VS1063A APPNOTE: AEC WITH VS1063A

Acoustic Echo Cancellation with VS1063a

All information in this document is provided as-is without warranty. Features are subject to change without notice.

Revision History			
Rev.	**Date**	**Author**	**Description**
1.21	2014-02-20	HH	Minor corrections.
1.20	2014-02-19	HH	Added Chapter 5, error fixes to Chapter 8.
1.10	2013-10-16	HH	Added Chapters 11 and 12.
1.01	2013-10-10	HH	Added parametric.encoding.aecAdaptMultiplier.
1.00	2013-10-09	HH	Initial version.

Contents

List of Figures

1 Introduction

A speaker phone application is a system which includes a loudspeaker and a microphone, as presented in Figure 1. The loudspeaker is used to present the sound from a far-end source, and the microphone is used to pick up the signal from a near-end source (a).

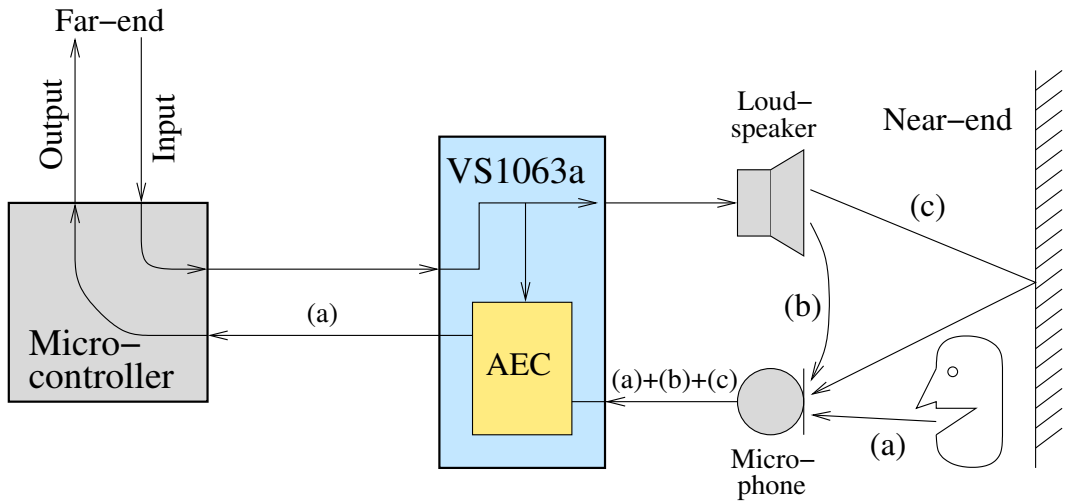

Figure 1: Speaker phone.

However, inevitably the microphone will also pick up some audio from the loudspeaker, both directly (b) or indirectly as e.g. echoes from walls (c). Sometimes this pickup can be very strong, even stronger than the signal (a) that the microphone was supposed to pick up. If nothing is done to signal (b) and (c), the human at the far-end will hear his/her speech as an echo, which is both distracting and may make conversation difficult, or even impossible. The echo signals (b) and (c) are dependent on the device use as well as the environment in which it is used. So when the speaker phone is moved, or if anything or anyone moves in the room where the speaker phone is located, echo signals (b) and (c) will also change.

VS1063a Acoustic Echo Cancellation (AEC) is an adaptive signal processing algorithm which removes as much as possible from the echo signals (b) and (c) as possible. At best, the signal processing attenuation can be in the range of 40-50 dB. This will make it possible for the speaker phone to cleanly pick up the intended user signal (a) in situations where it would otherwise be impossible.

AEC does not combat acoustic feedback caused by the proximity of two devices.

For more details on AEC, see Chapter 5, *Introduction to What AEC Is and What It Does*.

Chapters 1 through 11 explain how to set up, use, and verify AEC with VS1063a. Chapter 12 explains in a case study how good acoustic design can make the job of AEC easier and thus create better devices. Finally, Chapters 13 and 14 contain a version history and VLSI's contact information, respectively.

2 Definitions

ADC Analog to digital converter.

AGC Automatic gain control.

AEC Acoustic echo cancellation.

DAC Digital to analog converter.

dB Decibel, a logarithmic unit that indicates the ratio of two powers. 1 bel, which equals 10 decibels, is a power ratio of 10.

FES Far-end signal.

FES' Distorted Far-end signal as picked up by a microphone.

FES" AEC-cleaned version of FES' (or, if AEC is in pass-through mode, FES' without modifications).

LSb Least significant bit.

LSB Least significant byte.

MSb Most significant bit.

MSB Most significant byte.

NES Near-end signal.

SCI Serial Command Interface. Used to control VS1063a.

SDI Serial Data Interface. Used to write data to VS1063a.

3 Features and Limitations

VS1063a has the following features and limitations:

- Echo cancellation of upto 50 dB.
- Echo cancellation buffer size:
  - 128 ms for 8 kHz.
  - 85 ms for 12 kHz.
  - 40 ms for 16 kHz.
  - 16 ms for 24 kHz.
  - Other sample rates will not work.
- Cold start adaptation time typically approximately 10 seconds.
- Hot start adaptation time typically a few seconds.
- Echo cancellation coefficients may be stored and retrieved for more effective and much faster Hot Start.
- Requires 12.288 MHz (default for VS1063a), or 24.576 MHz crystal.
- Does not perform feedback elimination between two devices that are close to each other, as shown in Figure 3 on Page 9.

4 Prerequirements

The following prerequirements need to be met so as to be able to get reliable results.

4.1 Hardware

For testing and tuning VS1063a AEC you will need a VS1063a system capable of the functionality as shown in Figure 1 on page 4. For best tuning, you will need to be able to insert a 30-second test signal to the Far-end Input while recording from the Far-end Output.

Specific hardware requirements are:

- Exactly 12.288 MHz or 24.576 MHz crystal. Other clocks will not work.
- Microphone which is always at its linear range with the the voices excited by echo signals (b) and (c). User signal (a) may occasionally overflow the microphone input. See Figure 1 on Page 4 for what signals (a), (b), and (c) are.
- Loudspeaker which must always be operated at its linear range. AEC cannot filter away echo signals that are caused by the loudspeaker's harmonic distortion or by any rattles or other noises the speaker phone chassis makes when playing audio.

4.2 Software

The following software is required:

- VS1063a Patches v1.60 or higher, available at *http://www.vlsi.fi/en/support/software/vs10xxpatches.html* .

5 Introduction to What AEC Is and What It Does

5.1 An Acoustic Echo Model

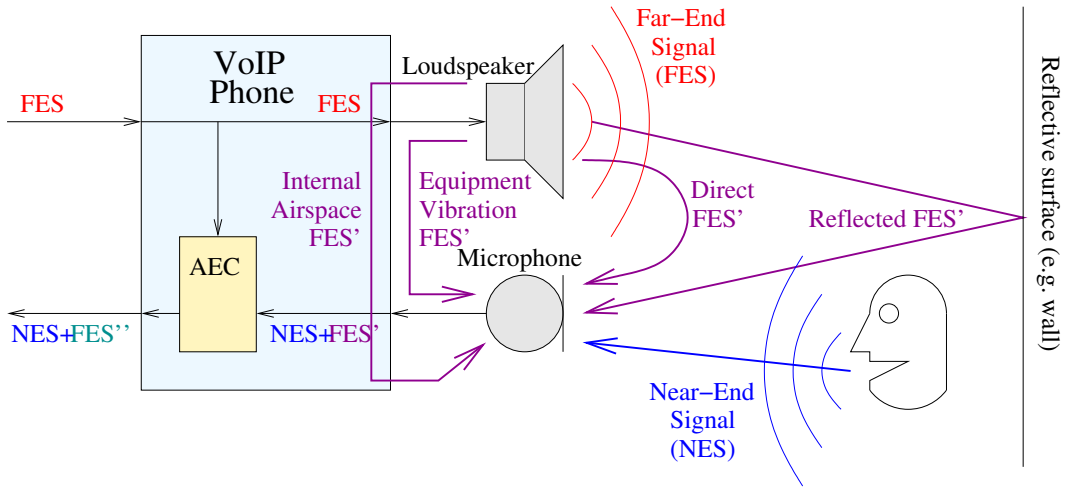

Figure 2: Speaker phone signals.

In the context of speaker phones, acoustic echo is considered to be audio that is by some means transmitted from the loudspeaker of a phone back to its microphone and thus back to the speaker phone and potentially to the other party.

Speaker phone audio signals, including four mechanisms for echo are shown in Figure 2.

- The Near-End Signal (NES) is the part of the audio signal that the microphone actually *is supposed to* capture and pass on.
- Direct FES' (Far-End Signal) is the part of the speaker signal that is transmitted directly from the loudspeaker through the air to the microphone. This can be minimized by designing the device with a directional loudspeaker faced away from the microphone, using a directional microphone, and by putting the loudspeaker and microphone as far away from each other as possible.
- Equipment Vibration FES' is usually caused by the main chassis, which vibrates along with the speaker. This vibration may be transmitted to the microphone and cause very high microphone signal levels.
- Internal Airspace FES' is sound that occurs inside a unit chassis. There may be structures which boost sound levels and may cause very high microphone signal levels.
- Reflected FES' is the part of the loudspeaker signal that gets reflected from different surfaces and gets transmitted back to the microphone.

Because of these different echo mechanisms, the microphone actually does not only pick up the NES signal, but also a distorted version of the FES signal, called FES'. So the total signal picked up by the microphone is NES + FES'.

Using information of what the original, undistorted FES signal was like, the AEC algorithm then tries to determine which parts of the NES + FES' signals are NES and which are FES'. Then it tries to remove the FES' part as well as possible. The output of the AEC algorithm is called NES + FES". If AEC is working properly, the amplitude of FES" is much lower than FES'. If AEC is switched off to pass-through mode, FES" = FES'.

Better audio design can affect the amount and quality of the FES' signal. The lower and the less distorted the FES' signal is, the better job AEC can create to minimize FES".

5.2 AEC Adaptation Time

AEC is an adaptive algorithm. When it is started for the first time, it knows nothing of the acoustic system and its echo. It starts learning about the echo, and adapts its filters in such a way that will remove as much of the acoustic echo as possible. This adaptation typically needs some 10 seconds of Far-End Signal (FES), but depending on the signal it can also be faster or slower. Typically speech signals are better for adaptation than pure sine tones. During this learning period AEC is not very effective.

To avoid every conversation starting with a learning sequence, it is possible to store the AEC Coefficients when a conversation ends (see Chapter 10). This makes it possible for AEC to be much more effective at the beginning of the next call.

5.3 AEC Is Not Feedback Elimination

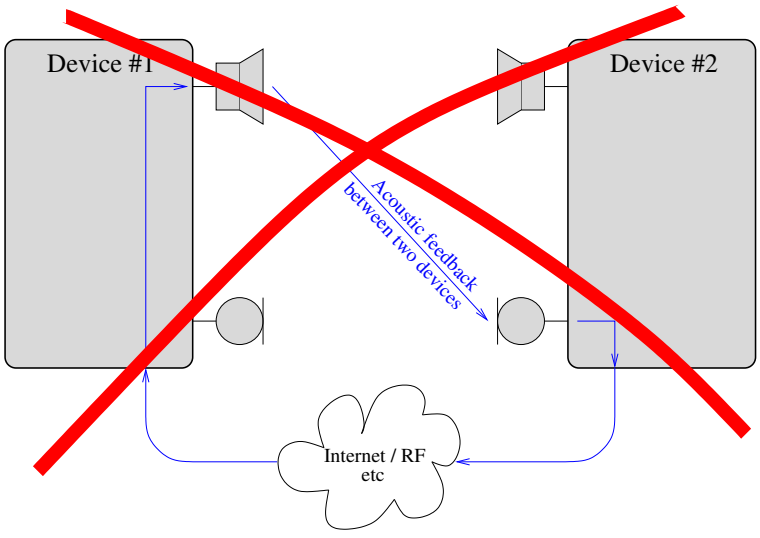

Figure 3: AEC does not combat audio feedback between devices close to each other.

AEC does not perform feedback elimination between two devices that are close to each other as shown in Figure 3.

5.4 AEC Example: Door Phone Conversation

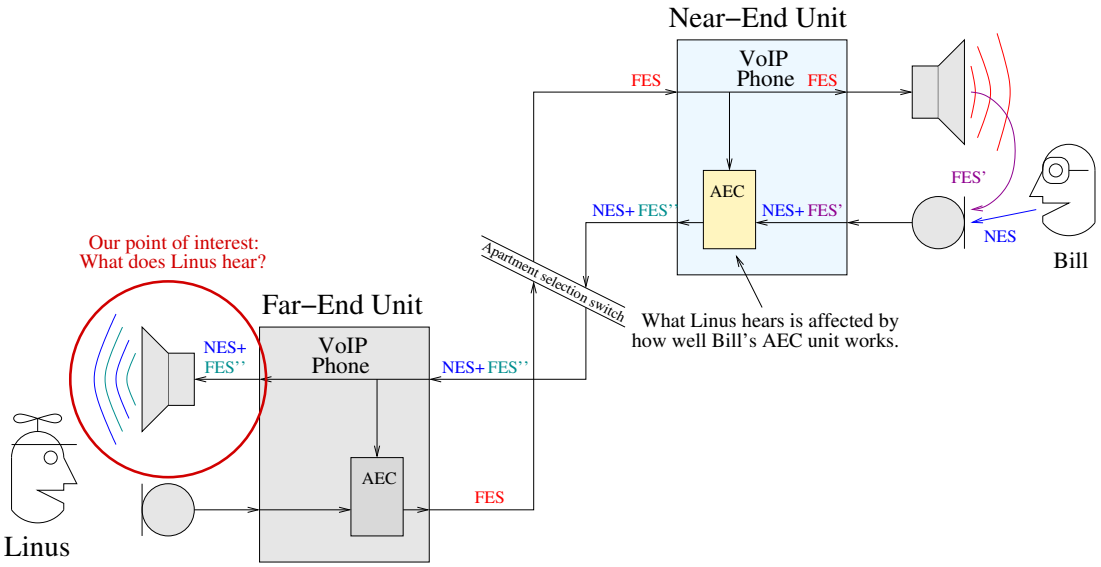

Figure 4: Bill and Linus' door phone conversation.

In this example, which is presented in Figure 4, Linus is coming to meet his friend Bill who lives in an apartment at the thirteenth floor. Linus calls Bill at the bottom floor doorphone, then they have a short conversation after which Bill unlocks the door.

We will concentrate on what Linus, who is at the Far-End from our point of view, hears from his speaker (NES+FES", red circle), and how that depends on Bill's phone's AEC.

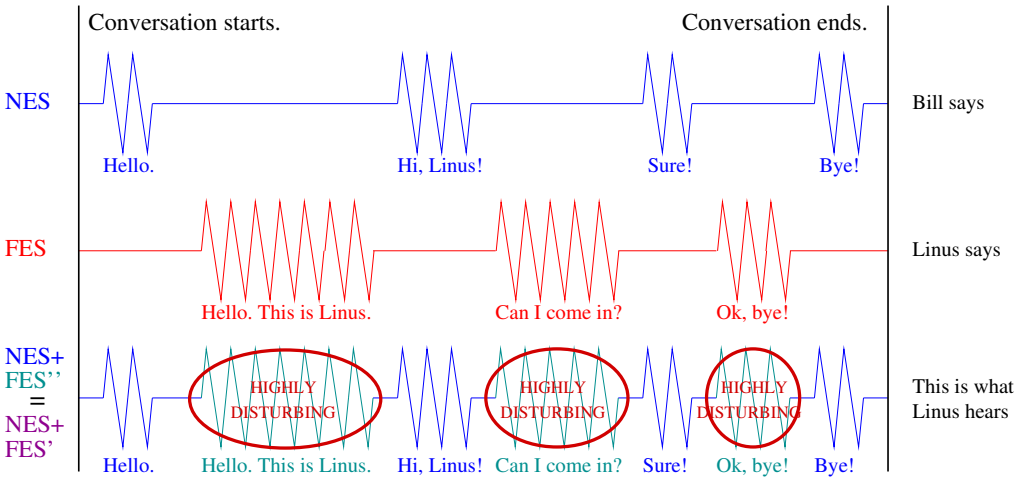

Figure 5: Example conversation without AEC.

In Figure 5 we see how the conversation goes if Bill's device doesn't have AEC at all. The bottom line shows what Linus hears from his speaker. Because Bill's unit doesn't use AEC, Linus' dialogue (FES) is echoed back to him as the FES" part of the NES+FES" signal. This is highly disturbing, and may cause great difficulty in having a conversation. It can also cause an acoustic feedback loop.

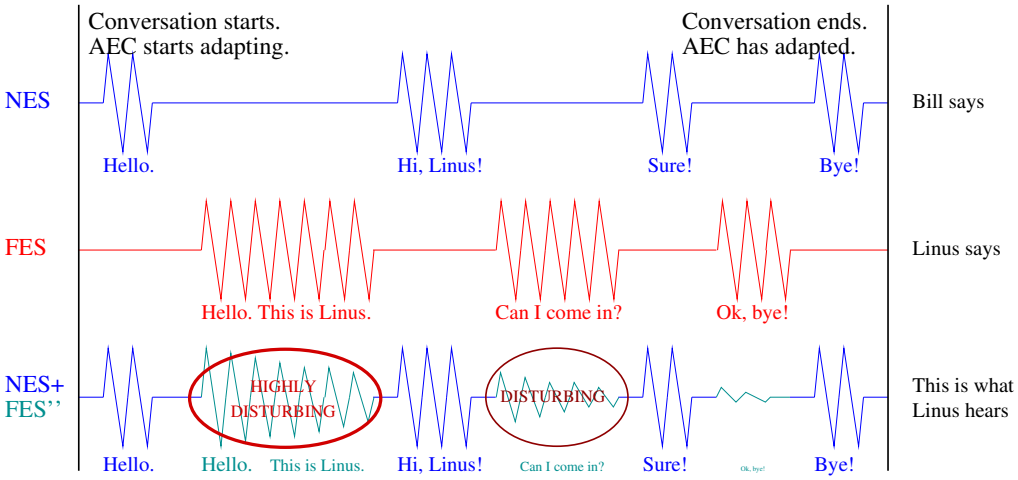

Figure 6: Example conversation with AEC cold start.

In Figure 6 Bill's phone uses cold start AEC. At the beginning of the conversation Linus hears just as much echo from his own sound as before, but as the conversation goes on, Bill's phone's adaptive AEC algorithm learns to differentiate between NES and FES signals, and starts suppressing unwanted FES" portion of the signal.

Cold start is acceptable if an audio connection is in continous use. However, for a door phone where conversations are short, full adaptation at every call is not satisfactory.

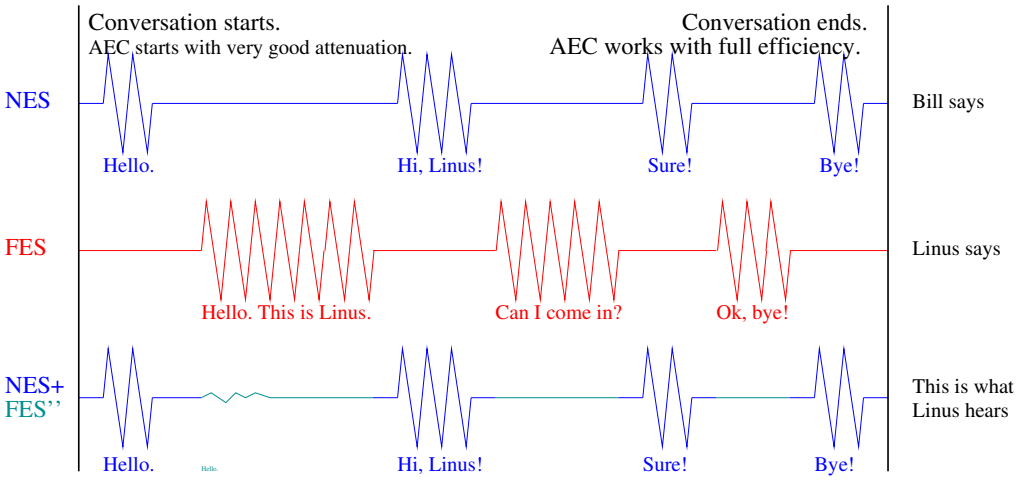

Figure 7: Example conversation with AEC hot start.

After the phone conversation shown in Figure 6, it is possible to store the AEC Coefficients (see Chapter 10 for details). These coefficients can then be loaded when the next conversation starts. If this is done, the next conversation is going to be like what is shown in Figure 7. In this case, the conversation starts with almost perfect echo cancellation: Linus hears only what Bill says, and no echo of his own speech.

To get this best performance, always store the AEC coefficients when a phone conversation ends, and reload them when a new one starts.

6 VS1063a AEC Signal Paths

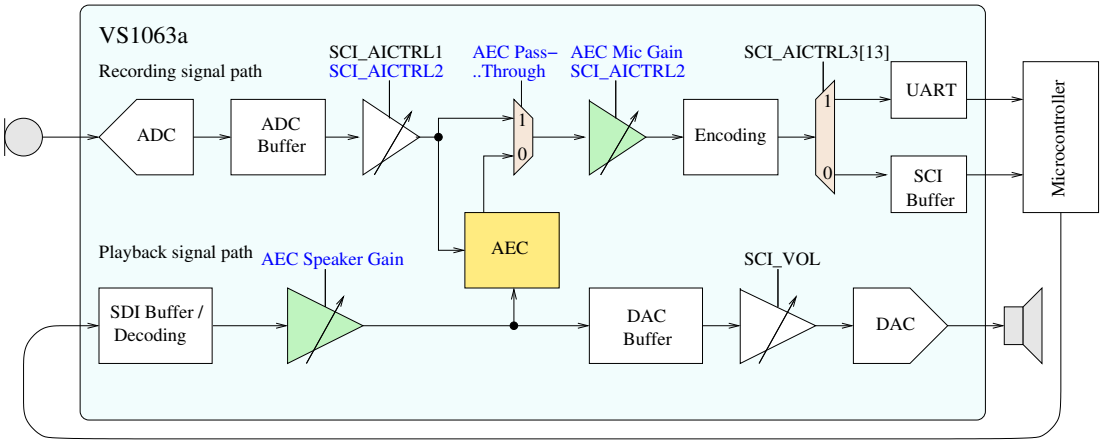

Figure 8: VS1063a signal paths in AEC mode.

The VS1063a signal paths when running in AEC mode are presented in Figure 8. Registers that may be changed while running AEC are presented in blue.

The microphone signal is digitized by the ADC and put to the ADC Buffer.

Recording gain is applied from SCI_AICTRL1. For AEC applications, SCI_AICTRL1 should *never* be set to zero.

If AEC Pass-through is zero, the AEC algorithm is applied to the recording signal.

User gain can be set with AEC Mic Gain. If AEC Mic Gain is zero, Automatic Gain Control (AGC) is used with a maximum value as set in SCI_AICTRL2. For AEC applications, this is the right place to set AGC if required.

After encoding to the requested audio format, and depending on SCI_AICTRL3 bit 13, output is either sent through the UART, or to the SCI Buffer, from where it can be read with the Microcontroller.

The Microcontroller sends playback audio to the VS1063a through SDI.

When data is read from the SDI buffer and decoded, AEC Speaker Gain is applied to it. This is the place where volume control should be applied.

Playback data is sent on the other hand to the AEC algorithm, and on the other hand to the DAC buffer.

Volume control controlled by register SCI_VOL is applied to the signal. This register should *not* be used when operating in AEC mode.

Finally, the playback audio is converted to analog from by the DAC, and sent to the loudspeaker.

7 AEC Configuration Fields

The AEC has several configuration fields which are located in the X data memory of VS1063a. To access the X data memory, run the following SCI commands.

To read *data* from X:*addr*:
WriteSci(SCI_WRAMADDR, *addr*);
data = ReadSci(SCI_WRAM); /* Can be repeated to read consecutive addresses */

To write *data* to X:*addr*:
WriteSci(SCI_WRAMADDR, *addr*);
WriteSci(SCI_WRAM, *data*); /* Can be repeated to write consecutive addresses */

X Mem Addr	Name	Description
0x2021	AEC NLMS Gain	AEC Coefficients scaling term
0x2022	AEC Mic Gain	Adjust output signal volume, 0x400 = 1
0x2023	AEC Speaker Gain	Adjust playback volume, 0x400 = max = 1
0x2024	AEC Pass-through	0 = AEC active, non-0 = Pass-through
0x1000..0x13ff	AEC Coefficients	Adaptive coefficients for AEC

NLMS Gain can be stored and restored along with AEC Coefficients for better startup times. For details, see Chapter 10, *Saving and Loading AEC Coefficients*).

The volume of the signal that is output to the Far-end can be adjusted with AEC Mic Gain. If AEC Mic Gain is zero, then AGC is applied. The maximum AGC Gain is read from register SCI_AICTRL2.
Example: if you want to use AGC with a maximum gain of +12 dB, set AGC Gain to 0, and register SCI_AICTRL2 to 0x1000 (0x1000 = 4×0x400 = +12 dB).

Loudspeaker volume can be adjusted with AEC Speaker Gain.

If AEC Pass-Through is set to a non-zero value, the AEC algorithm is disabled. This can be used when calibrating a system and for comparing the effectiviness of AEC.

AEC Coefficients hold, along with NLMS Gain, the information AEC has gathered of the echo characteristics of the surroundings of the device.

8 Initializing AEC

To initialize AEC, run the following operations:

1. Load and the VS1063a Patches Package. You don't need to start it yet (but it doesn't hurt).
2. Set clock speed register to 67.584 MHz:
   WriteSci(SCI_CLOCKF, 0xc800);
3. Set hardware volume. If your loudspeaker can play back full volume without distortion, that is a good default for benchmarking:
   WriteSci(SCI_VOL, 0x0000);
4. Set sample rate (8, 12, 16, or 24 kHz). Example for 8 kHz:
   WriteSci(SCI_AICTRL0, 8000);
5. Set linear encoding gain. Unless for very good reasons, set it to 1x gain. Never set it to AGC mode (0):
   WriteSci(SCI_AICTRL1, 0x400);
6. If you intend to use the UART output, set the UART speed. You can use formulas explained in the VS1063a Hardware Guide to calculate the coefficients. Examples: for 115200 bps the value is 0x5307, for 460800 bps the value is 0x1407. The example below is for 460800 bps:
   WriteSci(SCI_WRAMADDR, 0x1e2a);
   WriteSci(SCI_WRAM, 0x1407);
   WriteSci(SCI_WRAM, 0x0);
   WriteSci(SCI_WRAM, 0x0);
   If you use SCI, you don't need to do this step.
7. Set input/output audio format as instructed in the VS1063a Datasheet:
   - 16-bit PCM, left channel mono, UART output: WriteSci(SCI_AICTRL3, 0xe412);
   - 16-bit PCM, left channel mono, SCI output: WriteSci(SCI_AICTRL3, 0xc412);
   - 8-bit g.711 μlaw, left ch mono, UART output: WriteSci(SCI_AICTRL3, 0xe422);
   - 8-bit g.711 μlaw, left ch mono, SCI output: WriteSci(SCI_AICTRL3, 0xc422);
   - 8-bit g.711 A-law, left ch mono, UART output: WriteSci(SCI_AICTRL3, 0xe432);
   - 8-bit g.711 A-law, left ch mono, SCI output: WriteSci(SCI_AICTRL3, 0xc432);
   - 4-bit IMA ADPCM, left ch mono, UART output: WriteSci(SCI_AICTRL3, 0xe402);
   - 4-bit IMA ADPCM, left ch mono, SCI output: WriteSci(SCI_AICTRL3, 0xc402);
8. Set mode register. Example for New Mode and with mic amplifier:
   WriteSci(SCI_MODE, 0x1c00);
9. Set parametric.encoding.aecAdaptMultiplier which lies at X memory address 0x1e2d. Unless for good reasons, use default value 2:
   WriteSci(SCI_WRAMADDR, 0x1e2d); WriteSci(SCI_WRAM, 0x2);

10. Activate VS1063a Patches Package in codec/AEC mode:
    WriteSci(SCI_AIADDR, 0x50);

11. Wait until DREQ is high, or 1 ms.

12. If you want to load precalculated AEC coefficients, do it here. See Chapter 10.2, *Loading AEC Coefficients*, for details.

13. Set AEC Microphone (NES+FES') Linear Gain to the AEC Mic Gain register at X address 0x2022. A good initial value is 1x, which is 0x400:
    WriteSci(SCI_WRAMADDR, 0x2022); // Corrected for document v1.20
    WriteSci(SCI_WRAM, 0x400);
    If you set the register to 0, AGC limited by register SCI_AICTRL2 is used. If you, for instance, want to use AGC at a maximum gain of +12 dB, do as follows:
    WriteSci(SCI_WRAMADDR, 0x2022); // Corrected for document v1.20
    WriteSci(SCI_WRAM, 0x0);
    WriteSci(SCI_AICTRL2, 4*0x400); // 4x = +12 dB, corrected for document v1.20
    This step can later be repeated to change microphone gain on the run.

14. Set AEC Speaker (FES) Linear Gain to the AEC Speaker Gain register at X address 0x2023. A good initial value is 1x, which is 0x400:
    WriteSci(SCI_WRAMADDR, 0x2023); // Corrected for document v1.20
    WriteSci(SCI_WRAM, 0x400);
    This step can later be repeated to change loudspeaker volume.

9 Reading and Writing Data

This Chapter presents pseudo-code for how to read and write data. Depending on whether you use SCI or UART for data capturing, your main loop should include one of the read methods as presented in Chapters 9.1 and 9.2, and the write method from Chapter 9.3.

The pseudo-code assumes that you have created the following functions:

void WriteSci(u_int8 addr, u_int16 data);
Writes 16-bit value to an SCI register, big-endian (MSB first).

u_int16 ReadSci(u_int8 addr);
Reads and returns 16-bit value from an SCI register, big-endian (MSB first).

void WriteSdiByte(const u_int16 data);
Writes 8-bit value to the SDI bus.

u_int8 DReqIsHigh();
Returns non-zero if VS1063a pin DREQ is high.

u_int8 ThereAreUartBytes(void);
Returns non-zero if there is data waiting in the microcontroller's UART peripheral.

u_int16 GetUartByte(void);
Reads byte from microcontroller's UART peripheral.

void SendExtByte(u_int8 b);
Send byte to external medium (internet, RF link, etc).

s_int16 ThereAreExtBytes(void);
Returns non-zero if there are bytes in the microcontroller's external medium buffer.

s_int16 GetExtByte(void)
Reads byte from external medium (internet, RF link, etc).

9.1 Reading Data from SCI

This applies if you use the SCI (SPI) bus to transfer your audio data from VS1063a to your microcontroller.

```
while (1) {
  u_int16 n;
  /* Do other things */

  /* See if there is data to send */
  if ((n = ReadSci(SCI_HDAT1)) > 0) {
    /* Send data bytes. */
    for (i=0; i<n; i++) {
      u_int16 w = ReadSci(SCI_HDAT0);
      SendExtByte((u_int8)(w >> 8));   /* Big-endian, so MSB first */
      SendExtByte((u_int8)(w & 0xFF)); /* Big-endian, so LSB last  */
    }
  }
}
```

9.2 Reading Data from UART

This applies if you use the UART to transfer your audio data from VS1063a to your microcontroller.

```
while (1) {
  u_int16 n;
  /* Do other things */

  /* See if there is data to send */
  while (ThereAreUartBytes()) {
    SendExtByte(GetUartByte());
  }
}
```

9.3 Writing Data to SDI

This code is used to transfer data from your microcontroller to VS1063a using the SDI (SPI) bus.

```
while (1) {
  /* Do other things */

  /* See if there is data to send */
  while (ThereAreExtBytes() && DReqIsHigh()) {
    WriteSdiByte(GetExtByte());
  }
}
```

10 Saving and Loading AEC Coefficients

AEC convergence may be made much faster by using a pre-saved set of AEC coefficients. This will enhance the experience of the user because in typical cases echo cannot be heard even during the first seconds of a conversation.

Note that you need to have a separate set of AEC coefficients for each sample rate you use. If you use an incorrect set of coefficients, AEC will take a longer time to converge and work properly.

10.1 Saving AEC Coefficients

First start AEC and run it for as long as needed for full convergence. Typically 30 seconds with far-end speech only is enough. Then run the following pseudo-code on your microcontroller:

```
/* Save NLMS gain */
WriteSci(SCI_WRAMADDR, 0x2021);
write_to_microcontroller_file(ReadSci(SCI_WRAM));

/* Save AEC coefficients */
WriteSci(SCI_WRAMADDR, 0x1000);
for (i=0; i<1024; i++) {
  write_to_microcontroller_file(ReadSci(SCI_WRAM));
}
```

10.2 Loading AEC Coefficients

To load AEC coefficients, first start AEC. Then run the following pseudo-code before sending any audio data:

```
/* Load NLMS Gain */
u_int16 t;
WriteSci(SCI_WRAMADDR, 0x2021);
t = read_from_microcontroller_file();
if (t < 0x400) { /* Shouldn't happen, and 0 is a non-recoverable error, */
  t = 0x400;     /* ... so write default value to be sure.              */
}
WriteSci(SCI_WRAM, t);

/* Load AEC coefficients */
WriteSci(SCI_WRAMADDR, 0x1000);
for (i=0; i<1024; i++) {
  WriteSci(SCI_WRAM, read_from_microcontroller_file());
}
```

11 Testing AEC

To test if and how AEC is working, first send a known signal from the far-end to your system when in AEC mode, but with AEC Pass-Through register activated. Measure how much of that signal gets back to the far-end. (See Figure 1 on Page 4 for a Speaker phone example system.)

After you have done your measurement, clear the AEC Pass-Through registers, and continue sending the far-end signal. After a few seconds, the echo signal that gets sent back to far-end should be getting significantly reduced. If this doesn't happen, AEC has not been set up properly or isn't working because of some other reason.

Notice also that acoustic design affects a lot how efficient AEC can be. See Chapter 12, *AEC Acoustic Design: How to Combat Acoustic Echo*, for details.

12 Combating Echo with Acoustic Design: A Case Study

This chapter aims to answer to the question: How can you help combating acoustic echo with better acoustic design?

When a product is built, there has to be a chassis containing the electronics. Building this chassis creates several challenges to audio design. While not a comprehensive audio design guide by any means, this chapter will discuss an example product one of VLSI's customers have had, and will show some suggestions on how to solve audio echo problems. Although the product in question didn't use VS1063, the results are applicable.

12.1 Case Study Product

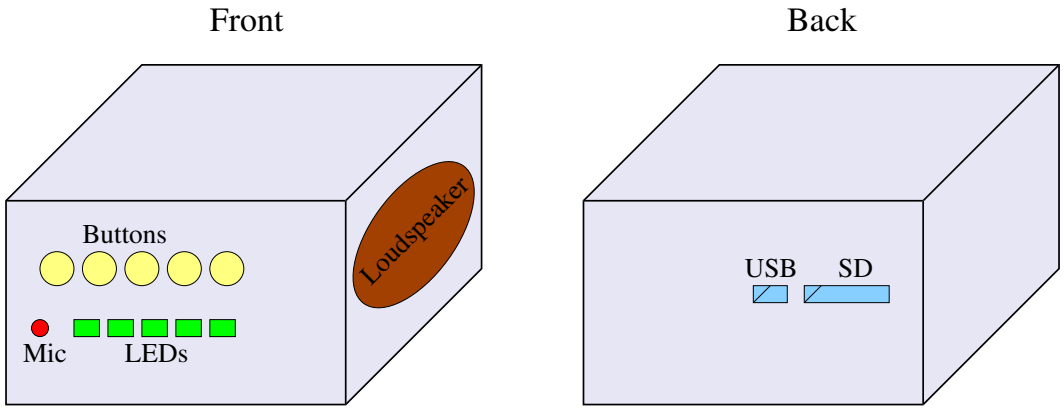

Figure 9: Case study speaker phone chassis.

The product is a speaker phone, which is in principle as depicted in Figure 9. The user interface (buttons and LEDs) and the microphone are on the front side. USB and SD connectors live on the back side.

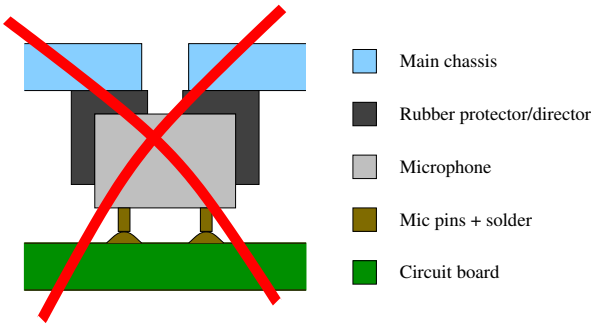

Figure 10: Case study speaker phone microphone encapsulation.

Figure 10 shows how the microphone is been connected. On the other side the microphone has been soldered to a circuit board, and on the other side it has been connected

to the main chassis through a rubber protection / director cap. It will later be shown why the microphone shouldn't be connected in this way in a real product. (If you want to see the better setup immediately, see Figure 19 at Page 28.)

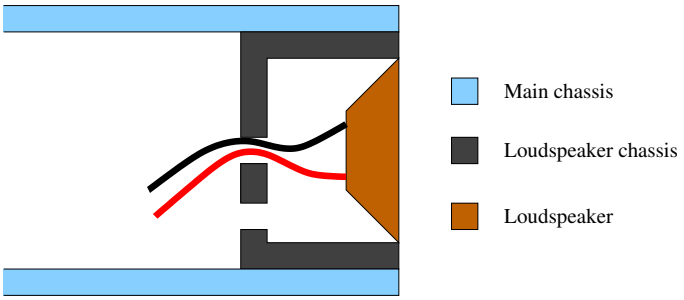

Figure 11: Case study speaker phone loudspeaker encapsulation.

Figure 11 shows that the loudspeaker has been installed in a separate, plastic chassis of its own. The chassis has relatively thin walls and two holes: one for allowing speaker wires to come through, and another for other purposes.

The reported problem with the product was that echo cancellation didn't work properly. The reasons and solution are discussed in this case study.

12.2 Case Study Measurement Methodology

To determine the amount of speaker sound picked up by the microphone (FES'), the following measurement methodology was used:

1. AEC was set to pass-through mode. In that mode, the software does not perform echo cancellation or any other post-processing to the input signal except for a DC removal filter. Thus it will accurately represent what was actually recorded by the microphone (FES" = FES').
2. Wideband white noise with an amplitude of -20 dB of maximum was sent as FES.
3. Microphone input was recorded for ≈10 seconds to a 16-bit, 8 kHz file.
4. The power and spectrum of the file was analyzed.

12.3 Reference 0.0 dB: VLSI Solution's Demonstration Board

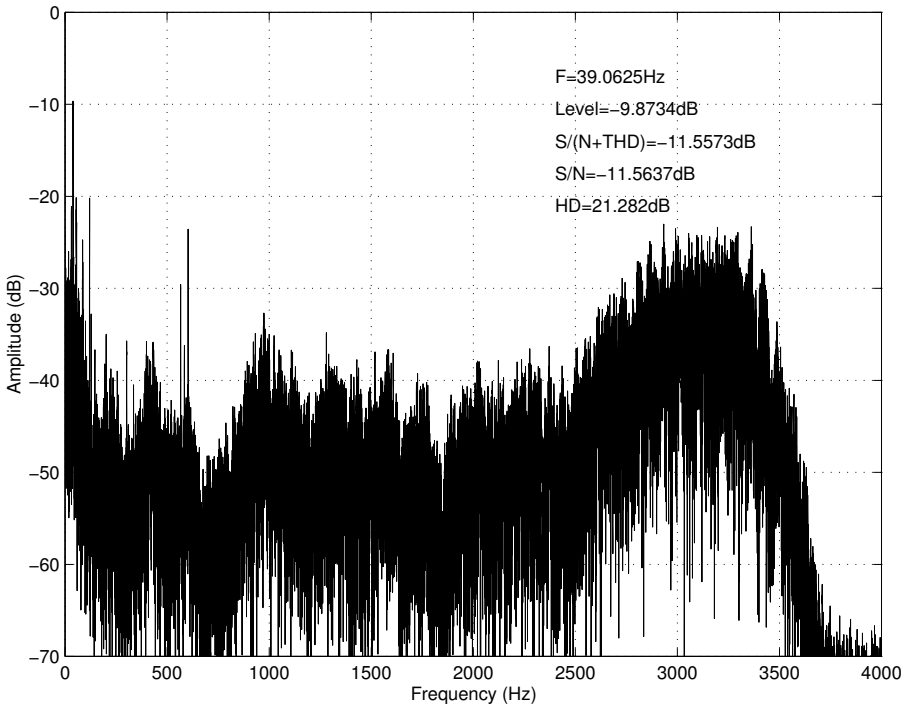

Figure 12: Case study: 0.0 dB: VLSI Solution's reference board.

As a reference, VLSI Solution's own demonstration board setup was measured. The frequency spectrum of this measurement is presented in Figure 12. The echo floor is very low between 200...2500 Hz, though it rises over 10 dB for frequences between 2700 and 3400 Hz.

Overall, this is performance that ensures that echo cancellation works exactly as intended. However, this is a demonstration board, and it is not realistic to expect a device with a cramped chassis to act as well. The question is, however, how close can we get in performance to this close to ideal setup?

12.4 Meas +30.5 dB: The Example Unit As It Originally Was

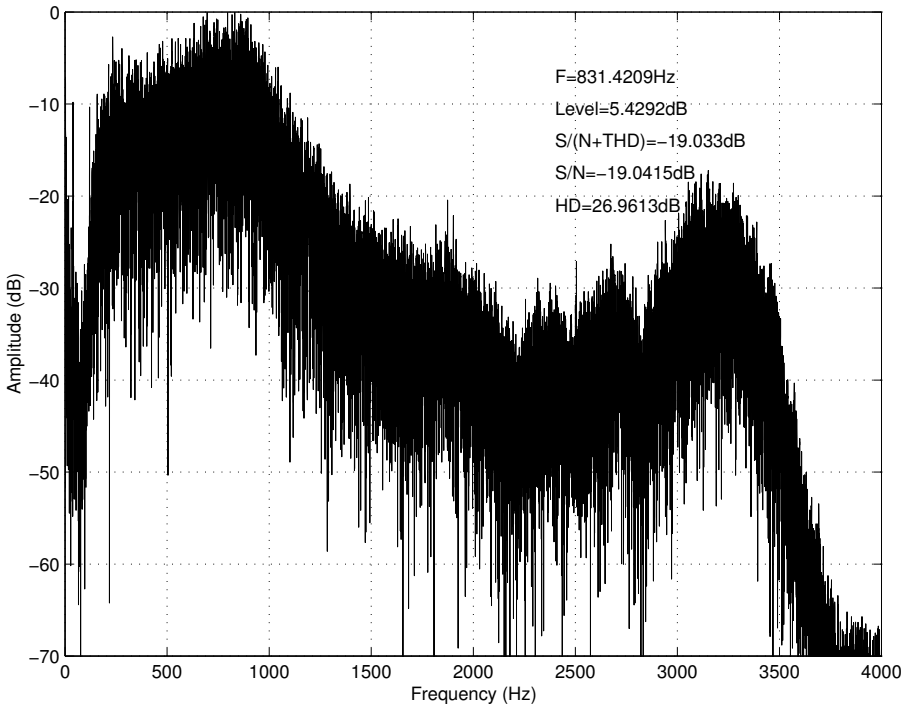

Figure 13: Case study: +30.5 dB: The example unit as it originally was.

First the unit was tested as it was, without any modifications except for activating the pass-through mode. The spectrum is presented in Figure 13. As can be seen, frequencies between 200. . . 1400 Hz are greatly boosted.

The overall signal power was 30.5 dB stronger than the reference and created problems for AEC.

12.5 Meas +16.0 dB: No Chassis

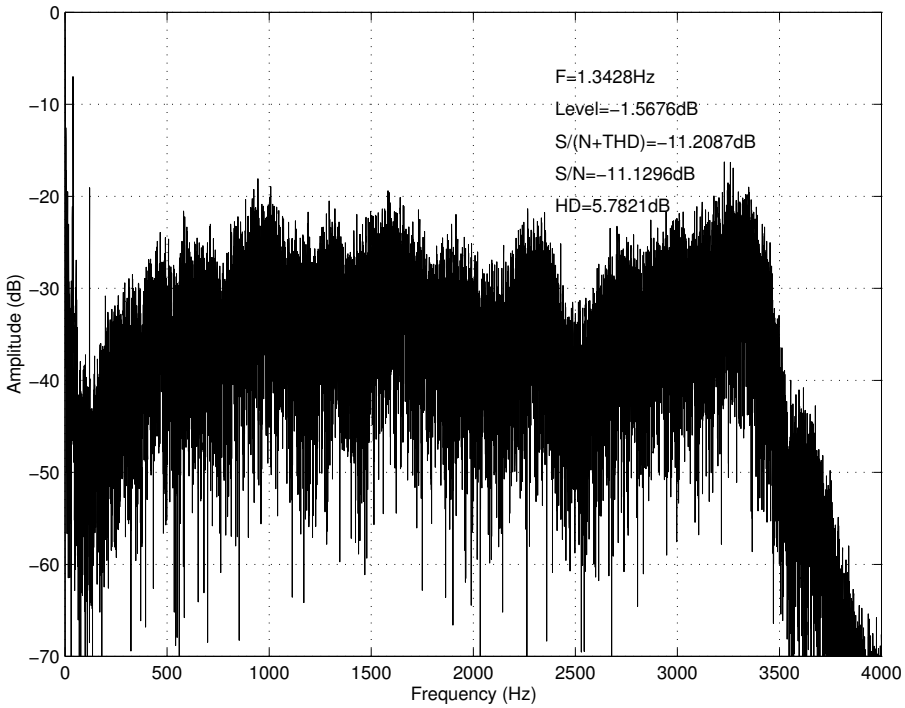

Figure 14: Case study: +16.0 dB: No chassis.

The main chassis was removed in such a way that the speakers and microphone still were at the same relative positions as before. This resulted in a much enhanced performance as seen in Figure 14. All the peaks are gone and the resulting frequency response is relatively straight.

The signal's power has decreased to +16.0 dB. Although it still was much greater than what was with the reference unit, the difference was big enough that AEC now worked properly.

This solution was of course not practical in itself, because a product needs to have a chassis. Nevertheless, now it was clear that the main chassis somehow acted as a sound conductor, because removing it decreased FES' so significantly.

The chassis was put together again for the next tests.

12.6 Meas +28.5 dB: Speaker Padding

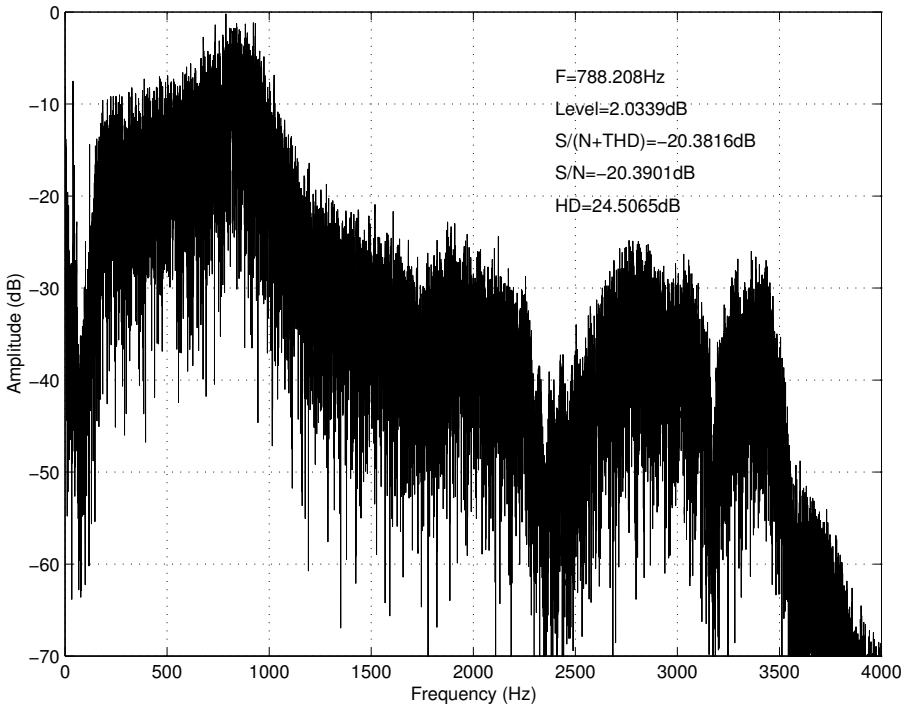

Figure 15: Case study: +28.5 dB: Speaker padding.

It was felt that the back wall of the speaker element might be vibrating too much, so some mass was added to the back of the inside of the speaker chassis to make it a little bit heavier. Also the speaker chassis holes were filled so that the speaker would be a little bit more like what is shown in Figure 16.

This resulted in noticeable attenuation at frequences over 2300 Hz and a total signal level of +28.5 dB as shown in Figure 15. Unfortunately the main problem, the peak between 200. . . 1200 Hz, was unaffected.

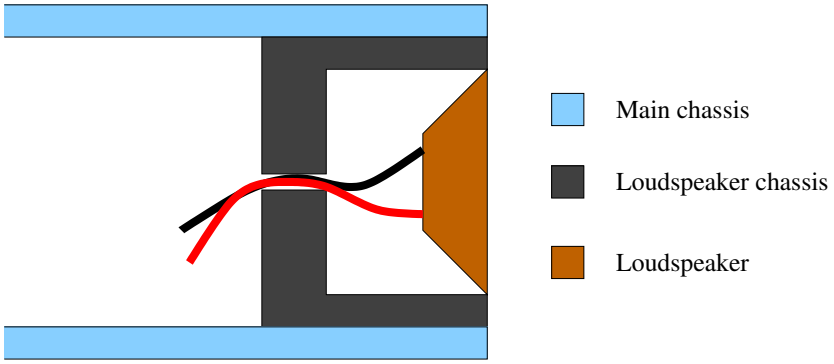

Figure 16: Case study speaker phone loudspeaker with thicker back wall.

12.7 Meas +27.6 dB: Padding around Microphone Circuit Board

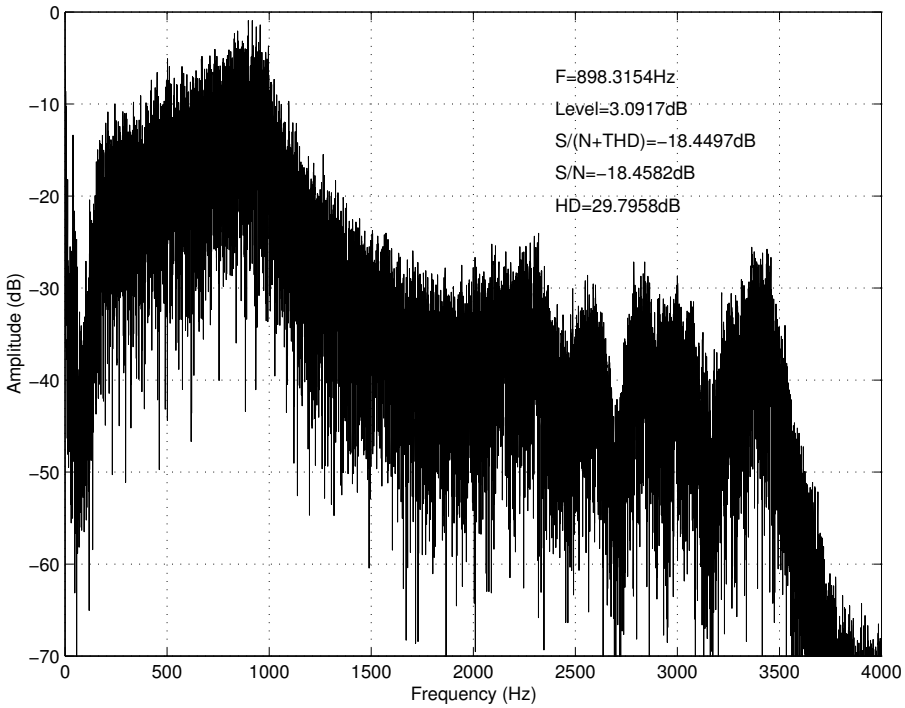

Figure 17: Case study: +27.6 dB: Blu-tack around microphone board.

Low-frequency boost could be caused by Equipment Vibration FES' through the microphone circuit board. Thus, padding material was put both above and below the microphone board to decrease vibrations.

Figure 17 shows that the difference between this and the previous case (Figure 15) was insignificant.

12.8 Meas +21.8 dB: Mic Removed from Circuit Board

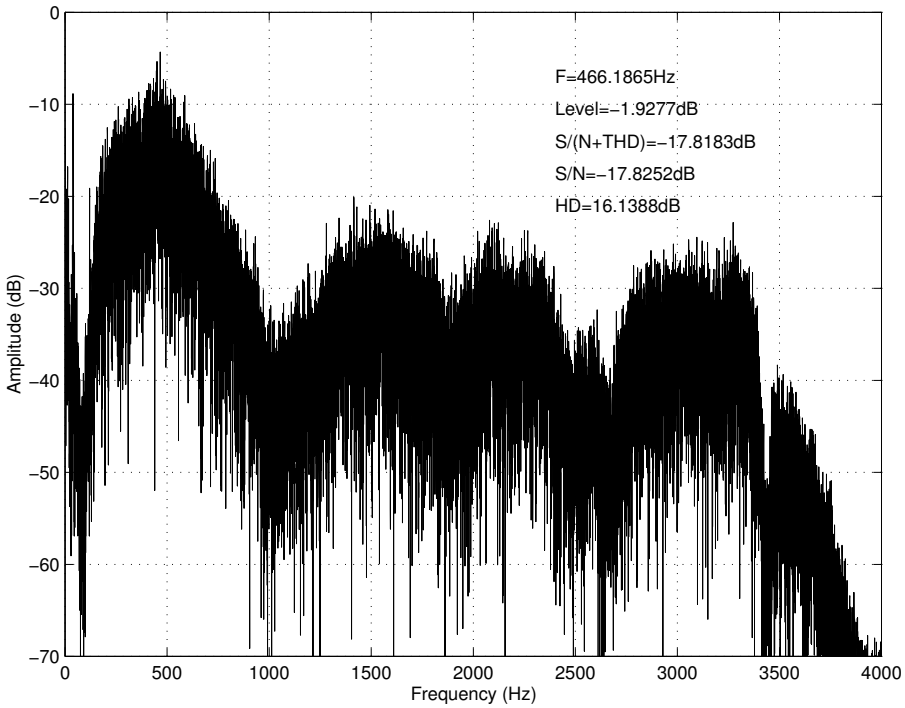

Figure 18: Case study: +21.8 dB: Free-hanging microphone, closed chassis.

To further combat potential Equipment Vibration FES', the microphone was disconnected from the microphone circuit board. Then it was connected back with thin electrical wires as shown in Figure 19. The microphone was secured to the main chassis with adhesive material.

As can be seen in Figure 18, preventing Equipment Vibration FES' from the circuit board helped: while it didn't affect frequencies below 500 Hz, it greatly reduced frequencies in the most problematic area at 800...1300 Hz. Now signal level was $+21.8$ dB.

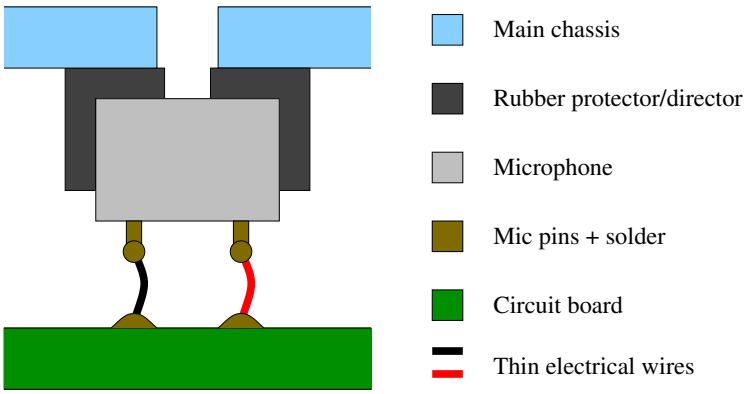

Figure 19: Case study: Better speaker phone microphone encapsulation.

12.9 Meas +16.2 dB: Mic Removed from Circ. Board + Chassis Hole

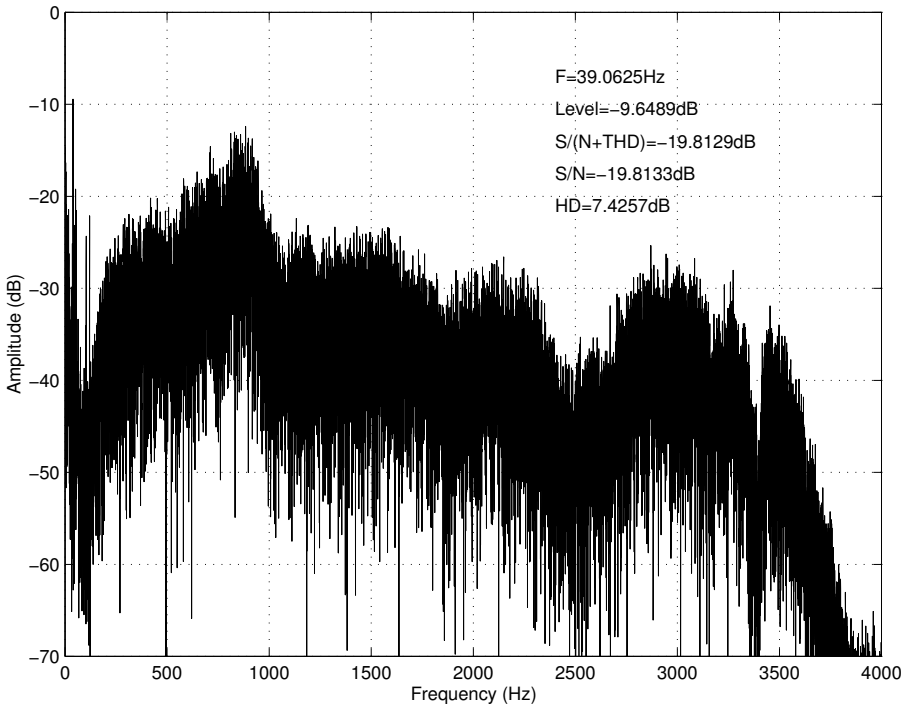

Figure 20: Case study: +16.2 dB: Free-hanging microphone, open chassis.

Removing the microphone from the main circuit board helped a lot, but there was still much room from improvement.

An extra hole, roughly the size of the SD port hole, was added to the main chassis. As can be seen in Figure 20, this moved the high peak frequency range from 300...600 Hz to 700...1000 Hz, but above all it lowered signal level. Now the total signal level was +16.2 dB above the reference level. At frequencies 2800...3400 Hz it was in fact better than the reference unit in Chapter 12.3, although at lower frequencies it still was significantly worse. The signal level was now as low as it was in Chapter 12.5 where no chassis was used. This was easily low enough for the VoIP speaker phone to work as intended, although even lower levels would be preferable.

12.10 Meas +15.8 dB: Microphone Glued to Chassis

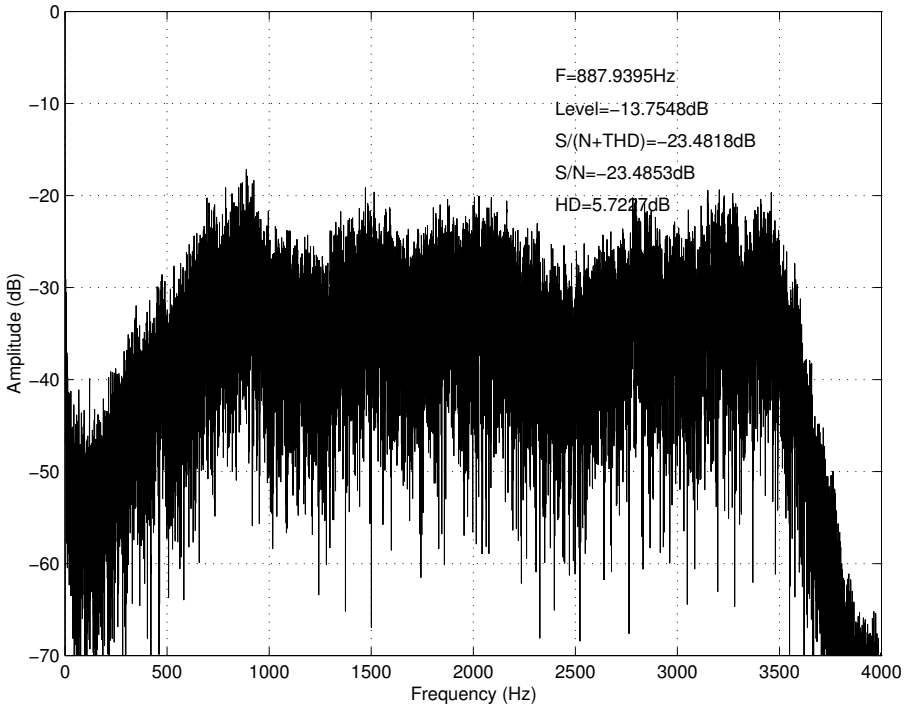

Figure 21: Case study: +15.8 dB: Microphone glued to chassis.

Another solution to separate the microphone from the vibrations of the main board and plastics was to solder the microphone to a smaller board and glue them tightly to the main chassis, as shown in Figure 22. This unit was then connected to the main board and plastics with a 15 cm thin ribbon cable.

Figure 21 shows that this gave the best results: a uniform frequency response and the lowest signal amplitude, and all of this without drilling extra holes to the chassis. Opening the chassis could potentially give even better results, but this was not tested.

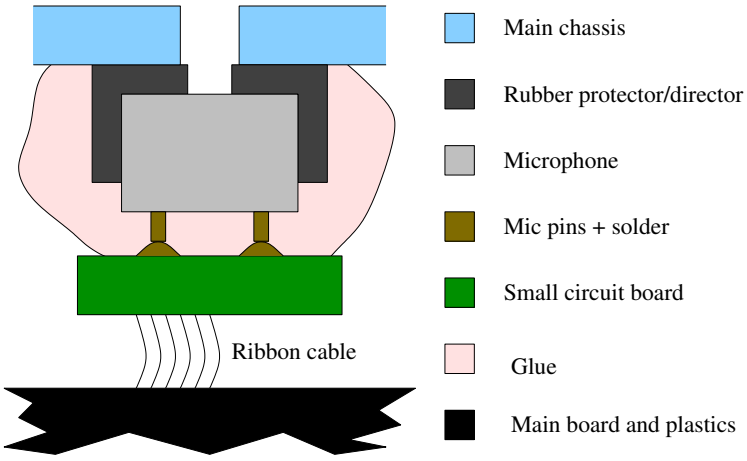

Figure 22: Case study: Microphone board glued to chassis.

12.11 Case Study Conclusions

The basic problem with the original product was that it transmitted too much audio signal from the speaker to the microphone (Figure 13 on page 24), which prevented echo cancellation from working efficiently. The main mechanisms appeared to be Equipment Vibration FES' and Internal Airspace FES' (see Chapter 5 for different types of FES echo). When these FES echo methods were addressed, the FES' signal level reaching the microphone could be lowered by a significant amount of 14.7 dB (from +30.5 dB in Chapter 12.4 to +15.8 dB in Chapter 12.10).

The research was not exhaustive, and further experiment would almost certainly result in lower FES' echo levels, which in turn would lead to even better sound quality. Still, the FES' signal was now low enough for the product to work to satisfaction.

All in all, electrical AEC is not a miracle cure: the device that intends to use echo cancellation must have good acoustic design to begin with. Only when acoustic design is good enough can echo cancellation do the thing it is designed for: to filter out different FES echo types for a good, clean audio signal.

Remember also that AEC can only filter echo if the speaker-microphone system is linear. If your speaker or device is rattling or crackling because of the speaker sound is too loud, there is nothing AEC can do to help.

13 Latest Version Changes

Version 1.21, 2014-02-20

This release of the document explains better what AEC is.

- Clarified colour scheme and terminology in Figures 2 and 4, and updated terminology in text where it depended on those images.
- Typo Corrections.

Version 1.20, 2014-02-19

This release of the document explains better what AEC is.

- Added new Chapter 5, *Introduction to What AEC Is and What It Does*.
- Corrected error in Chapter 8, *Initializing AEC*, where a write register for setting AEC Microphone (NES) Linear Gain was incorrect (read SCI_AICTRL1 when it should have read SCI_AICTRL2). Also AEC Mic Gain and AEC Speaker Gain register addresses were mixed.
- Added Figure 3 on Page 9, *AEC does not combat audio feedback between devices close to each other.*

Version 1.10, 2013-10-16

This is a major release of this document, with lots of new information.

- Added new Chapter 11, *Testing AEC*.
- Added new Chapter 12, *AEC Acoustic Design: How to Combat Acoustic Echo*.

Version 1.01, 2013-10-10

Added parametric.encoding.aecAdaptMultiplier to Chapter 8, *Initializing AEC*.

Version 1.00, 2013-10-09

Initial version.

14 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

Fax: +358-3-3140-8288
Phone: +358-3-3140-8200
Commercial e-mail: sales@vlsi.fi
URL: http://www.vlsi.fi/

For technical support or suggestions regarding this document, please participate at
http://www.vsdsp-forum.com/
For confidential technical discussions, contact
support@vlsi.fi