 or 32 bit weighting coefficients.

The software could easily be adapted to handle longer echo’s by changing some of the update parameters.

All code is c-callable optimized assembler code, and includes subroutines for both the G.165 and DECT styles of echo control.

Contents

1. G.165 and DECT Echo Control 5
2. A brief Introduction to Telecom Levels in Echo Cancellers 7
2.1 The Echo Cancellation Algorithm 7
2.1.1 Echo Prediction 7
2.1.2 Predictor Update 8
2.1.3 Echo Suppresser 10
2.1.4 Non Linear Processor 11
2.1.5 Modem Answer Tone Detection 11
2.2 Performance Limitations for Echo Cancellation 12
2.2.1 Linear Signal Noise 12
2.2.2 Codec Companding Distortion 13
2.2.3 Convergence Noise 13
2.2.4 Leakage Noise 13
3. Echo Canceller Performance 14
4. Description 16
4.1 RAM requirements 16
4.2 ROM requirements 16
4.3 MIPs requirements 17
5. Software Routines 18
5.1 C-code and C-callable 18
5.1.1 TestEchoC6x.c 18
5.1.1.1 GlobalEcho 18
5.1.1.2 ChannelEcho 19
5.1.1.3 ChannelModem 19
5.1.1.4 ToneGen 19
5.1.2 c Externals 20
5.1.2.1 Init_Echo() 20
5.1.2.2 Echo_Cancel(short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global) 20
5.1.2.3 Echo_Update(short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global) 20
5.1.2.4 No_Update(ChannelEcho *Channel) 21
5.1.2.5 Echo_Suppress(short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global)21
5.1.2.6 Echo_NLP(short sin, short rin, ChannelEcho *Channel, GlobalEcho *Global) 21
5.1.2.7 ModToneReset(ChannelModem *Channel, short Channels) 21
5.1.2.8 ModTone(short sin, ChannelModem *Channel) 22
5.1.2.9 InitTone(short f1_a1,short f1_sr2,short f2_a1,short f2_sr2,ToneGen *Tone) 22
5.1.2.10 InitDTMF(short Digit, oneGen *Tone) 22
5.1.2.11 ToneGenerate(ToneGen *Tone) 22
5.2 Echo Control assembler code 22
5.2.1 Echoc6xxx.asm 22
5.2.2 Echoc6xxx.asm16 22
5.2.3 Echoc6xxx.asm32 24
5.2.4 ModemTone.asm 24
5.2.5 ToneGenerate.asm 25
6. Acronyms 26
7. References 26

List of Tables

Table 1 Digital Codec Levels 7
Table 2 Normalized Digital Codec Levels 7
Table 3 Echo Canceller leak rates 10
Table 4 4mS Echo Tail using 16 bit Filter Weights (½ Second) 14
Table 5 8mS Echo Tail using 16 bit Filter Weights (½ Second) 14
Table 6 16mS Echo Tail using 32 bit Filter Weights (½ Second) 14
Table 7 32mS Echo Tail using 32 bit Filter Weights (½ Second) 14
Table 8 64mS Echo Tail using 32 bit Filter Weights (½ Second) 15
Table 9 32mS Echo Tail using 32 bit Filter Weights (1 Second) 15
Table 10 64mS Echo Tail using 32 bit Filter Weights (1 Second) 15
Table 11 How to Calculate memory requirements 16
Table 12 Worked Example Memory requirements 16
Table 13 Program Memory Requirements 16
Table 14 CPU cycle requirements 17
Table 15 CPU cycles/MIPs requirements 17
Table 16 16-Bit Cancellation z0 word aligned pipeline 23
Table 17 16-Bit Cancellation z0 not word aligned pipeline 23
Table 18 16-Bit Update Pipeline 23
Table 19 32-Bit Cancellation pipeline 24
Table 20 32-Bit Update Pipeline 24

List of Figures

Figure 1 G.165 Echo Control on both ends of International Link 5
Figure 2 DECT Echo Control Software Block 6
Figure 3 Modem Answer Tone Response Curves 12

1. G.165 and DECT Echo Control

Both of these specifications 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 re-introduction of the comfort noise. In G.165 residual echo is removed by a Non-Linear Processor (NLP) which basically mutes all signals below a given threshold, in DECT residual echo is suppressed using an Echo Suppresser that suppresses far end speech by 9-12dB when the near end user is talking. In G.165 comfort noise is normally introduced in the same Echo Controller that cancels the data, whereas in DECT it takes a more system view introducing the comfort noise for its partner Echo Control at the other end of the RF link, the DECT approach gives shorter echoes for further cancellers but does require a common control strategy for both ends of the link, whereas G.165 allows both ends to have different strategies. Examples of G.165 and DECT Echo Controls are shown in figures 1 and 2 respectively. In both these diagrams the echo control blocks are shown bold, other DSP functions are shown in solid boxes, and external blocks are shown dotted, the A/D/A and hybrid are replaced by digital ISDN style links in some systems. Side-tone is sometimes also referred to as "comfort noise".

Whilst the removal of distant echo's and their replacement by a "nearer" echo's is good for the human perception of speech, the distortion introduced can up-set modem's and their in-built echo controllers, for this purpose modems and other equipment requiring no external echo control transmit a disabling tone at the start of a call, this tone is detailed in V.25 and G.164. This application code also contains routines to detect this tone, though the use of this information to control the actual routines is left to the end-user as this varies from application to application.

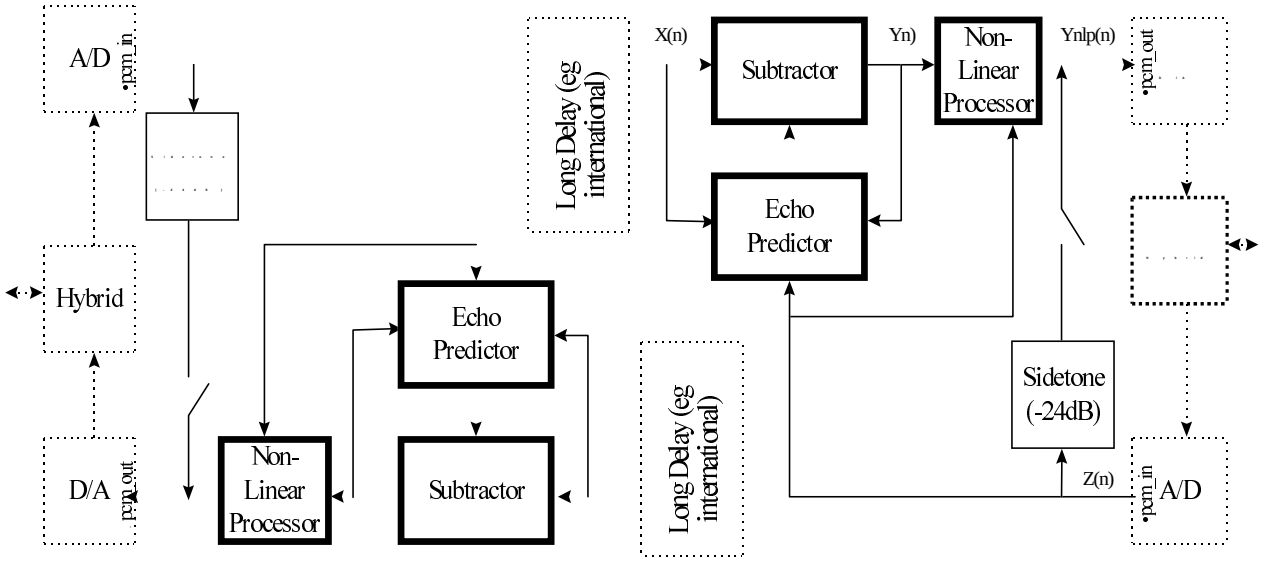

Figure 1 G.165 Echo Control on both ends of International Link

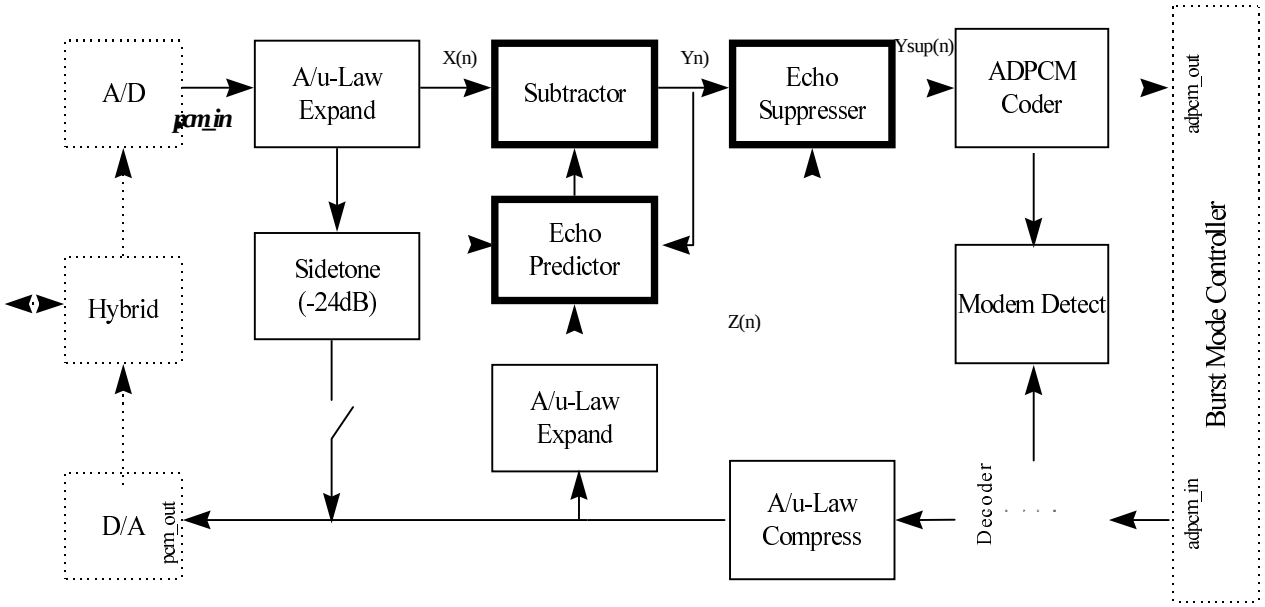

Figure 2 DECT Echo Control Software Block

2. A brief Introduction to Telecom Levels in Echo Cancellers

Voice data is represented in telecommunications by discrete digital levels these can be linear or logarithmic coded, most of the worlds communications is implemented using logarithmic coding complying with either the A-Law or μ-Law standards specified in G.711, for voice coding these values must be linearized to produce linear digital code, different algorithms linearize to different resolutions. The digital equivalents defined in G.711 for A-Law or μ-Law are shown below.

Law	Min	Max
A-Law	-4032	4032
μ-Law	-8031	8031

Table 1 Digital Codec Levels

In order to provide uniform levels into voice coders like G.726 A-law levels are multiplied by 2 to produce 14 bit 2's complement or Q13 numbers.

The 0dBm point is defined as the r.m.s. level of a sin wave 3dB below the clipping level, therefore the mean amplitude for 0dB is half the maximum level

Law	Min	Max	0dBm point
A-Law*2	-8064	8064	4032
μ-Law	-8031	8031	4016
Linear	-8192	8191	4095

Table 2 Normalized Digital Codec Levels

2.1 The Echo Cancellation Algorithm

2.1.1 Echo Prediction

The general algorithm of a Transversal FIR predictor is given in 1 below:-

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

(1) Transfer Structure of FIR Predictor

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)

$$Y(n) = X(n) - Yp(n)$$

(2)Calculation of Error Signal from input and predicted signal

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 cancelled from the return path as shown in 2 to produce the predicted error or echo cancelled signal. When there is no signal from the near end this signal will be the error in the echo predictor however when there is a signal from the near end this signal will represent the near end signal with the echo from the far end cancelled out. In order to properly update the predictor it is necessary to know which of these is the case, the algorithm therefore contains a number of conditions which have to be met before the predictor is allowed to be updated.

1) The signal being sent from the far end must be above a certain threshold. i.e. There must be speech at the far end for their to be an echo capable of being cancelled.

2) The signal being received from the near end must be significantly less than that being sent from the far end. i.e. 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 it is better not to update the predictor.

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

$$\mathrm{Signal}_{\mathrm{Level}}(n) = \mathrm{Signal}_{\mathrm{Level}}(n-1) + \left[\frac{|\mathrm{Signal}_{\mathrm{Input}}(n)| - \mathrm{Signal}_{\mathrm{Level}}(n-1)}{32}\right]$$

(3) Equation for Measuring Mean Input Signal Level

By comparing the results of this equation for Z(n) a suitable threshold for speech transmission can be determined and the predictor only updated when the level exceeds this threshold. Double talk is detected by comparing the signal on both these filters and when the Z(n) filter is sufficiently greater than the Y(n) filter then the predictor can safely be updated. Alternatively the scaling factor for the update can be based on the relative strengths of the 2 signals though this can give problems when the algorithm is initially reset if the echo signal is to strong.

2.1.2 Predictor Update

In order 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 will converge on to the actual echo, however if these parameters are changed rapidly a significant distortion is introduced to the signal received from the lines; indeed if too much feedback is then the filter can even go unstable. It is therefore necessary to reach a compromise between update rate and the distortion introduced to the signal. From 1 and 2 it is obvious that the error in the predicted signal can be given as in 4 below.

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

(4) Error Calculation for FIR Predictor

It is beyond the scope of this document but it has been shown that the update for this predictor can be performed using the LMS algorithm as shown in 5 below. Anyone interested should read the section on "Implementation of Adaptive Filters" in reference 1.

$$W_k(n+1) = W_k(n) + u * Y(n) * Z(n-k)$$

(5)Ideal Predictor Update Equation

Given that the delay to the echo that is being cancelled 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 giving an echo canceller that produces a low level of distortion. As the update rate of this filter is proportional to both the input signal level and the echo signal level, the filter tends to converge much faster for higher signal levels, this can be counteracted by normalizing the update rate to the level of these input signals, in fact the echo is proportional to the input signal so the normalization can be to the input signal $Z(n)^2$. Care has to be taken not to place to much emphasis on individual samples as near zero crossings the error can be comparatively 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. The normalization constant is given in 6 which when substituted in 4 gives 7. 7 Also shows the actual constant used to set the convergence stability compromise.

$$Normalisation = 2^{INT(2*-LOG_2(Z_{Level}))}$$

(6)Normalisation Equation

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

(7)Normalized Ideal Predictor Update Equation

In order to stabilize this filter under tones and prevent overflows a leak of 1/m is introduced into the update, the value of this leak is dependent on the level of cancellation achieved. The signal level before and after cancellation is measured using equation 3 above and the function shown in equation 8 below to determine their approximate amplitudes.

$$INT(-LOG_2(Signal_{level}))$$

(8) Logarithmic Measure of Signal Level

The difference between these 2 logs gives the cancellation in 6dB steps, this is translated to a leak rate according to the values shown in the table below:-

Measured Cancellation	Leak (16 bit weights)	Leak (32 bit weights)
>36Db		...
>24dB	1/32768	
30..36dB		1/65536
24..30dB		1/32768
18..24dB	1/16384	1/16384
12..18dB	1/8192	1/8192
6..12dB	1/4096	1/4096
0..6dB	1/2048	1/2048
-6..0dB	1/1024	1/1024
…	…	…

Table 3 Echo Canceller leak rates

This equation then becomes:-

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

(9)Leaky Normalized Error * Data Predictor Update Equation

In terms of the actual inputs the overall equation for the update of the predictor coefficients is therefore:-

$$W_k(n+1) = \left(\frac{W_k(n)*(m-1)}{m}\right) + \left(\frac{2^{INT(2*-\mathrm{LOG}_2(Z_{Level}(n)))} * \left(X(n) - \sum_{l=0}^{31} W_l(n)*Z(n-l)\right) * Z(n-k)}{512}\right)$$

(10)Full Predictor Parameter Update Equation

Overall this predictor will converge onto the echo in about 250μS, nearly cancelling the echo down to the floor level of the system. In order to further reduce the echo the use of either an echo suppresser (DECT) or non-linear processor (G.165) is required,

2.1.3 Echo Suppresser

The Echo Suppresser can be programmed to give various echo suppressions from 0dB to 24dB, whatever it is programmed to give the echo suppression is soft switched in and out with an attack time of 5mS and a decay time of 25mS. If full suppression is reached there is a hangover time of 70mS after the end of speech before the echo suppresser is switched out.

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

$$If \sum_{k=0}^{7} |Z(n-k)| > Supp\_Thresh$$
$$And \sum_{k=24}^{31} |Z(n-k)| > Supp\_Thresh$$
$$Then\ Suppressor_{Change}(n) = Attack\_Constant$$
$$Else\ Suppressor_{Change}(n) = Decay\_Constant$$

(11) Echo Suppresser FIR Speech Detector

The $Suppress_{Change}$ 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 70mS, which is initiated once echo suppression has reached the maximum value, there is no hangover if echo suppression does not get fully turned on. The equation for the IIR is shown in 12 below. The time constant for this filter is altered between the attack and the decay by varying the constant k1.

$$Suppress(n) = Suppressor_{Change}(n) + k1 * Suppress(n-1) + k2 * Surpress(n-2)$$

(12) Echo Suppresser IIR Suppression Filter

After filtering Suppress will have a range from 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 equation for this suppression function is shown in 13 below.

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

(13) Application of Echo Suppression to Signal

2.1.4 Non Linear Processor

Alternatively to echo suppression the echo cancelled signal can be further reduced by a non-linear processor, in this case the output is simply muted when the cancelled signal falls below a given threshold.

$$if \quad Z_{level}(n) < Threshold$$
$$then \quad Y_{NLP}(n) = 0$$
$$else \quad Y_{NLP}(n) = Y(n)$$

(14) Non Linear Processor

2.1.5 Modem Answer Tone Detection

In addition to cancellation it is necessary to detect the 2100Hz Modem answer tone in applications where modem data might be used instead of voice, this is more common in G.165 type echo controllers than DECT echo controllers, though where DECT s used in Radio Local Loop applications this is also needed for DECT. Answer tone detection is performed using two basic 2nd order IIR (Infinite Impulse Response) filters used to measure the tone energy and the out 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 as the predicted signal is an exact opposite of the real signal, causing cancellation). The Basic IIR equation is:-

$$y_n = x_n * b_0 + x_{n-1} * b_1 + x_{n-2} * b_2 + y_{n-1} * a_1 + y_{n-2} * a_2$$

(15) Infinite Impulse Response Filter

The performance graphs of the band-pass and band-stop filters are:-

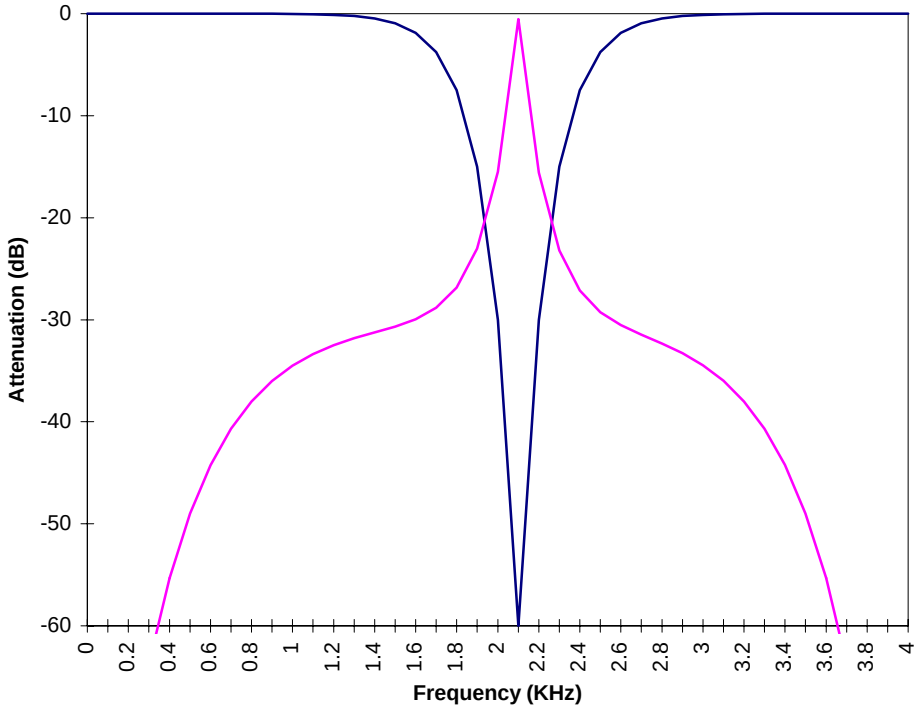

Figure 3 Modem Answer Tone Response Curves

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

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

(16) Equation for Measuring Mean Tone Signal Level

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 the tone being present for sufficient samples to be a valid tone.

2.2 Performance Limitations for Echo Cancellation

The performance of the Echo Canceller is limited by one of four factors:-

2.2.1 Linear Signal Noise

This noise comes from the ability to represent the output signal within the echo canceller, this is a resolution limit and as such is ½ the lsb * the 0dB level or 20*log(4095*2) = 78.2dBm. Achieving this with linear data gives a good indication to the integrity of a simulated echo canceller though for real systems one of the remaining 3 factors will usually dominate.

2.2.2 Codec Companding Distortion

This noise is specified for an ideal codec at 33dB in G.711, for a real codec G.712 specifies 31dB. This will limit a cancellers ability to cancel to 31-33dB depending on the quality of the codec used. This is fundamental to all telephone systems.

2.2.3 Convergence Noise

This noise is the noise introduced into the output signal by the changes to the weighting parameters, this in turn 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 will be limited by the minimum error needed to change the weights by one, and will be this error * Length of the predictor * affect on the predictor. With 32 bit weights the minimum error will be that of the input signal resolution, and so both the error and the affect are reduced, giving a significant performance improvement

Generally if the echo tail is more than 8mS then this noise will exceed the Codec Companding distortion. This is the limiting factor in choosing to use single or double precision weighting factors.

2.2.4 Leakage Noise

Certain tones will cause an echo canceller to go unstable, this is counteracted by adding in 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 given by the Length * Leakage for an 8mS canceller the Length is 64 samples and the minimum leakage that can easily be obtained is 1/32768, the leakage degradation is therefore 54dB, when canceling artificial hybrids where most of the delay values are zero this will dominate instead of convergence noise.

3. Echo Canceller Performance

The echo canceller algorithm has been simulated using the TMS320C62xx simulator using tests similar to those described in G.165 for a variety of echo delays, signal levels and echo types. These experimental results are summarized in the tables. In all case's convergence is specified as that obtained after 4000 samples or 500mS of data below and has a standard deviation of ±1.5dB.

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

Table 4 4mS Echo Tail using 16 bit Filter Weights (½ Second)

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

Table 5 8mS Echo Tail using 16 bit Filter Weights (½ Second)

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

Table 6 16mS Echo Tail using 32 bit Filter Weights (½ Second)

	-6dB Echo		-12dB Echo		-18dB Echo		-24dB Echo	
LineOut	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant
-0.8dBm	-46.8dB	-54.3dBm	-46.7dB	-60.2dBm	-46.2dB	-65.8dBm	-44.5dB	-70.0dBm
-6.8dBm	-46.8dB	-60.3dBm	-46.8dB	-66.4dBm	-44.2dB	-69.8dBm	-41.2dB	-72.8dBm
-12.9dBm	-45.5dB	-65.0dBm	-44.1dB	-69.7dBm	-41.4dB	-72.9dBm	-38.1dB	-75.7dBm
-18.9dBm	-42.8dB	-68.4dBm	-40.7dB	-72.3dBm	-37.1dB	-74.7dBm	-29.8dB	-73.3dBm
-24.9dBm	-40.9dB	-72.5dBm	-36.7dB	-74.3dBm	-31.7dB	-75.2dBm	-25.8dB	-75.3dBm
-30.9dBm	-26.9dB	-64.5dBm	-27.1dB	-70.7dBm	-24.0dB	-73.6dBm	-20.6dB	-76.2dBm

Table 7 32mS Echo Tail using 32 bit Filter Weights (½ Second)

	-6dB Echo		-12dB Echo		-18dB Echo		-24dB Echo	
LineOut	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant
-0.8dBm	-18.4dB	-25.9dBm	-18.4dB	-31.9dBm	-18.4dB	-37.9dBm	-18.4dB	-43.9dBm
-6.8dBm	-18.4dB	-31.9dBm	-18.4dB	-37.9dBm	-18.4dB	-43.9dBm	-18.4dB	-49.9dBm
-12.9dBm	-18.3dB	-37.8dBm	-18.2dB	-43.7dBm	-18.2dB	-49.8dBm	-18.0dB	-55.6dBm
-18.9dBm	-18.3dB	-43.8dBm	-18.2dB	-49.8dBm	-18.2dB	-55.8dBm	-17.8dB	-61.4dBm

Table 8 64mS Echo Tail using 32 bit Filter Weights (½ Second)

For long echo cancellations, echo cancellers can become unstable if their update rate and hence convergence is to quick the 32 bit cancellers will converge to a residual of -76..-78dB. For comparison the cancellation of the 32 and 64mS cancellers is shown below after 8000 samples or 1 second.

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

Table 9 32mS Echo Tail using 32 bit Filter Weights (1 Second)

	-6dB Echo		-12dB Echo		-18dB Echo		-24dB Echo	
LineOut	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant	Cancellation	Resultant
0.4dBm	-33.0dB	-38.6dBm	-33.0dB	-44.7dBm	-33.0dB	-50.7dBm	-33.1dB	-56.8dBm
-5.6dBm	-33.0dB	-44.7dBm	-33.0dB	-50.7dBm	-33.1dB	-56.8dBm	-32.5dB	-62.3dBm
-11.7dBm	-32.3dB	-50.0dBm	-32.4dB	-56.1dBm	-32.3dB	-62.0dBm	-31.2dB	-67.0dBm
-17.7dBm	-32.0dB	-55.7dBm	-32.0dB	-61.8dBm	-31.9dB	-67.6dBm	-30.6dB	-72.4dBm

Table 10 64mS Echo Tail using 32 bit Filter Weights (1 Second)

4. Description

The Texas Instruments Echo Control code for the TMS320C62xx is designed to meet either the G.165 or DECT echo control standards. The code is designed for implementing multi-channel echo-controllers.

4.1 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, the table below shows how to calculate memory requirements, note that the circular buffer must be an integral power of 2, this is due to the way circular buffers are implemented on the c62xx.

Use	Bytes (16bit 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	4n*w+4
Modem Answer Tone Detect	28n	28n

Table 11 How to Calculate memory requirements

Below are some worked examples for common echo canceller configurations without modem answer tone detection.

Canceller			Parameters			Memory	
Time	Channels	Weights	n	z	w	Bytes Equation	Bytes Total
4mS	1	16	1	5	30	$8+40*1+2*2^5+2*1*30+4$	176
8mS	1	16	1	6	62	$8+40*1+2*2^6+2*1*62+4$	304
16mS	1	32	1	7	126	$8+40*1+2*2^7+4*1*126+4$	816
32mS	1	32	1	8	254	$8+40*1+2*2^8+4*1*254+4$	1584
4mS	32	16	32	5	30	$8+40*32+2*2^5+2*32*30+4$	5632
8mS	32	16	32	6	62	$8+40*32+2*2^6+2*32*62+4$	9728
16mS	32	32	32	7	126	$8+40*32+2*2^7+4*32*126+4$	25864
32mS	32	32	32	8	254	$8+40*32+2*2^8+4*32*254+4$	50440

Table 12 Worked Example Memory requirements

4.2 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.

Use	Bytes (16bit weights)	Bytes (32 bit weights)
Echo Control Program	2568	2152
Modem Answer Detect Program	464	464
Example C	2960	2960
Total	5992	5576

Table 13 Program Memory Requirements

4.3 MIPs requirements

The MIPS required for echo control depend on the length of the tail being cancelled and on whether the canceller is being updated or not, typically a canceller only needs to be updated during the initial stages of a call, once converged the canceller can be frozen and the update stage avoided, this 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 needs to be implemented, an alternative update routine "No_Update" that requires less cycles needs to 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 following benchmarks were obtained.

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 Non Linear Process	14
Modem Answer Tone Detect	26

Table 14 CPU cycle requirements

Below are some worked examples for common echo canceller configurations without modem answer tone detection.

Use	Channels	During Update			After Update		
		Formula	Cycles	MHz	Formula	Cycles	MHz
16 Bit Weights 4mS	1	96+3*n/2	144	1.2	83+n	115	.9
16 Bit Weights 8mS	1	96+3*n/2	192	1.5	83+n	147	1.2
32 Bit Weights 16mS	1	92+5*n/2	412	3.3	72+n	200	1.6
32 Bit Weights 32mS	1	92+5*n/2	732	5.9	72+n	328	2.6
32 Bit Weights 64mS	1	92+5*n/2	1372	11.0	72+n	584	4.7
16 Bit Weights 4mS	32	364+3*n/2	4608	37	83+n	3680	29
16 Bit Weights 8mS	32	364+3*n/2	6144	49	83+n	4704	38
32 Bit Weights 16mS	32	92+5*n/2	13184	105	72+n	6400	51
32 Bit Weights 32mS	32	92+5*n/2	23424	187	72+n	10496	84
32 Bit Weights 64mS	32	92+5*n/2	43904	351	72+n	18688	150

Table 15 CPU cycles/MIPs requirements

5. Software Routines

Whilst all of the G.165 and DECT echo control code is written in C62xx assembler, they are all written within the guidelines for C6x C callable routines, and can be used with Texas Instruments 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. Texas Instruments C compiler uses parent protection for registers a0..9 and b0..9 with child protection for 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.

5.1 C-code and C-callable

All of the main G.165 and DECT echo control subroutines are C callable, where 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. Below the test vector example program and the C functions are explained.

5.1.1 TestEchoC6x.c

This file provides an example of how to call the various echo control assembler routines from a C environment, this program is the one that was used to generate the echo control test results. All benchmark figures above 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 address’s in the simulator where test vector or input control can be found. Note the use of local variables gives significant improvement in the MIPs achieved, as they allow the assembler to optimize them into. The four typedef’s provide c-context to the data structures used within in the assembly modules. The #define of ToneChannels, specifies how many channels of Modem answer tone detection channels that are implemented. Due to the indeterminate size of the Echo Control channels (different echo tail lengths), these cannot be assigned by C (without significantly increasing the size of the code) and need to be assigned in the assembler modules instead.

5.1.1.1 GlobalEcho

This defines variables that affect the operation of all channels

Back_Samp_Rate is the rate at which the DECT echo suppresser updates its estimate of the background noise. Default every 180 samples. Not used with non-linear processor.

EC_EchoGain defines the maximum echo that can be cancelled as a fraction n/256. Default=128 (-6dB)

ES_Marg for the echo suppresser defines the amount the signal must be over the background to be treated as voice rather than noise as a fraction n/16384. Default 24576 (3.52dB); for the non-linear 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 uncancelled before the canceller, used to reduce MIPs requirements where there are known digital echo-less delays between the canceller and the start of the echo. Deafaul=0.

5.1.1.2 ChannelEcho

This defines variables that are local to each channel of the Echo Controller.

ES_BackCount	counter for updating background noise.
ES_BackThres	current estimate of background noise * ES_marg.
ES_FIRold1/2	current signal level for Echo Suppresser filter.
ES_k1/2	internal IIR variable for current Echo Suppression.
ES_attack	rate at which suppresser switches on.
ES_decay	rate at which suppresser switches off.
ES_threshold	level to which echo suppresser suppresses
ES_Back	current estimate of background noise
ES_Hangover	Time after end of speech that suppresser 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[]	Unbounded array of data weights for predictor (requires -pe option in c compiler)

5.1.1.3 ChannelModem

This defines variables that are local to each channel of the Modem Answer Tone Detection. These variables are order optimized for zero data page faults.

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-stop 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.

5.1.1.4 ToneGen

This 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.

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.

5.1.2 c Externals

These are definitions of the external of the c-callable assembler functions. With the exception of Init_Echo() all of these functions have the same structure, requiring the same basic parameters.

1. The Input signal with the echo

2.The Reference Signal

3.A pointer to the channel specific variables

4.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).

5.1.2.1 Init_Echo()

This requires no parameters but returns the number of channels that the assembly code was assembled for, see later for more details. It resets all the channels defined at assembler time in the echo control routine. It is defined in echo_c6xxx.asm.

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

This sub-routine 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	Cancelled signal.

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

This sub-routine performs the non-time critical part of the echo cancellation process. It still needs to be called once per channel per loop but can be called later in the loop as 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 Cancelled Signal.

Rin Reference signal for canceller.

*Channel Pointer to local channel data.

*Global Pointer to global channel data.

Returns Cancelled signal.

5.1.2.4 No_Update(ChannelEcho *Channel)

This sub-routine replaces Echo_Update for the 16 bit code only when Update is not required for the 32 bit code no action is required, this is 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.

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

This sub-routine 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 Cancelled Signal.

Rin Reference signal for canceller.

*Channel Pointer to local channel data.

*Global Pointer to global channel data.

Returns Cancelled and suppressed signal.

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

This sub-routine performs the echo non-linear process for G165. It is defined in echo_c6xxx.asm. Interrupts are enabled during this routine. It returns the non-linear processed value.

Sin Cancelled Signal.

Rin Reference signal for canceller.

*Channel Pointer to local channel data.

*Global Pointer to global channel data.

Returns Cancelled and non-linear processed signal.

5.1.2.7 ModToneReset(ChannelModem *Channel, short Channels)

This sub-routine 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, 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 data.

Channels No of channels to reset.

5.1.2.8 ModTone(short sin, ChannelModem *Channel)

This sub-routine 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.

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

This sub-routine is not part of the echo control code but is used to generate test data tones for the modem answer tone detection. It initialize the data structure.

5.1.2.10 InitDTMF(short Digit, oneGen *Tone)

This sub-routine is not part of the echo control code but will program the tone generator to generate DTMF tones.

5.1.2.11 ToneGenerate(ToneGen *Tone)

This sub-routine is not part of the echo control code but is used to generate test data for the modem answer tone detection. It generates a sample from the data structure.

5.2 Echo Control assembler code

All of the Echo Control core subroutines are written in highly optimized assembly code to give very efficient performance in terms of MIPs, several of the subroutines contain multiple assignment code and disable interrupts, these routines also enable interrupts at the end. Some routines also change the AMR register to make a5 circular, again this is returned to the C default value of 0 before returning. Which functions are in which file is described below:-

5.2.1 Echoc6xxx.asm

This module reserves the data RAM for all the channels defined by the ".equ" near the top of the file by the variable "channels", and the length of the echo tail by the ".equ" for the variable "order". "order" has been tested with values of 5..9 giving 4-64mS echo canceller tails. Values above 9 have not been tested and may not meet the convergence time requirements of G.165.

It defines the C-callable sub-routines Init_Echo, Echo_Cancel, Echo_Update, Echo_Supress and Echo_NLP; and the C- reference-able variables _ChannelTable and EchoGlobal which can be used by C to alter the behaviour of the algorithm. and G726_reset, which resets all or one of these channels.

5.2.2 Echoc6xxx.asm16

This module must be included via a .copy or .include directive, from a master assembler program in which the variables "order", "zorder", "worder" and "AMRVal", have been

predefined. It contains the code that defines the data structures, reset code and echo cancellation subroutines that perform the 16 bit weighting factor versions of the echo canceller. The pipelines for the cancellation and update are shown in the 3 tables below. All of the main loops are coded to have zero memory stalls.

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 16 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)	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	

Table 17 16-Bit Cancellation z0 not word aligned pipeline

(note LDW z-1,z0 and LDW z3,z4 are really the same instruction)

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					MPYLH 9,2,7		SUB 7,8,7	SHR 7,8,0		OR a0,b0,b1	MV b1,a3		STW a3,*a6++
LDW *b6++,b4					MPYSU b4,2,8									
					MPYHSLU			SUB	CLR					
	LDH					MPYLH								
								B Loop	Dec Loop					

Table 18 16-Bit Update Pipeline

(A6 saves in old B6 and is 28 ram locations behind to avoid memory stalls)

5.2.3 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”, have been predefined. It contains the code that defines the data structures, reset code and echo cancellation subroutines that perform the 32 bit weighting factor versions of the echo canceller. The pipelines for the cancellation and update are shown in the 2 tables below. All of the main loops are coded to have zero memory stalls.

1	2	3(1)	4(2)	5(1)	6(2)	7(1)	8(2)	9(1)	10(2)	11(1)
		LDH *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 19 32-Bit Cancellation pipeline

1	2	3	4(1)	5(2)	6(3)	7(1)	8(2)	9(3)	10(1)	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			

Table 20 32-Bit Update Pipeline

5.2.4 ModemTone.asm

This module defines the C-callable sub-routines ModToneReset and ModTone. The data memory for these routines needs to be reserved by the C-code, as it is only referenced indirectly. There are no user defined .equ’s in these routines.

5.2.5 ToneGenerate.asm

This module defines the C-callable sub-routines InitTone, InitDTMF and ToneGenerate; these routines are used to generate test data only and are not part of the echo control code though they could have other uses. The data memory for these routines needs to be reserved by the C-code, as it is only referenced indirectly. There are no user defined .equ's in these routines.

6. Acronyms

DECT	Digital Enhanced Cordless Telecommunications
IIR	Infinite Impulse Response
ISDN	Integrated Services Digital Network
LMS	Least Mean Squared
NLP	Non-Linear Processor

7. References

ITU Recommendation G.164

General Characteristics of International Telephone Connections and International Telephone Circuits. Echo Suppressers

ITU Recommendation G.165

General Characteristics of International Telephone Connections and International Telephone Circuits. Echo Cancellers

ITU Recommendation G.711

Pulse code modulation (PCM) of voice frequencies

ITU Recommendation G.712

Transmission performance characteristics of pulse code modulation

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 of Echo Control Devices for both Manually and Automatically Established Calls.

ETSI 300 175-*

Digital Enhanced Cordless Telecommunications. (DECT)

TMS320C62xx CPU and Instruction Set

TMS320C62xx Programmers Guide