Dual Microphone Tuning Guide: Tuning and Diagnostics Guide

Table of Contents

1 Introduction

This document is an overview of the web based tuning panel for SoundClear Far Field (SCFF) multi-microphone voice processing software/hardware solution (CS48LV41f chip) inside the FlexConnect Module (FCM). This tool allows users to modify parameters such as gains, enable and disable processing blocks, and observe the results in real-time.

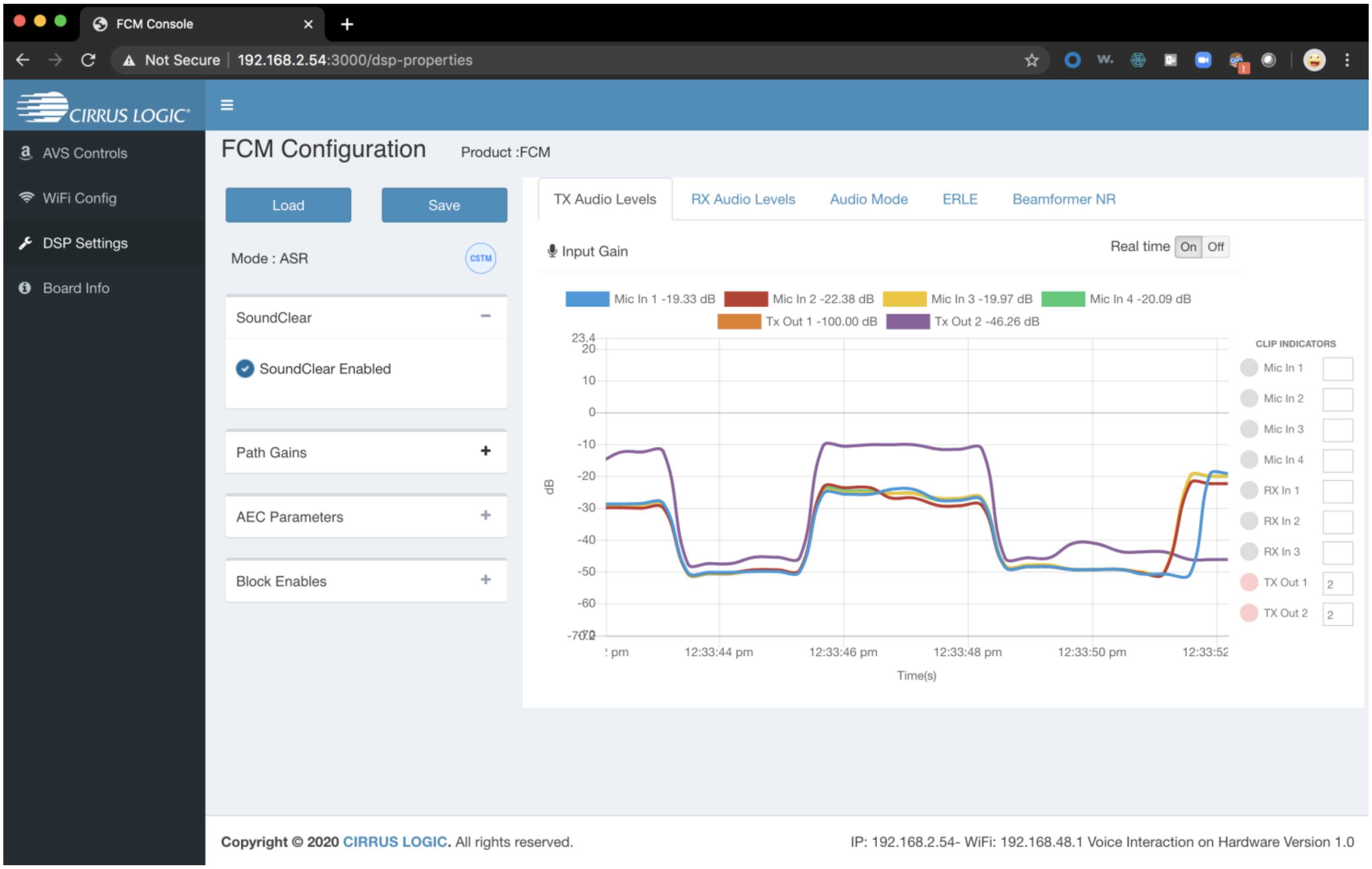
Note! This simplified web interface is developed for the FCM. For tuning SCFF to match a new hardware prototype design, users are referred to the professional SoundClear Studio application (for Windows PCs) for deep access to all registers of the CS48LV41 chip and control of all tunable parameters of the SCFF algorithm.

Additionally, this document gives example usage of a number of command-line “helper scripts”, found on the FCM filesystem under the /utils directory. The command line unlocks a deeper level of control of the CS48LV41f than is available from the simplified web panel.

2 FCM Web Console Overview

The FCM Configuration console is a dynamic web page, allowing users to observe signals and algorithm states in real-time, to download or upload tuning files (.acf) from their computer to the CS48LV48f, and to modify various algorithm states, e.g. enable or disable processing blocks, or change mic or music playout gains (volume). This web page is accessed from the “DSP Settings” item on the left-hand control panel, of the webpage that is automatically served by the FCM at the URL: http:// ip-address-of-the_FCM:3000

Figure 1: DSP Settings page of FCM Console web page: FCM Configuration



Not Secure | 192.168.2.54:3000/dsp-properties

FCM Configuration Product :FCM

Mode : ASR

☑ SoundClear Enabled

Real time On Off

Mic In 1 -19.33 dB, Mic In 2 -22.38 dB, Mic In 3 -19.97 dB, Mic In 4 -20.09 dB, Tx Out 1 -100.00 dB, Tx Out 2 -46.26 dB

Copyright © 2020 CIRRUS LOGIC. All rights reserved.

IP: 192.168.2.54- WiFi: 192.168.48.1 Voice Interaction on Hardware Version 1.0

2.1 IMPORTANT CAVEATS! READ THIS FIRST

- **The web console is a tool for simple visualization and demonstrations.** It is not the best tool for ***tuning*** the CS48LV41f to fit a new product hardware prototype, as only a limited set of parameters are presented by the simplified interface. For more complete access to tuning parameters, use one of the following mechanisms:

    - Command-line tool ***nanomix***, which can edit single parameters by name, or download complete tuning files (.acf).

    - ***SoundClear Studio (SCS)*** a Windows 10 application for dynamic control of Cirrus hw/sw solutions like the CS48LV41f, with control to the register level and visualization tools. Contact your PSP or Cirrus Logic for assistance. The FCM is SCS-ready; running a small communications process called studiobridge in the background. SCS is the primary tuning tool for Cirrus solutions, and offers controls such as a microphone geometry tuning tool, and displays (spectrograms, etc.)

- Parameters modified on the LINUX COMMAND LINE, e.g. gain values changed by direct nanomix commands, MAY NOT BE REFLECTED IN THE web console control panels. There are two masters in this example! It is recommended:

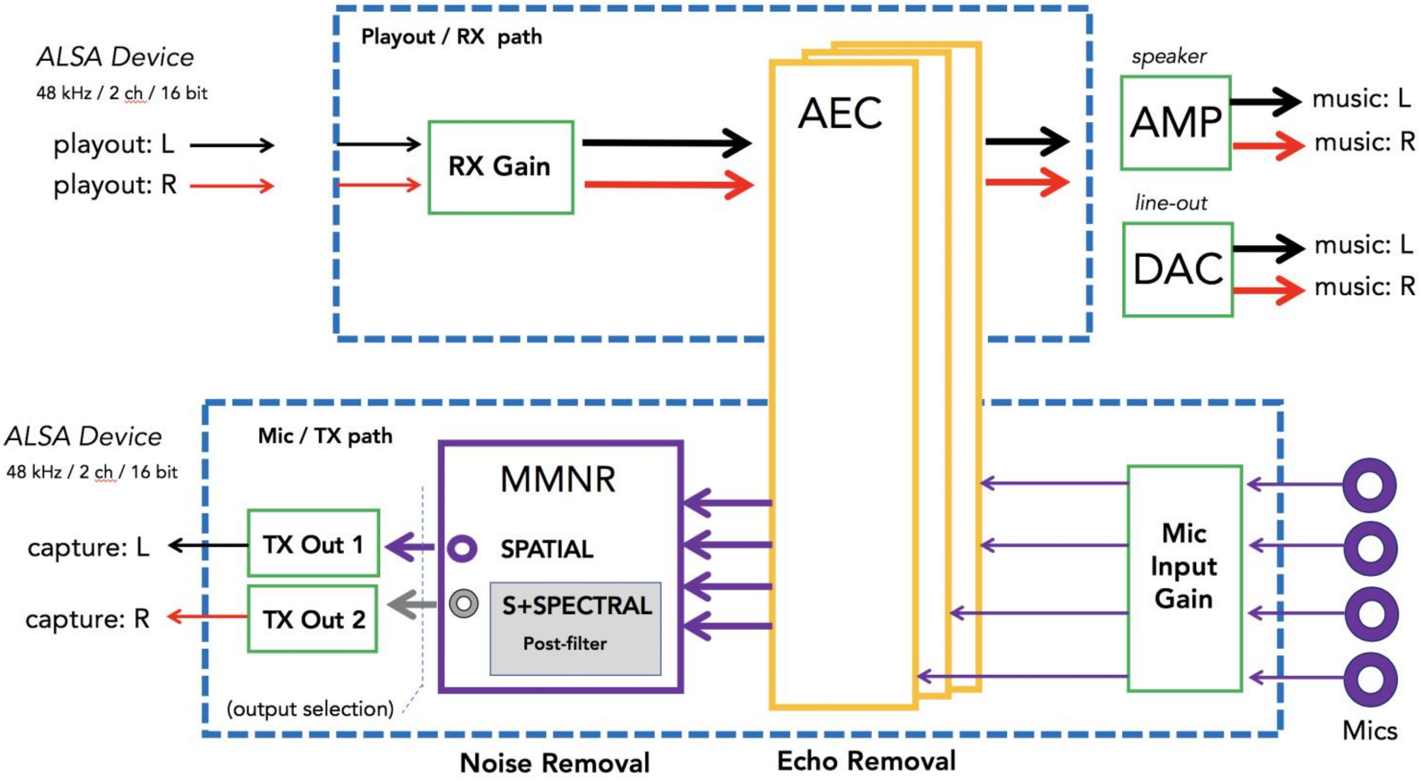
    - Use the webconsole by itself, for simple tasks OR
    - If using command line to set the state of the device, trust the webconsole as a DIAGNOSTIC DISPLAY only.

    For example, consider modifying the Microphone Array Pattern via the command-line script, while the web console is running. The mic pattern will change, but the webconsole may not be aware of new gains in effect. Treat the webconsole as a simple viewer for real-time diagnostics, in this case. The diagnostics (TX, RX levels, ERLE, Beamformer NR, etc.) will be correct as they are continuously refreshed. Be aware of which master (webconsole, command-line) spoke last!

- Having the web console open and active in a browser, even perhaps in a hidden tab, presents a load on the FCM device, and may cause intereference with tasks undertaken on the command line. For example, stop the webconsole diagnostics display (via the real-time on/off, or kill the browser tab) when downloading a new tuning (.acf) file to the FCM via the command line. In this example, we have two nanomix consumers active simultaneously, which can cause slowdowns.

3 Overview of SoundClear Far-Field (SCFF) system

Figure 2: SC Far-Field processing on CS48LV41f: Functional Model



Applications like Alexa execute on the Application Processor (AP) of the FCM, in a Linux environment. They talk to the world by reading and writing PCM bitstreams to an ALSA (Advanced Linux Sound Architecture) soundcard device, which typically comprises a playout (speaker) and a capture (microphone) device. For example, one can list the soundcards available on the FCM by entering “aplay -L” and “arecord -L” in the command-line.

The CS48LV41f chip is a hardware/software solution from Cirrus Logic that performs two main functions: echo removal and noise removal, accomplished through novel microphone array processing techniques. It takes the signals arriving at 1,2,3 or 4 microphones, removes the effect of music or speech playing out of the speakers into the room (acoustic echo cancellation, AEC) and employs an adaptive beamformer to suppress noise coming from fixed locations (like a television set) in a room, without suppressing real human talkers. This allows people to wake and interact with automatic speech recognition (ASR) engines like Alexa, from ‘far-field’ distances of up to 6m, speaking in conversational tones even when the device is playing loud music into the room.

Through linux drivers, the CS48LV41f presents itself as a stereo (2 channel), 48 kHz / 16 bit ALSA PLAYOUT device, that can be opened by any application. For example, “aplay stereotest.wav” will cause that file to be played out through the chip, and the signals will appear at the amplifier terminals and the line-out jack of the FCM simultaneously. This is the PLAYOUT path shown in Fig 2. If you ask alexa to “sing me a song”, this is the path her voice takes, from application to sound waves.

Similarly, Alexa, or applications like arecord, obtain their mic signals by opening the CS48LV41f as a stereo/48kHz/16-bit ALSA CAPTURE device. Fundamentally, the SCFF algorithm takes 4 individual mic signals and synthesizes a mono “best” mic signal to send up to listening applications. The SCFF algorithm generates two candidate outputs, however, and since the ALSA capture device is a stereo mic, with Left and Right channels, the default routing is that the left mic capture channel is the SPATIAL output signal, and the right mic channel contains the +SPECTRAL output. This output routing is configurable in the linux command line via the helper scripts.

The SPATIAL output is the output of the MMNR (mult-mic noise reduction) block, which is an adaptive beamformer. This signal represents the output of a noise reduction process based on SPATIAL characteristics of the signals arriving at the microphones. It reduces noise coming from sources, like TVs or radios, that have a fixed location in SPACE, even if their signals are voice-like.

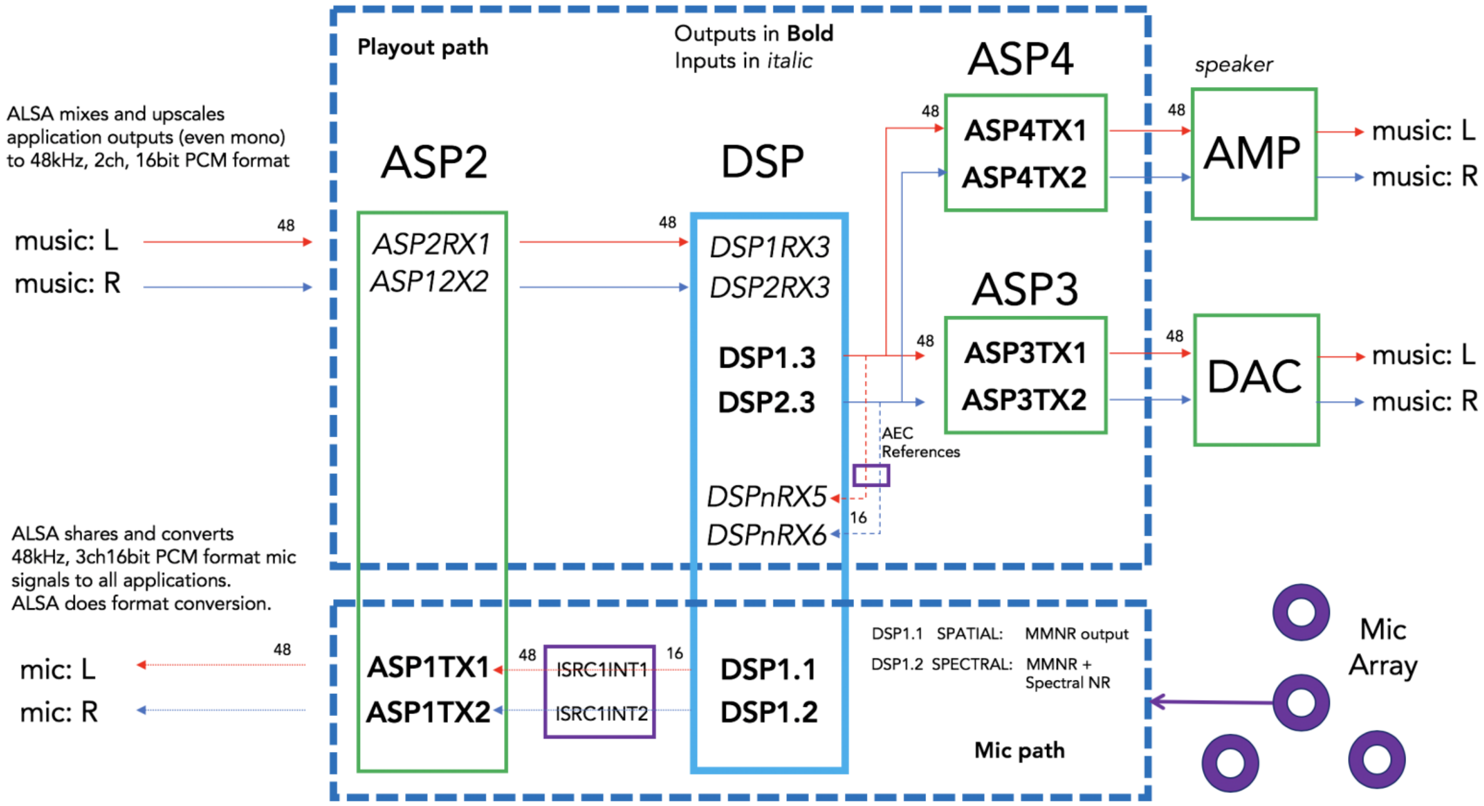
The +SPECTRAL output takes the spatial output of the MMNR block, and performs ADDITIONAL spectral subtraction noise reduction on it, in a block known as the PF or post-filter. This type of noise reduction processing is single-channel, and does not exploit spatial characteristics. It removes the “predictable” noise in the mic signal with classical signal processing techniques – the hum of appliances or the noise of fans or HVAC systems.

In general, the SPATIAL output (default left channel) mic signal is intended for use by cloud ASR systems like Alexa or Google, that is, machine listeners.

The +SPECTRAL output (default, right channel) mic signal is generally recommended for VOICE COMMUNICATIONS applications, like Zoom, Skype or Jitsi, in which the listener is a human rather than a machine. The additional depth of noise reduction is a net positive for people in telephone or video conference calls.

Note that the +SPECTRAL output signal *may* be used to feed ASR engines, since the level of spectral noise reduction is mild. It has been found to improve the performance of some wakeword engines, for example. However, aggressive spectral subtraction techniques can cause whooshing or phasing in the sound, which can degrade the performance of ASR engines which have been trained on speech without such artifacts. This is why ASR vendors discourage the use of spectral noise reduction techniques.

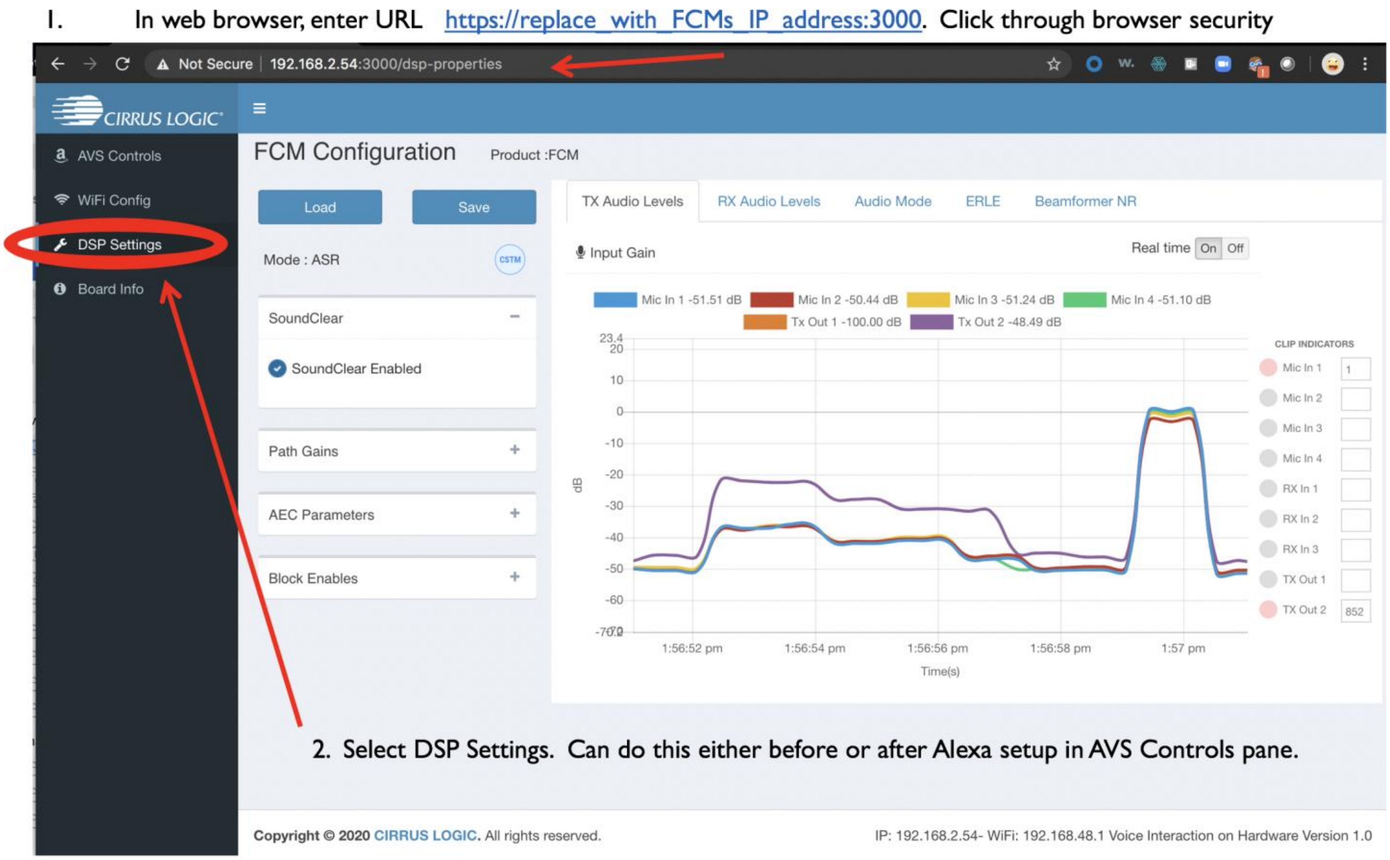
Figure 3: FCM Default Internal Routing (nanomix names)



4 Web Console – Walkthrough

4.1 Getting Started

Figure 4: Open a web browser, go to ip address of FCM:3000



1. In web browser, enter URL https://replace_with_FCMs_IP_address:3000. Click through browser security

Not Secure | 192.168.2.54:3000/dsp-properties

FCM Configuration   Product :FCM

Mode : ASR

SoundClear Enabled

Real time On Off

Mic In 1 -51.51 dB   Mic In 2 -50.44 dB   Mic In 3 -51.24 dB   Mic In 4 -51.10 dB   Tx Out 1 -100.00 dB   Tx Out 2 -48.49 dB

2. Select DSP Settings. Can do this either before or after Alexa setup in AVS Controls pane.

Copyright © 2020 CIRRUS LOGIC. All rights reserved.

IP: 192.168.2.54- WiFi: 192.168.48.1 Voice Interaction on Hardware Version 1.0

Determine the IP address of your FCM according to the bring-up procedures in the FCM manual: ethernet or WiFi. In a web browser on a computer on the same LAN, enter the FCM's IP address terminated by port 3000 (e,g. 192.168.0.2:3000) in. You will see a webpage like the above.

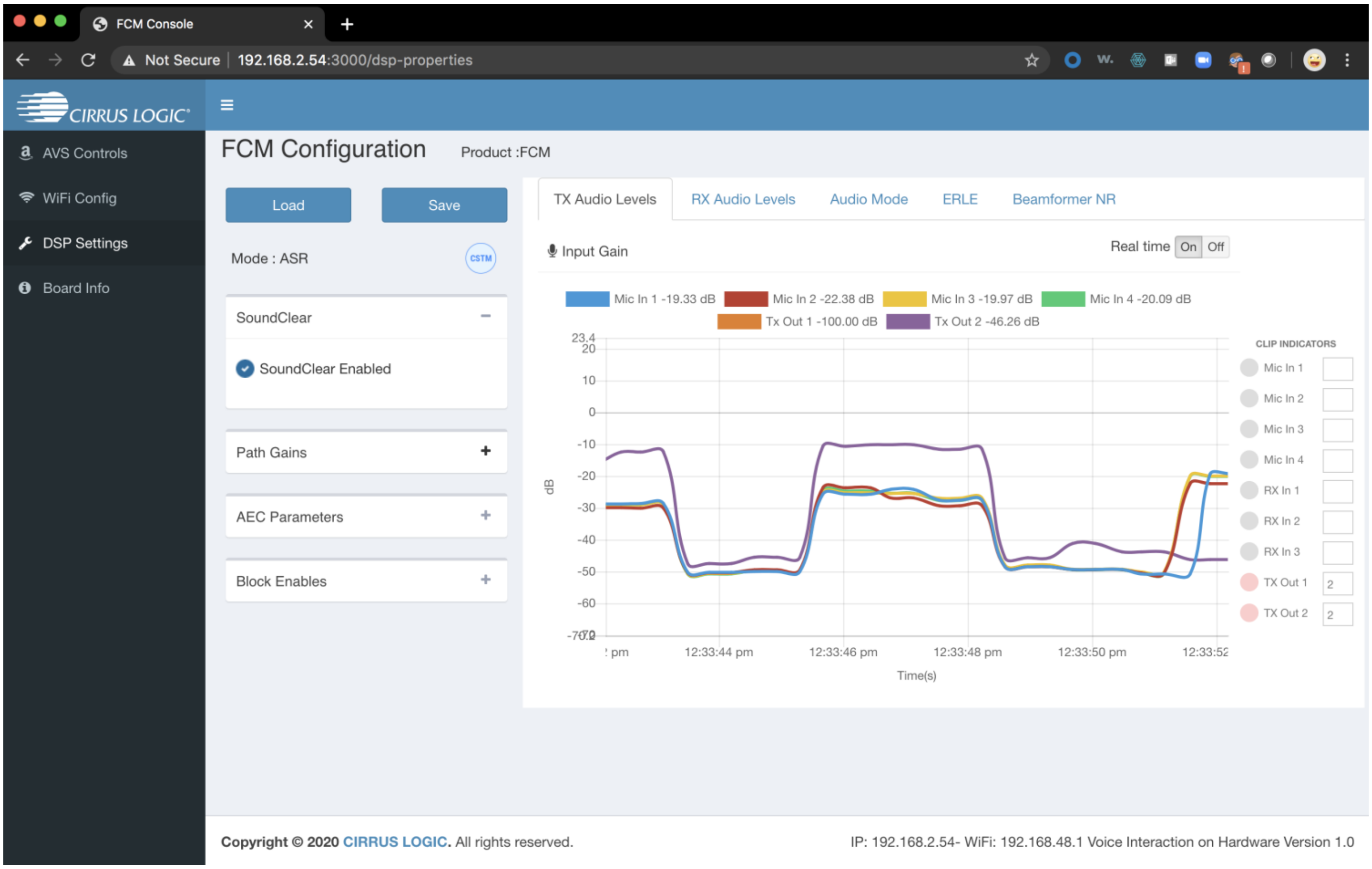
The FCM has a built-in serial-port-over-microusb jack, which is very helpful during device bring-up. If you have trouble finding the FCM's IP address, we suggest starting with a very simple network using an USB-ethernet-dongle to connect to the FCM. Connect the FCM to your PC with a microUSB-to-USB-A cable, note the COM port that appears under Device Manager, if you are running Windows. Download and run a free terminal emulator like PuTTY (Windows) or ZTerm (mac), with settings 8-N-1, 115200 baud. Login to FCM as 'root'/password 'root'. Plug in FCM ethernet and use "ifconfig" to get a listing of the active IP addresses (eth1, wlan0). When you have a useable IP address, open the web browser.

If you start talking, you should see the envelopes of your speech signal evolving in the main display, under the TX Audio Levels tab. This is the most basic sanity check of the FCM, as it proves hardware, networking and boot processes are working correctly.

If you haven't already, do register your device with Amazon using the "AVS Controls" tab at top-left. This walkthrough assumes that you are running Alexa, can hear Alexa responses in a connected loudspeaker, and can see mic signals in evolving in the web console as in the figure.

4.2 Web Console Layout

Figure 5 Web Console Layout: Load/Save on top, Control parameters on left, Diagnostics Display on right.



FCM Configuration

Product :FCM

Mode : ASR

SoundClear Enabled

Real time On Off

Mic In 1 -19.33 dB

Mic In 2 -22.38 dB

Mic In 3 -19.97 dB

Mic In 4 -20.09 dB

Tx Out 1 -100.00 dB

Tx Out 2 -46.26 dB

Copyright © 2020 CIRRUS LOGIC. All rights reserved.

IP: 192.168.2.54- WiFi: 192.168.48.1 Voice Interaction on Hardware Version 1.0

The web console has controls on the left, and a large display of real-time signals in the middle. At the top are a pair of buttons for pushing or pulling tuning files to and from the CS48LV41f chip on the FCM.

4.2.1 LOAD (write/download/push) and SAVE (read/upload/pull)

LOAD opens a window to select an.acf tuning file from your PC’s local file system and **downloads** it to the CS48LV41f.

SAVE uploads the current state of the CS48LV41f to create a new .acf tuning file on your PC.

4.2.2 ASR and VOICE mode : DYNAMIC ALEXA SWITCHING

CRITICAL NOTE

Under the LOAD/SAVE button pair, there is a small indication of the ASR/VOICE state of the SCFF algorithm. The default tunings (states) of the LV41f used in the Alexa application reside on the filesystem of the FCM:

/opt/tunings/asr.acf

/opt/tunings/voice.acf

WHILE THE ALEXA APPLICATION IS RUNNING, alexa will switch dynamically between these two tuning files. If you have manually loaded a tuning file, be mindful that using Alexa ACM features (voice calls, drop in, broadcast a message) can switch the tuning file underneath you! Avoid using ACM features, or even disable Alexa if you require tuning certainty!

FCM will boot into ASR mode by default, whether Alexa is running or not. ASR mode means the microphone signal processing is optimized for listening by an AUTOMATIC SPEECH RECOGNITION engine such as Alexa.

Voice mode a tuning optimized for human listeners. For example, it employs slightly deeper noise reduction which is helpful for humans in a voice call, but which can degrade the recognition accuracy of some ASR engines.

Voice mode is used during episodes when the user employs Alexa ACM (communications) features such as:

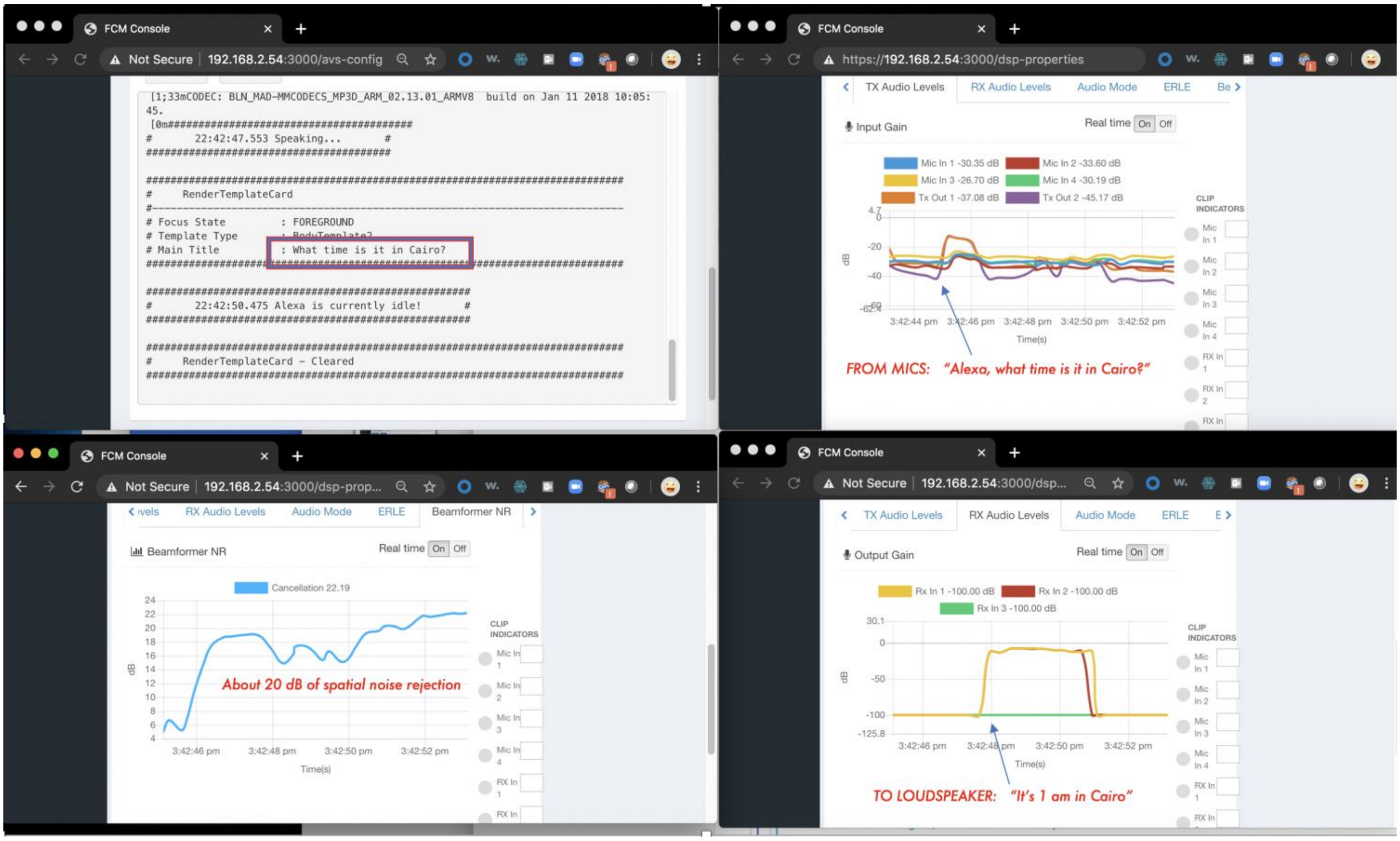
- **Voice calling** (“Alexa, call 555 1212”, “Alexa, call Susan”)
- **Drop In** (“Alexa, call the kitchen”)
- **Intercom** (“Alexa, send a message – dinner is ready”)

If you manually load a tuning file, be careful that Alexa does not clobber it. Stay in the mode you want (don’t make a voice call if you are in ASR mode; keep the voice call active if you are in voice mode). It may be simplest to merely DISABLE ALEXA if your testing or use case doesn’t require it, for example, using the FCM as a USB speaker/mic for an application like Zoom running on a PC.

As a last resort, to force Alexa to use a single tuning file in all cases, one may overwrite the two locations. /opt/tunings/asr.acf , /opt/tunings/asr.acf with copies of the desired .acf tuning. This MUST be done after every reboot, as these locations are automatically refreshed as /home/root/AVSnix/runAvs.sh executes during the start up process.

4.3 Real Time Diagnostics

Figure 6 Interaction Example with Multiple Simultaneous Diagnostics Traces



```
[1;33mCODEC: BLN_MAD-MMCODECS_MP3D_ARM_02.13.01_ARMV8  build on Jan 11 2018 10:05:
45.
 [0m########################################
#       22:42:47.553 Speaking...       #
########################################

#####################################################################################
#     RenderTemplateCard
#------------------------------------------------------------------------------------
# Focus State         : FOREGROUND
# Template Type       : BodyTemplate2
# Main Title          : What time is it in Cairo?
#####################################################################################

############################################################
#       22:42:50.475 Alexa is currently idle!       #
############################################################

#####################################################################################
#     RenderTemplateCard - Cleared
#####################################################################################
```

FROM MICS: "Alexa, what time is it in Cairo?"

About 20 dB of spatial noise rejection

TO LOUDSPEAKER: "It's 1 am in Cairo"

The main display window supports five sets of traces, accessible by clicking the tabs.

- TX Audio levels. These are the MICROPHONE levels. Talk or clap to see them move.
- RX Audio levels. These are the ALEXA OUTPUT (music, response) signals. Ask Alexa a question.
- Audio Mode. This is an AEC diagnostic showing an estimate of WHO IS TALKING.
- ERLE This is an AEC diagnostic showing depth of echo reduction (in dB).
- Beamformer NR This is a multimic noise reduction (MMNR) diagnostic, showing depth of NR (in dB).

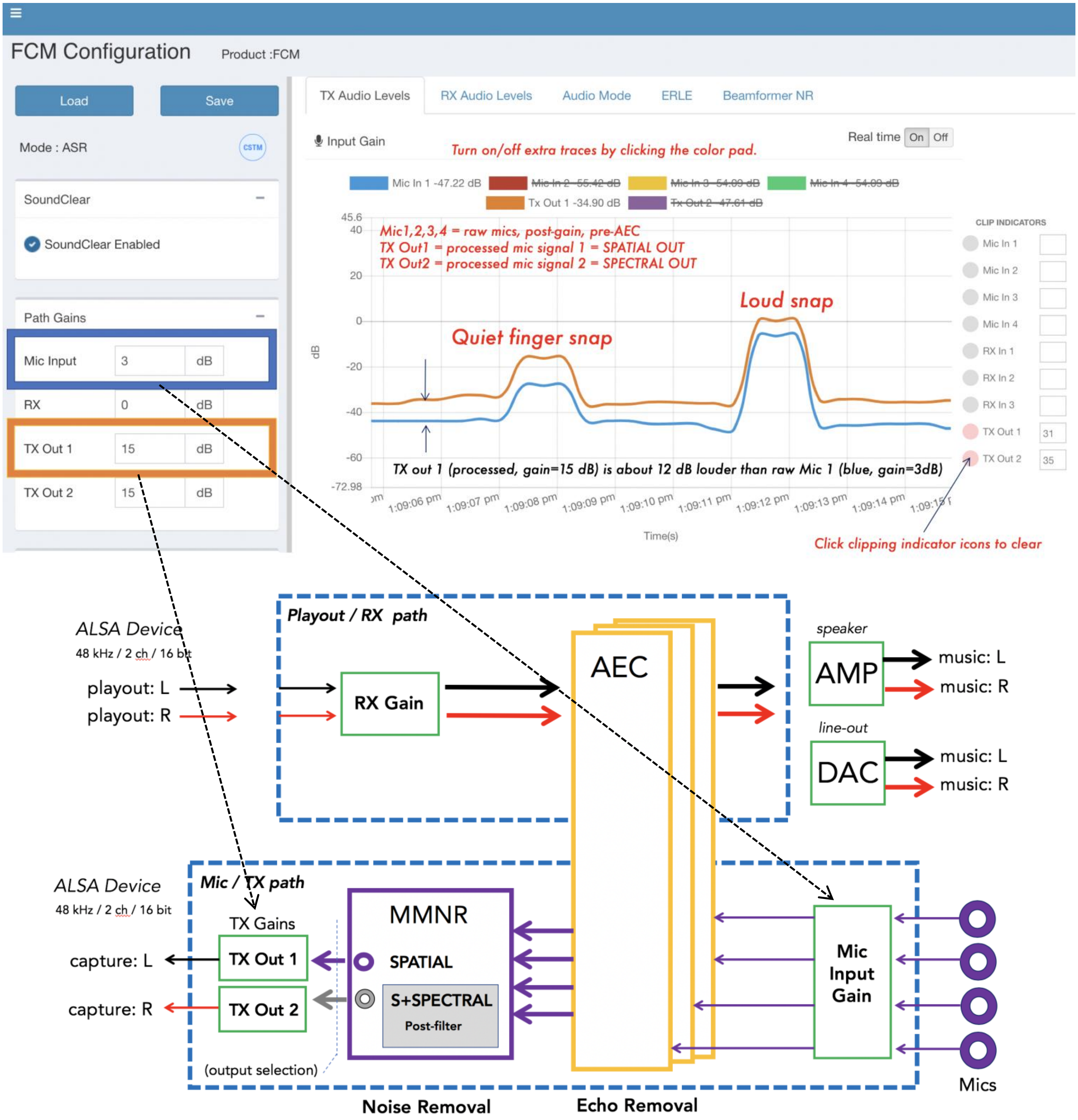
The figure shows a simple Alexa interaction. The user says "What time is it in Cairo?", visible in the TX audio levels window. We see the command correctly interpreted by Alexa, in the AVS window. Alexa's spoken response is visible in the RX audio levels display, a few seconds after the question was uttered. During this capture a television set started playing in the backgound, the MMNR algorithm quickly adapted to this noise source, and we see about 20 dB of spatial noise rejection during the interaction. Without MMNR, Alexa might not have understood the question!

All five diagnostics windows have a real-time ON/OFF radio button, that can be used to pause the display. This is useful for screenshots. It's also a good idea to pause diagnostics or kill unused tabs if you are working on the FCM command line. Diagnostics over http can interfere with nanomix script execution on the command-line.

4.3.1 TX Audio Levels and Gain Controls: Microphone Signals: Raw and Processed

TX refers to the TRANSMIT or MIC direction, from the physical microphones on the FCM, captured as four raw mic signals (mic1,2,3,4), through AEC and array processing, up to the Applications processor and made available to Linux applications as an ALSA microphone or capture device.

Figures 7 TX Audio Levels display and Reference SCFF Block Diagram



FCM Configuration Product :FCM

Mode : ASR

SoundClear

☑ SoundClear Enabled

Path Gains

Mic Input	3	dB
RX	0	dB
TX Out 1	15	dB
TX Out 2	15	dB

Input Gain — Real time On Off

Turn on/off extra traces by clicking the color pad.

🟦 Mic In 1 -47.22 dB 🟥 ~~Mic In 2 -55.42 dB~~ 🟨 ~~Mic In 3 -54.09 dB~~ 🟩 ~~Mic In 4 -54.09 dB~~ 🟧 Tx Out 1 -34.90 dB 🟪 ~~Tx Out 2 -47.61 dB~~

Mic1,2,3,4 = raw mics, post-gain, pre-AEC
TX Out1 = processed mic signal 1 = SPATIAL OUT
TX Out2 = processed mic signal 2 = SPECTRAL OUT

Quiet finger snap

Loud snap

TX out 1 (processed, gain=15 dB) is about 12 dB louder than raw Mic 1 (blue, gain=3dB)

Click clipping indicator icons to clear

The TX Audio Levels diagnostics panel is shown in Figure 5 above. There are 6 traces, each of which can be added or removed from the display (to reduce clutter) by clicking the associated colored pad in the legend. The raw input mic signals are available, as mic1,2,3,4. These signals are captured at a point AFTER "Mic Input Gain" is applied, but before AEC is applied to the mic signal. Refer to the green boxes in figure 6, which represent the gain blocks set by the Path Gains control box in the webpage. Try increasing the MIC INPUT GAIN in the Path Gains control box at left, and observe that this gain is applied to all four mics.

The other two traces, TX Out 1 and TX Out 2 are two PROCESSED microphone output signals. These are the "cleaned up" microphone signals sent up to the application via the stereo ALSA microphone device, in the LEFT and RIGHT channels of the mic device, respectively. Refer back to Section 3 for details about the meaning of the SPATIAL and +SPECTRAL outputs, and instructions on how to configure mic output selection in the linux command-line. The diagram shows the DEFAULT routing.

Let's walk through the example of Figure 4. Try it yourself. Open the TX Audio display, and the Path Gains control panel. Speak into the FCM and watch the mic traces follow the contour of your speech – the energy envelope.

Your room will probably be quiet, and if no one is talking, a typical level will be about -50dB. If you yell into the mics, they will clip, and exceed 0dB. There are clipping indicators to the right of the display. They will turn red if clipping occurs, and can be reset by clicking on them.

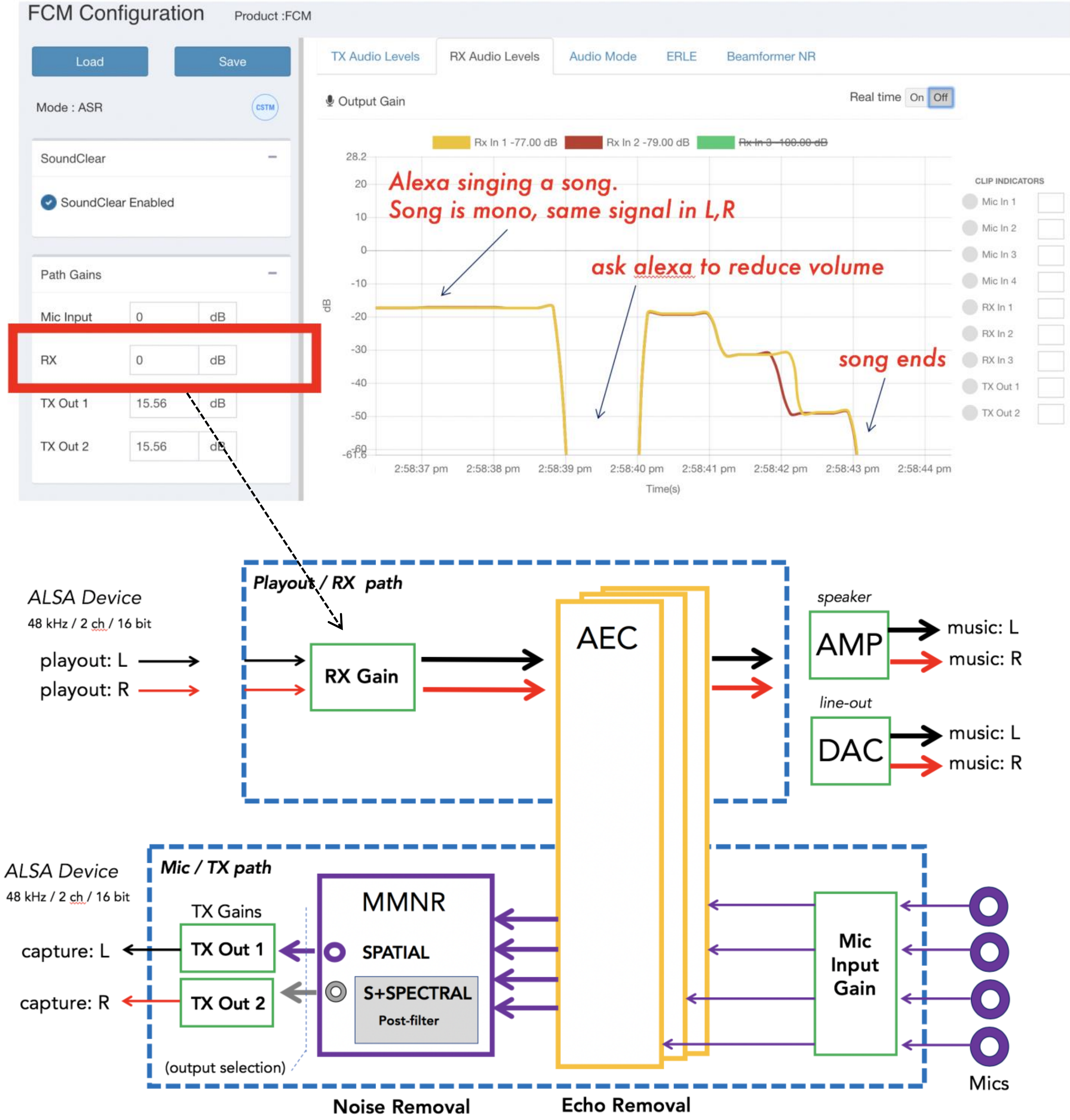
De-clutter the display by tuning individual traces on and off, clicking on their color pad in the legend at top of the TX Audio display. Pick one microphone. In the Path Gains control panel, add 10 dB or -10 dB of Mic Input gain, and see that the mic level jumps up or down as expected. Note that this display shows the raw mics AFTER the application of input gain. The FCM uses digital PDM microphones, so the input gain applied by the CS48LV41f is digital.

Similarly, the levels of the TX Out 1 and TX Out 2 output mic signals can be manipulated by changing the gain applied to them in the control box at left. This affects how loud the mic signals appear to the application that is reading them, such as Alexa.

4.3.2 RX Audio Levels and Gain Controls: Application Playout, Music and Responses

RX refers to the PLAYOUT direction: Music and speech from apps running in Linux on the Applications Processor, passing through the CS48LV41f and out to the real word via lineout DAC and amplifier outputs. It is critical that playout signals flow through the chip, as they provide AEC REFERENCE signals. A major feature of the CS48LV41f is the performance of the Acoustic Echo Canceller, which monitors the signals playing out into the room, forms a functional model of the room using an adaptive filter, estimates the echo signal and cancels it from the returning mic signals.

Figure 8 RX Audio Levels – Music and Responses from Alexa, and SCFF Block Diagram



FCM Configuration Product :FCM

Mode : ASR

SoundClear

☑ SoundClear Enabled

Path Gains

Mic Input	0	dB
RX	0	dB
TX Out 1	15.56	dB
TX Out 2	15.56	dB

Output Gain

Real time On [✓ Off]

🟨 Rx In 1 -77.00 dB 🟥 Rx In 2 -79.00 dB 🟩 ~~Rx In 3 -100.00 dB~~

Alexa singing a song.
Song is mono, same signal in L,R

ask alexa to reduce volume

song ends

The RX Audio Levels diagnostics panel is shown in Figure 8 above. There are 3 traces, each of which can be added or removed from the display (to reduce clutter) by clicking the associated colored pad in the legend. In the example, the third rx channel (green) has been removed. Recall the FCM v1.0 software presents the CS48LV41f as a STEREO, 48kHz/16bit soundcard. Hence only two channels are used – RX In 1 = left channel, RX In 2 = right channel of soundcard.

Refer to the green boxes in figure 8, which represent the gain blocks set by the Path Gains control box in the webpage. There is a single RX GAIN box, which applies the same value to both incoming RX channels.

Try increasing the RX GAIN in the Path Gains control box at left, while alexa is singing a song. You will immediately hear the effect, as you are effectively increasing the music playout level through the chip and out towards the DAC. Try a negative value to reduce the volume. If you choose a large, positive value, the playout level may become loud and distorted, and the clipping indicators may trigger. These can be cleared by clicking on the circular clipping icon.

The above test is just to show that the RX Gain control works, and has an effect on the RX In 1,2 energy traces.

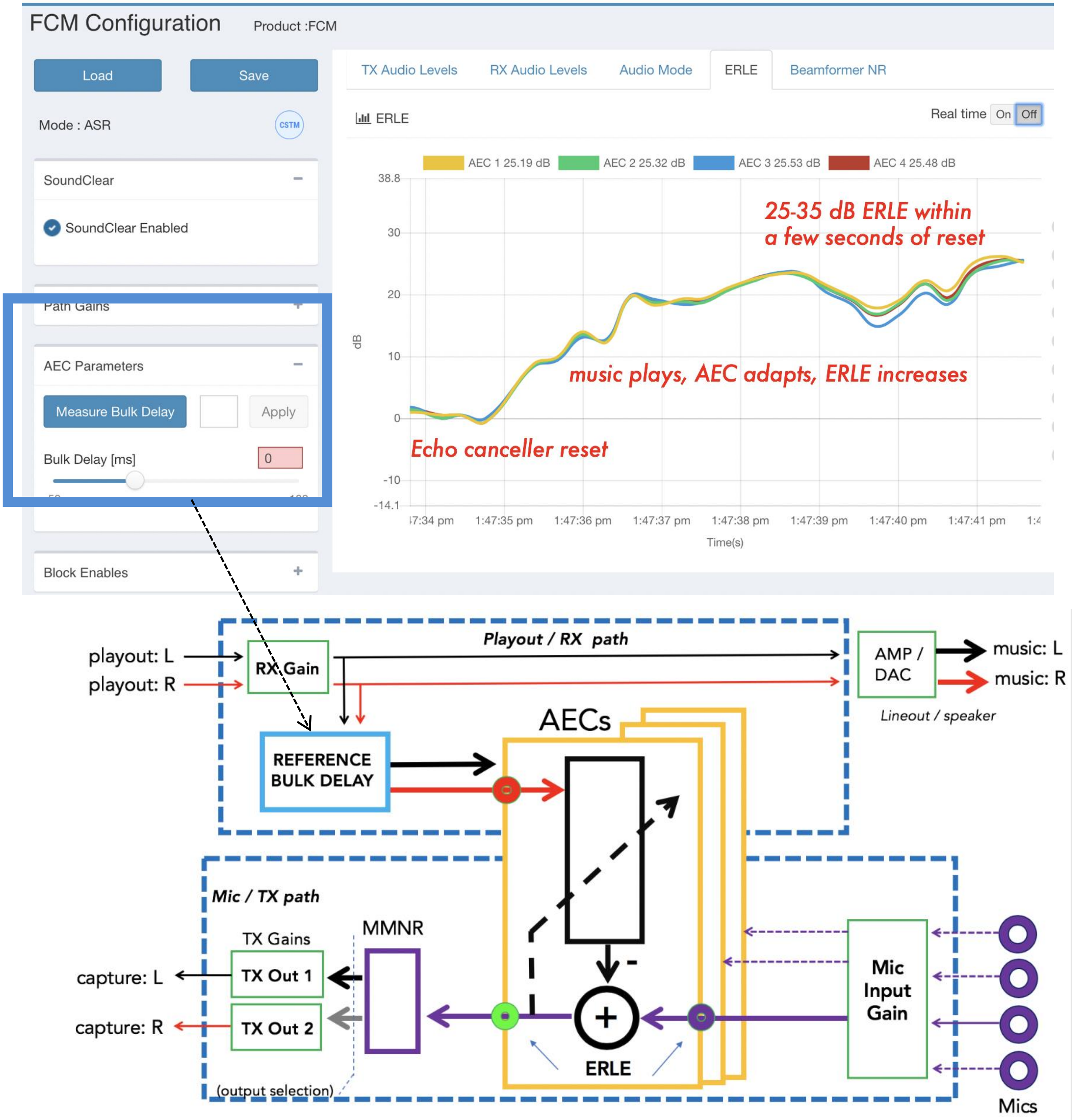
The default RX Gain value is 0 dB, i.e. unity gain, or zero amplification. Note that this is gain applied to the signal coming OUT of the music playing application (like Alexa or Spotify). These applications typically have their own volume controls and are capable of playing full-scale digital signals. The RX Gain controls the level seen by the SCFF application. It is NOT the primary control for how loud the music will be in the real world however.

In tuning a prototype design, it is recommended to:

- Keep RX Gain (playout gain) at 0dB. Keep all digital ASP gains = 32 = unity gain.
- Play the loudest expectable signal (e.g. full-scale sine wave) from the application in Linux.
- Choose amplifier settings (analog or digital) in your design that reproduce this loudest signal with acceptable compromise among: distortion, loudness, and speaker-to-mic coupling.
- You are done. Let the application set the output volume. This way, volume changes are seen by the CS48LV41f and the echo reference signal tracks the volume correctly.

4.3.3 Acoustic Echo Canceller (AEC): ERLE and Voice Mode

Figure 9 AEC Parameters and ERLE diagnostics traces



FCM Configuration   Product :FCM

Load   Save

Mode : ASR

SoundClear

☑ SoundClear Enabled

Path Gains

AEC Parameters

Measure Bulk Delay   Apply

Bulk Delay [ms]   0

Block Enables

TX Audio Levels | RX Audio Levels | Audio Mode | ERLE | Beamformer NR

ERLE   Real time On Off

ERLE = echo reduction from AEC mic in ● to AEC mic out ● nodes

4.3.4 Acoustic Echo Canceller (AEC): Interpretation of ERLE

Figure 9 shows the Acoustic Echo Canceller (AEC) parameters control panel at left, a live ERLE diagnostic in the main window, and a functional diagram of the CS48LV41f processing paths highlighting a single AEC instance, for clarity.

The AEC block in yellow has two input nodes and one output. The red input node at top left is the Rin or AEC REFERENCE input, the incoming music or speech signal that is being played out to the world over loudspeakers. The purple input node at bottom right, Sin, is the raw microphone signal (post gain) that contains both undesired echo (music we are playing out) and desired user speech. The green node at bottom left, Sout, is the output of the AEC. It is the “cleaned up” microphone signal, from which the echos have been stripped by action of the AEC. There are multiple AEC blocks – one stereo block for each microphone, in fact. This is why there is an individual ERLE trace for each active microphone in the diagnostics window, marked AEC1,2,3,4.

ERLE, or Echo Return Loss Enhancement, in dB, is an instantaneous measure of the “effect” or echo reduction obtained by the action of the AEC. Higher is (generally) better. Here is how to interpret this metric:

Imagine that Alexa is playing some stereo music from a streaming radio station. Loud music enters from the top left, and is directly routed to the amplifier and DAC for playout over speakers, with a minimum of delay.

The microphones in the device will naturally pick up all the sound in the room – including desired signals such as a talker asking Alexa to pause the music, and undesired signals such as TVs, fans, and most importantly, the loud music that is playing out into the room! That music is playing out of loudspeakers that are probably located within inches of the microphones! Additionally, the sound bounces all over the room, off the walls and ceilings and objects, and these reflections are also picked up by the microphones.

The role of the AEC is to strip away (as far as possible) the music from the mic signals, removing the effect of the music we ourselves are pumping into the room. This is echo removal or reduction (refer back to Figure 2). Post-AEC processing, these cleaned-up mic signals are subject to further noise reduction (Fig.2) in the MMNR (multi-mic noise reduction) block.

An echo canceller works by comparing the outgoing signal (red node) with the incoming mic signal (purple node). Over time, the AEC forms an internal model of the output-to-input relationship. This internal model is used to synthesize a real-time estimate of the echo signal, and estimate is subtracted from the incoming raw mic (purple node) to reveal a hopefully “echo-free” version of the mic signal, to form the output (green node).

ERLE is hence a measure of the “improvement” or “amount of echo energy subtracted” from the raw mic. It is a measure of how well the echo canceller is doing at its job, expressed as an energy ratio (in dB). From an empty initial state, the AEC will achieve zero ERLE, or no improvement. Look at the example in Fig 9. Over time (typically a few seconds) the AEC will converge and become a better estimator of the echo, and this results in the ERLE growing over time, while the music is playing. Depending on the nature of the echoes and the volumes, the AEC will typically top out between 20 and 35 dB of ERLE.

The graph shows instantaneous ERLE, and will hence move up and down depending on the energy present in the loudspeaker and mic signals. Looking at Figure 9, one can interpret that the “true” ERLE obtained by the AEC was a smooth upwards curve from zero to about 25 dB within about 3 seconds, then a slower, flatter increase up to about 28 dB over the next four seconds. ERLE is the “improvement”. It’s not always better to have high ERLE. It would be great to have low echo in the first place (known as high ERL or Echo Return Loss) in the room, which would result in zero ERLE! Having high ERLE means the AEC is working well, but it also means it is being forced to work hard in the first place. It’s a good sanity metric, but the best metric for echo is that the residual signal transmitted to the far end (cloud or human listener) is very small, when the near-end user is not talking.

4.3.5 Acoustic Echo Canceller (AEC): Control of Reference or Bulk Delay

The control panel at left of Figure 9, "AEC Parameters" has a slider and text-entry box for manually setting the AEC REFERENCE BULK DELAY compensation (ms), and a button for performing an active measurement (loudspeaker must be connected) of the actual loudspeaker-to-mic loop delay. Recall the job of the AEC is to remove leakage of playout (music) from the loudspeaker into the mics. It takes a small amount of time for signals to get from the loudspeaker to the mic, the minimum echo path, or "bulk delay". It is a waste of DSP resources to compute echo models in a region (time < minimum echo delay) where there cannot ever be echoes! The bulk delay control can be considered a one-time initialization parameter for an AEC, to tune it to known latencies in the product design

CRITICAL: Note that manually adjusting the bulk delay will reset the AEC.

CRITICAL: The FCM platform has a default setting of 0ms. This value does NOT need to be changed.

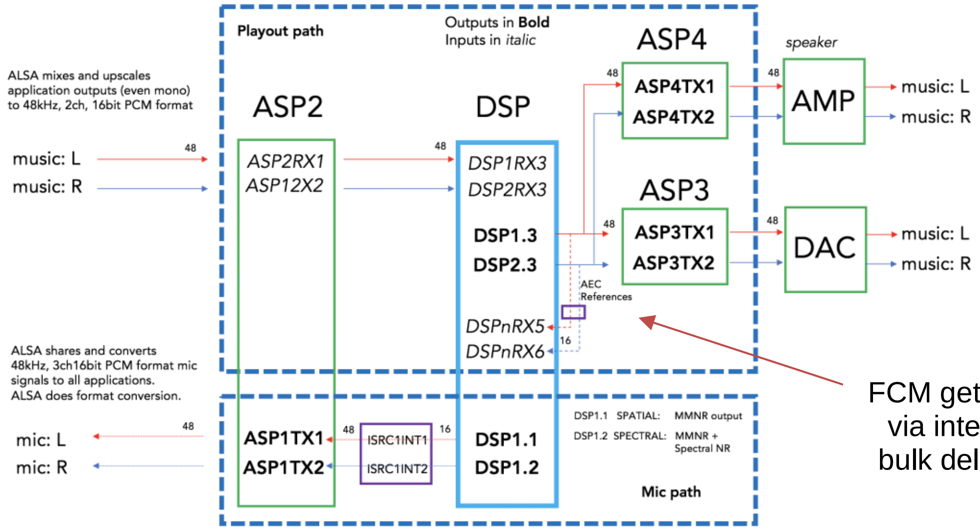
The REFERENCE input to the AEC is a local copy of the signal being played out into the world, which is then picked up by the mics in both direct form (the mic hears the music) and in the form of innumerable reflections arriving at the mics after the music bounces off ceilings, floor, walls, windows and other objects in the room.

It is this direct path of audio transmission we are interested in. Imagine that the music signal is a recording of a balloon popping, a short, sharp sound. The instant that pop in the playout signal passes through the RX GAIN square in the block diagram of Figure 9, an imaginary stopwatch is started. It will take a certain number of milliseconds for that 'pop' to continue through the playout path, undergo some additional signal processing, be accumulated as samples into a frame and be transmitted over the fast I2S bus to the DAC or digital amplifier, and eventually emerge in the real world as a physical 'pop' vibration in the air. Further time will elapse as the 'pop' sound propagates though the air from the loudspeaker to the microphones, which are some distance away, as sound travels at about 341 m/s. Similarly, the digital audio signal from the PDM microphones will take a finite time to pass up the mix/tx path before reaching the purple input node of the AEC block. In sum, there is a time delay between output and input, and no echo could ever arrive at the AEC before this minimum delay. This is known as the BULK DELAY of the tail circuit, which is largely determined by the digital audio output and input paths of the device, beyond the CS48LV41f.

Follow the arrow in Fig 9 from this control to the bulk delay compensation block in the AEC input path. This element copies the stereo playout signal headed for the loudspeaker, stores it for the desired number of milliseconds (up to 100ms) then replays it out to the AEC block. This has the effect of pulling the first echo seen by the AEC back towards time t=0.

For the ***FCM in particular, the default bulk delay setting is zero ms as the AEC reference signal is obtained by an internal routing***, seen in Figure 3 (reproduced again here as Figure 10)

Figure 10: FCM Default Internal Routing (nanomix names)



FCM gets AEC references via internal path, hence bulk delay compensation is zero

Because the output to input minimum delay is very short in this FCM routing, the actual BULK DELAY is very close to zero, about 0.5 ms in fact.

However, as one moves the mics away from the loudspeaker, or more likely, incorporates the CS48LV41f in a product design in which the loudspeaker-to-mic minimum delay is large (e.g. a wireless RF loudspeaker in a home theater product), the value of a controllable bulk delay becomes clear.

One wishes to ***underestimate*** the bulk delay compensation by a small amount (3-5ms), so that the AEC does not miss the actual echo onset. Overestimating the bulk delay causes late arriving reflections to be missed, not catastrophic, but this reduce the depth of echo reduction. It is easy to notice a “too short” or “too negative” bulk delay value, as the ERLE will be sharply degraded (drop off) as one moves the delay past the actual loop delay, in the negative direction, and restored if moved back across that point. For example, if the actual loop delay were 10ms, setting the bulk delay compensation = 0 ms (underestimate) would have little apparent effect, but setting it to 22ms (overestimate) would cause an immediate sharp drop in ERLE.

In sum, for the FCM platform, the AEC references are tapped internally (Fig.3), and the default bulk delay compensation is zero. This parameter does not need to change, unless the circuit is placed in a design in which:

- AEC references are delivered externally, e.g. via external input ASP2. (This is the case in which it is possible to feed the loudspeakers before the AEC, and a NEGATIVE bulk delay value is required. This is achieved by delaying the MIC INPUT signal, rather than the AEC Reference).
- A long delay is inserted in the tail circuit.

For these non-FCM use cases, the reader is referred to the CS48LV41f tuning guide for guidance. This would be a good place to remind designers that the tail circuit seen by the echo canceller (everything between ASP3/4 output and the transducers) MUST BE LINEAR, TIME-INVARIANT.

This is standard AEC theory. Elements like simple EQ, pre- or de-emphasis, simple transmissions delay and crossovers are linear processes, and are perfectly acceptable. However, the use of non-linear devices like DYNAMIC RANGE COMPRESSORS, CURRENT LIMITERS, decision elements, etc. will degrade the operation of a linear adaptive filter like an AEC. It is perfectly acceptable to use these, but in that case the AEC REFERENCE SIGNAL must be taken as close to the actual transducers as possible. The AEC reference must be the same as the signal going into the air. For this reason, the CS48LV41f has multiple ASP ports, to allow external AEC references to be routed into the chip.

It can be an effective demo to listen to the mic output (tx out 1) with the AEC alternately turned ON and OFF via the AEC Enable switch in the BLOCK ENABLES control panel in in the left side of the web console. This is most effective in realtime, monitoring the FCM mic signal on a macbook via USB. Another option is to record the mic signal(s) on the FCM itself, via the command-line with the “arecord” utility. An example usage is “arecord -f S16_LE -c 2 -r 48000 mictest.wav”. You can use the complementary “aplay” utility to play out a stereo music file, in a second terminal, at the same time, e.g. “aplay stereo_music.wav”. Alternatively, to generate loud music that you can use to test barge-in one can ask alexa to play a radio station, or count to one hundred. Barge-in refers to the ability to interrupt Alexa

4.3.6 Acoustic Echo Canceller (AEC): Audio Mode (Single-talk, Double-Talk) Classifier

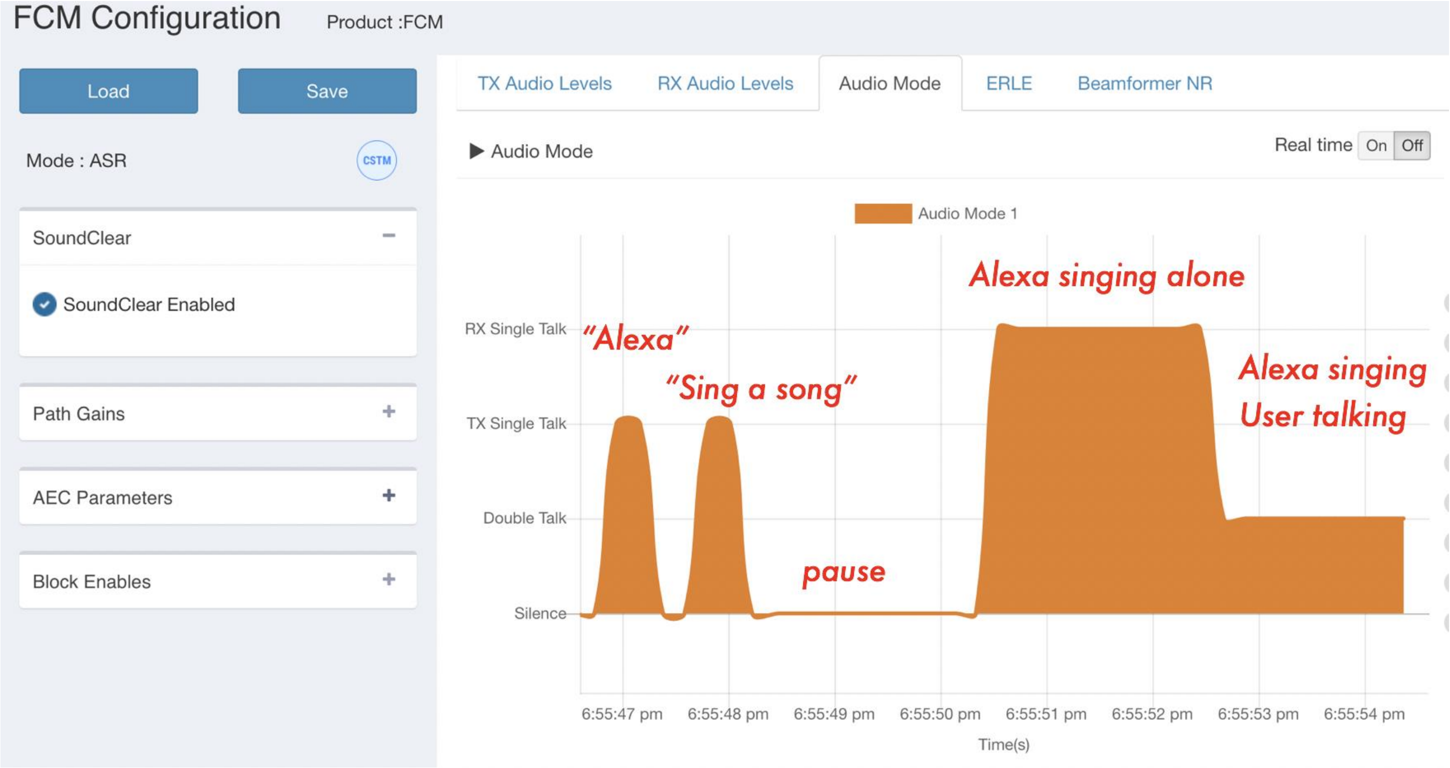
This diagnostic shows internal states of the echo canceller's "speech classifier". See Figure 2 for definitions: RX refers to the (music) playout path, towards the loudspeaker. TX refers to the microphone path, from the user towards listening applications.

RX Single Talk - Alexa is talking; Near-end, local user is silent.
- Mic signal is mostly/all undesired echo.
- AEC adapts quickly during this phase.

TX Single Talk - Near-end, local User is talking; Alexa is silent.
- Mic signal is mostly/all desired User speech, not echos.
- AEC adaptation is frozen during this phase.

Double Talk - Both Alexa and near-end User are talking simultaneously.
- Mic signal is mix of desired User speech and unwanted echo.
- AEC adaptation is frozen during this phase.

Figure 11 : AEC "Audio Mode" Diagnostics Display



FCM Configuration

Product :FCM

Mode : ASR

SoundClear Enabled

"Alexa"

"Sing a song"

pause

Alexa singing alone

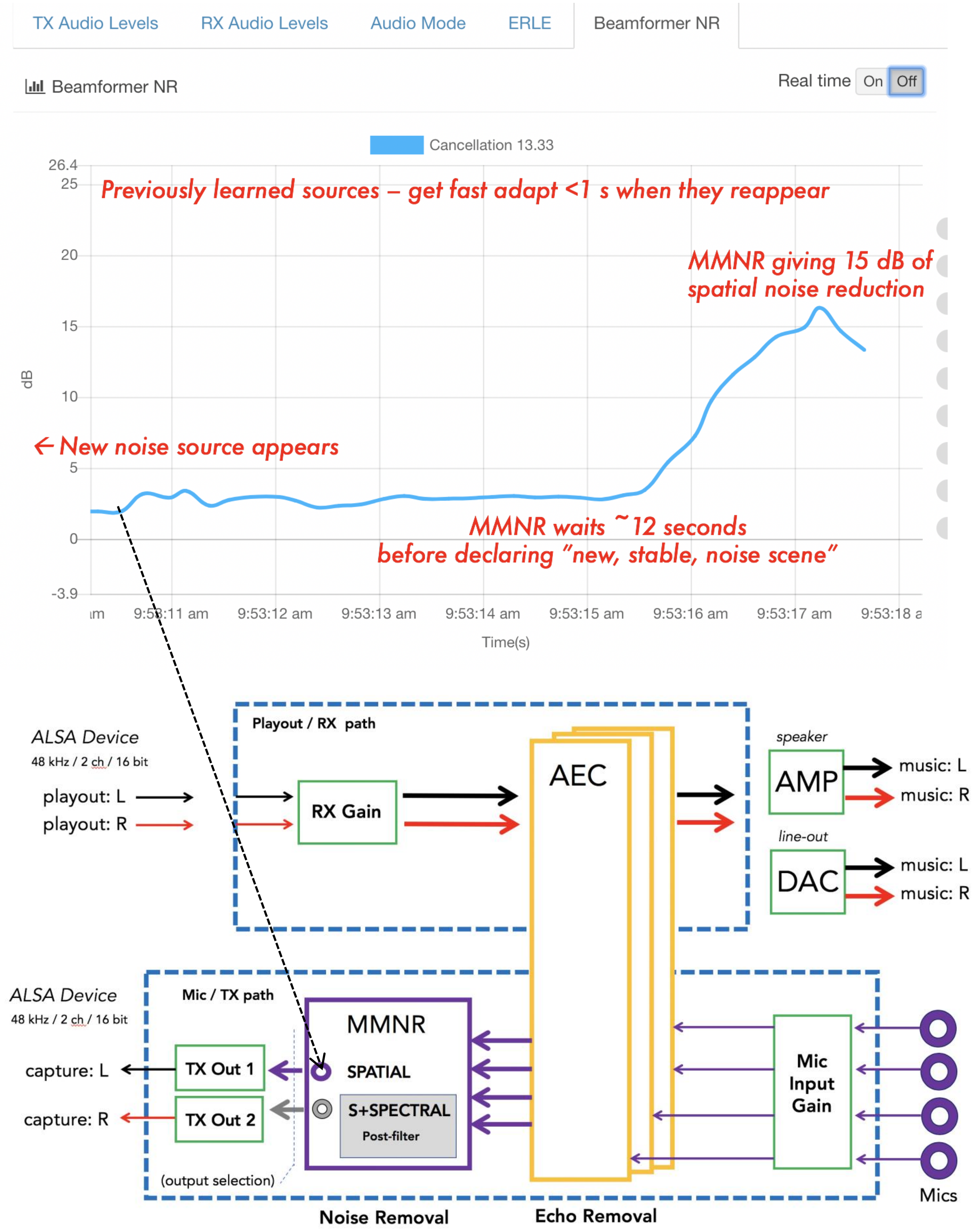
Alexa singing
User talking

Notes on AEC Diagnostics: ERLE, Bulk Delay, Audio Mode

- During Alexa playback, the AECs cancel echo — typically 25+ dB
- ERLE varies due to differences in the level of the music or speech playback
- During double talk, that is, when Alexa is playing music and the near-end talker speaks, the measured ERLE will drop. The AEC is still working, it is just that the instantaneous energy difference is lower.
- AECs do not adapt when there is no playout/RX signal (when Alexa is silent).
- Loudspeaker distortion will lower ERLE performance.
- Microphones clipping (too much mic gain) will cause drops in ERLE similar in nature to the way double talk changes ERLE
- bulk delays that are too long/too positive will cause the AECs to NEVER converge.
- bulk delays that are too short/too negative will cause slow convergence and sub-optimal ERLE

4.3.7 MMNR: MultiMic NR / Beamforming NR Diagnotics Display

Figure 12 : MultiMic NR (MMNR, “Beamformer NR”) Diagnostics Display



Previously learned sources – get fast adapt <1 s when they reappear

MMNR giving 15 dB of spatial noise reduction

← New noise source appears

MMNR waits ~12 seconds before declaring "new, stable, noise scene"

The MMNR diagnostics panel is shown in Figure 12 above. This is an instantaneous measure of the depth of noise removal that is being achieved by the MMNR block, which is an adaptive beamformer designed to identify sources of noise in a room, such as televisions or air ducts, which are stable in their position relative to the device. It is a measure of how much noise is being removed from the SPATIAL output of the MMNR block. Recall that the +SPECTRAL output of the MMNR block uses the spatial output as its own source, so the +SPECTRAL output will also exhibit at least this much noise reduction -- typically a further 6 dB.

It is a challenge for traditional noise reduction techniques to suppress dynamic, impulsive and speech-like noises. It is hard to suppress speech coming from a TV, for example, but not interfere with Alexa commands from real human users. The MMNR beamformer accomplishes this by forming a picture of the noise environment in a room, identifying the directions of loudest stable noise sources, and diminishing the input from those directions. It's a form of selective deafness to known noise sources.

Figure 12 shows the behaviour of the MMNR when presented with a new noise source. There is a waiting period of about 12 seconds, and then the MMNR adapts quickly to give about 15 dB of suppression in this example. The MMNR has memory, so that a previously learned noise source (TV, HVAC duct, radio, etc.) will be suppressed very quickly when it starts up again, without the 12s waiting period. The reason for this "new source" hold off behavior is so we don't accidentally lock on to a real human and treat him as a noise source! MMNR distinguishes real humans from televisions, for example, by that fact that real humans move about. Even if sitting down, your head is always moving to a small degree.

Higher noise reduction performance (in dB) is generally better: It indicates the algorithm is doing a great job at removing noise from the mic signals, and making life easier for listeners in the cloud like Alexa, or human listeners at the other end of a voice/video call. Recognition accuracy and ease of understanding are greatly enhanced by the action of the MMNR giving significant amounts of NR. Like the ERLE example, however, it is a mixed blessing, as higher values indicate that the environment is increasingly hostile in the first place, with noisy interference sources, or loud echoes, respectively.

It can be an effective demo to listen to the spatial mic output (tx out 1) with the MMNR alternately turned ON and OFF via the MMNR Enable switch in the BLOCK ENABLES control panel in in the left side of the web console.

5 Helper Utilities – Linux Command Line

The simplified web console can't give the detailed control of the CS48LV41f offered by the Windows 10 SoundClear Studio application (read/write control of every register in the chip datasheet). However, the linux drivers written for the FCM allow read/write control of hundreds of chip and algorithm parameters via a command line tool called "nanomix". All the examples given in the document, e.g. enabling or disabling the AEC block, can be accomplished with a series of nanomix commands, typed into the linux command line directly or embedded into a shell script.

5.1 Getting Started with the Command-Line: Connecting, SSH, SFTP

WiFi setup can be tricky – it is generally simpler to bootstrap up to WiFi connectivity via a wired serial console (usb cable) and then ethernet or wifi. Determine the IP address of your FCM according to the bring-up procedures in the FCM manual: ethernet or WiFi, Section 4.1. Some highlights are reproduced here:

The FCM has a dedicated serial-port jack – a micro-usb port (NOT the USB-C port). Connect the FCM to your PC with a microUSB-to-USB-A cable, note the COM port that appears under Device Manager, if you are running Windows. Download and run a free terminal emulator like PuTTY (Windows) or ZTerm (mac), with settings 8-N-1, 115200 baud.

Login to FCM with the password 'root'.

Plug in FCM ethernet and use "ifconfig" to get a listing of the active IP addresses (eth1, wlan0). When you have a useable IP address on eth1, open the web console via the steps in Section 4.1.

```
root@pico-imx8mm:~# ifconfig



eth0      Link encap:Ethernet  HWaddr 00:1f:7b:1e:02:d4
          inet addr:169.254.16.140  Bcast:169.254.255.255  Mask:255.255.0.0
          inet6 addr: fe80::21f:7bff:fe1e:2d4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29860 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:15610768 (14.8 MiB)

eth1      Link encap:Ethernet  HWaddr 00:50:b6:b7:ff:2b
          inet addr:192.168.2.75  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::250:b6ff:feb7:ff2b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:302923 errors:0 dropped:0 overruns:0 frame:0
          TX packets:412874 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28732033 (27.4 MiB)  TX bytes:321126533 (306.2 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:344772 errors:0 dropped:0 overruns:0 frame:0
          TX packets:344772 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:43560086 (41.5 MiB)  TX bytes:43560086 (41.5 MiB)


wlan0     Link encap:Ethernet  HWaddr 00:1f:7b:31:7c:bc
          inet addr:192.168.0.14  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::21f:7bff:fe31:7cbc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:802022 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51579 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3000
                   RX bytes:834243757 (795.5 MiB)  TX bytes:7772459 (7.4 MiB)
```

Although the command line utilities are useable via the console window (puTTY, Zterm), it is generally faster and simpler to use an IP connection to open a terminal on the target. The primary tools are SSH and SFTP.

Open a terminal window on your computer. On mac, click the spotlight icon at top right and type "terminal" to open a new terminal window. On PC, hover over the Start menu and type "cmd".

In this terminal window, use SSH in the format: ssh username@IP_ADDRESS_OF_TARGET. Recall that the IP address of our example FCM is 192.168.2.75.   SSH is for opening a terminal ON THE TARGET machine. You are effectively typing on the command line of the TARGET, in this case, the FCM.

```
kmac:~$ ssh root@192.168.2.75
Last login: Mon May  4 21:15:39 2020 from 192.168.2.1

root@pico-imx8mm:~# pwd
/home/root
```

Hit Ctrl-C to exit the target and return back to “your” computer.

SFTP is for transferring files, especially large .wav files, between a PC and the FCM. It uses the same format: sftp username@IP_ADDRESS_OF_TARGET. We take the perspective of the computer, here. We open a tunnel, and GET things from the target, or PUT things from our computer ON TO the target.

```
kmac:~$  sftp root@192.168.2.75
Connected to 192.168.2.75.
sftp> ls
AVSnix                  Alexa_SDK                InfoReporter
audioTests               example.wav
sftp> get example.wav

Fetching /home/root/example.wav to example.wav
/home/root/example.wav                    100%   15MB   9.7MB/s   00:01

sftp> put file.txt
Uploading file.txt to /home/root/file.txt
file.txt                               100%    8     7.3KB/s   00:00
sftp> exit
```

5.2 Audio Applications: aplay, arecord

aplay and arecord are fundamental audio play/record utilities in the ALSA (Advanced Linux Sound Architecture) software layer which translates PCM bitstreams between Linux applications and hardware audio devices (soundcards), providing services like mixing, sharing, sample rate conversion and format translation. Applications like Alexa or audacity or aplay each open a generic "default soundcard", and play and record PCM audio samples through it, simultaneously. Each application sees a virtual soundcard tailored to its own desired format: Number of channels, endian-ness, number of byes, and sampling rate. The actual physical soundcard in the FCM, the Cirrus CS48LV41f chip, is connected to the operating system with a fixed format: 2 channel, 16-bit, 48 kHz. Each soundcard speaks to ALSA in its own language, and ALSA speaks to the physical soundcard (via hardware drivers) in its desired format.

Let's get familiar with the command line by playing a wav file and then making a new recording from the mics. Make sure a speaker is plugged into the amp terminals, or use a 3.5mm audio cable to plug something (speaker, headphones) into the FCM's lineout jack. First, open a terminal window on the FCM target (section 4.4.1).

```
kmac:~$ ssh root@192.168.2.75
root@pico-imx8mm:~# cd /home/root/utils
root@pico-imx8mm:~/utils# aplay stereotest_fcm.wav
```

You should hear a voice saying "left, left, left" from the left speaker, followed by "right, right,right" out of the right hand speaker. If you are listening to only one speaker connected to a single amplifier terminal, you will naturally hear only one half of the output! We will discuss ways to change the output to mono, so that you don't miss out on any stereo content coming from applications like Alexa, due to the fact that the device has only a single mono loudspeaker. For now, let's move to the next step if we hear something rather than nothing.

```
root@pico-imx8mm:~#   arecord -c 2 -f S16_LE -r 48000 test.wav

Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
```

This starts recording a new wav file to be called test.wav, and we have explicity specified the recording format to be used, namely stereo (-c channels =2), linear 16-bit samples (-f format = signed, 16-bit, little endian) at a rate of 48kHz (-r rate=48000 Hz). Hit Ctrl-C to stop recording, or use the -d duration flag to specify a recording time in seconds.

Some useful variants:

```
root@pico-imx8mm:~#   arecord -f DAT x.wav (-f DAT is shortcut for c2/s16_LE/48k)
root@pico-imx8mm:~#   arecord -f DAT -d 10 y.wav  ( -d duration in seconds )
```

Now we have a recording stored on the filesystem of the FCM, in our current working directory.

```
root@pico-imx8mm:~# aplay test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
```

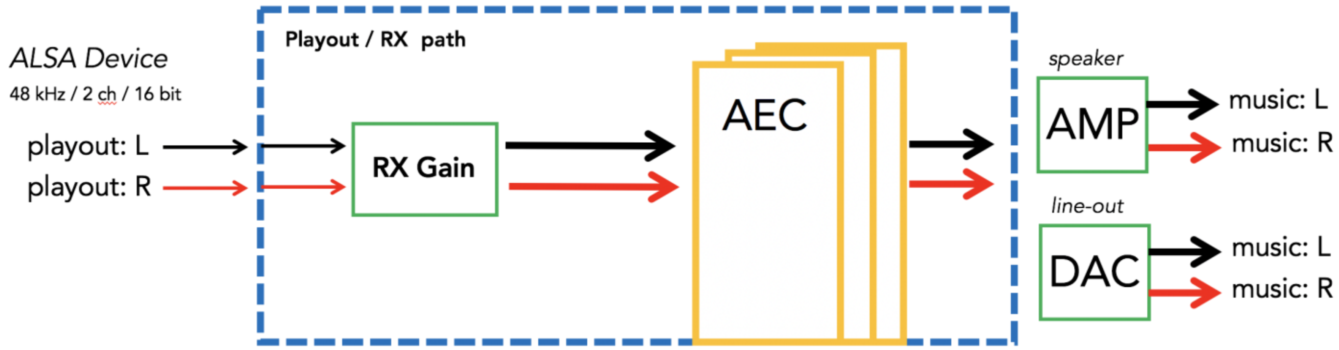
You should hear your recording coming out of the connected loudspeaker device. It may sound a bit funny, as you have made a stereo recording, and do recall that the mic (capture) signal has both left and right channels. By default, the CS48LV41f sends DIFFERENT signals in the left and right capture channels, see Fig.2 and Fig.3. This is to allow maximum flexibility, but is changeable via simple command line utilities described in S.4.4.4 – set_capture_(x,y).

Try exiting the SSH session (type "exit" at the prompt), open an SFTP session instead, and "get" the test.wav file you just created, following the example in the previous section (4.4.1). View test.wav in an audio editor like Audition or Audacity. Notice that the right channel output, the +SPECTRAL output, has more noise reduction applied than the left channel (SPATIAL) does. This will be very obvious if you make a recording of yourself making a sustained “shhhhhhhhhh” sound near the mics. The energy in the right channel will fade dramatically.

5.3 Application Output / Playback Routing

What if we have only ONE LOUDSPEAKER, but Alexa is playing stereo content? Here is another copy of Figure 2, just the PLAYOUT path (top) portion:

Figure 13 : Stereo Playout Signal Path of SCFF



Note that the SCFF signal path is fundamentally STEREO. Applications like Alexa play *stereo content* (the black and red arrows) into a *stereo* soundcard which comes out of a *stereo* TRS jack (DAC, line-level output) or a pair of digital class-D amplifiers (AMP). There are TWO amps, one for left, one for right. The echo canceller is a *stereo* AEC.

If it so happens that the content coming from the application is mono, there is no problem. The ALSA layer will duplicate the mono signal from the application to fill both the left and right inputs of Fig 13. Imagine the red lines in Fig.13 are black – it’s just a particular case of stereo. The stereo AEC will still function as expected.

But back to our dilemma, what if Alexa is playing stereo content but we have only ONE LOUDSPEAKER? We can connect our loudspeaker to the left AMP or the right AMP terminals, but not both. If Alexa is playing a the 1965 stereo mix of Rubber Soul, where Paul’s bass is panned hard left and George’s guitar is panned hard right, we have an impossible choice to make. If we plug our single loudspeaker into AMP terminal pair #1, we would hear the left channel with bass only, and none of George’s guitar!

The simplest solution is to set one or both of the potential outputs, the AMPs or the LINEOUT, to be a MONO device, and mix both channels together before they go out to the mono loudspeaker. This solution allows us to keep a single signal flow diagram for both mono and stereo use cases. We don’t have to reconfigure things for each application, and now stereo and mono applications can co-exist and speak to the same soundcard. We just reconfigure the *physical outputs* to accommodate our need to play a MONO MIX signal to our one loudspeaker.

Let’s try it. We have already heard (section 4.4.2 above) a test file that is very obviously stereo, with someone helpfully saying “left left left” in the left channel, and “right right right” in the right channel.

```
root@pico-imx8mm:~# cd /home/root/utils
root@pico-imx8mm:~/utils# aplay stereotest_fcm.wav
```

now plug a loudspeaker into one pair of amp terminals, aplay the file, and we hear our dilemma. It’s Paul or George, left or right, but not both! Try switching to the other amp terminals to verify this. We are playing a stereo file, and have to choose which channel to listen to with our single loudspeaker. This is unacceptable!

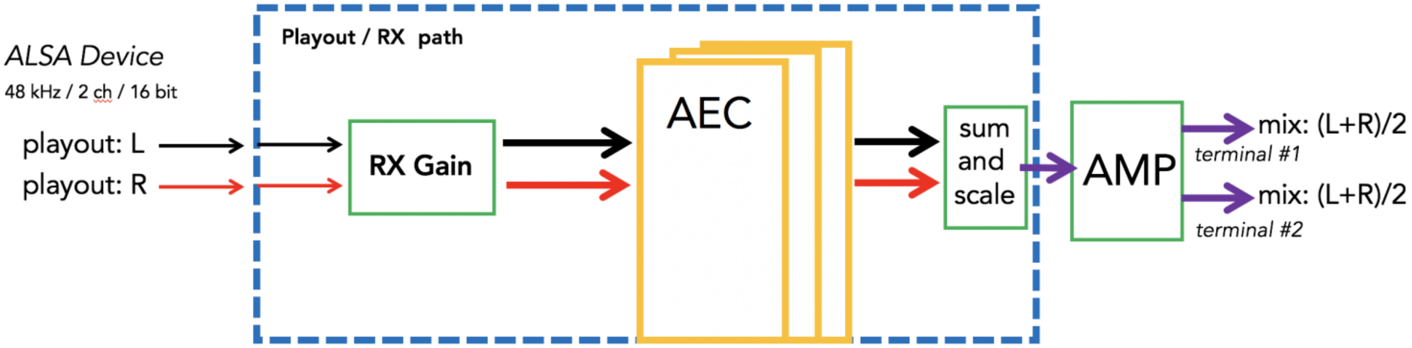
Now type the following, to run the script called “set_amp_mono”. Notice the “dot slash” characters that precede it! This tells linux to treat the file as a script, and execute it immediately.

```
root@pico-imx8mm:~/utils# ./set_amp_mono
root@pico-imx8mm:~/utils# aplay stereotest_fcm.wav
```

Now we hear BOTH channels coming out of the single loudspeaker. All the script does (view it with the ‘nano’ editor, or type “cat set_amp_mono” to print it to screen) is to MIX the stereo signals (black, red) before they enter the AMP block. It doesn’t matter if you plug the loudspeaker into amp terminal #1 or #2. This is much safer, to avoid potential error when using a single loudspeaker.

Mixing two PCM audio streams is simply to ADD the two signals together, sample by sample, and divide the result by two to prevent saturating. Mono signal = (left signal + right signal)/2. In the digital domain, dividing by two is the same as reducing the volume (voltage; amplitude) by 6 dB.

Figure 14 : set_amp_mono – sends same mono mix to both AMP terminals



To restore the amp output to normal, default stereo operation, where terminal #1=left, terminal #2 = right, use

```
root@pico-imx8mm:~/utils# ./set_amp_stereo
root@pico-imx8mm:~/utils# aplay stereotest_fcm.wav
```

Note the same “mono mix” trick can be done with the stereo line-out jack also, via the “set_lineout_mono” and “set_lineout_stereo” scripts in the same directory.

This function is really only useful for demonstrating an Alexa device with a single loudspeaker. Otherwise – stick with stereo. The applications will take care of themselves, there is no need to switch anything in the signal path because Alexa alternates between mono speech and stereo music. The power-on default is stereo, as in Fig.2 and Fig.3.

5.4 Application Capture / Mic Routing:

“Capture” refers to the ALSA input device, typically a microphone, to which applications listen when they read PCM bitstreams from an ALSA soundcard.

The SCFF implementation is very flexible. Here are command-line utilities for selecting which processed or raw mic signals will fill the left and right channels of the CAPTURE path, seen by Linux applications on FCM like Alexa or arecord.

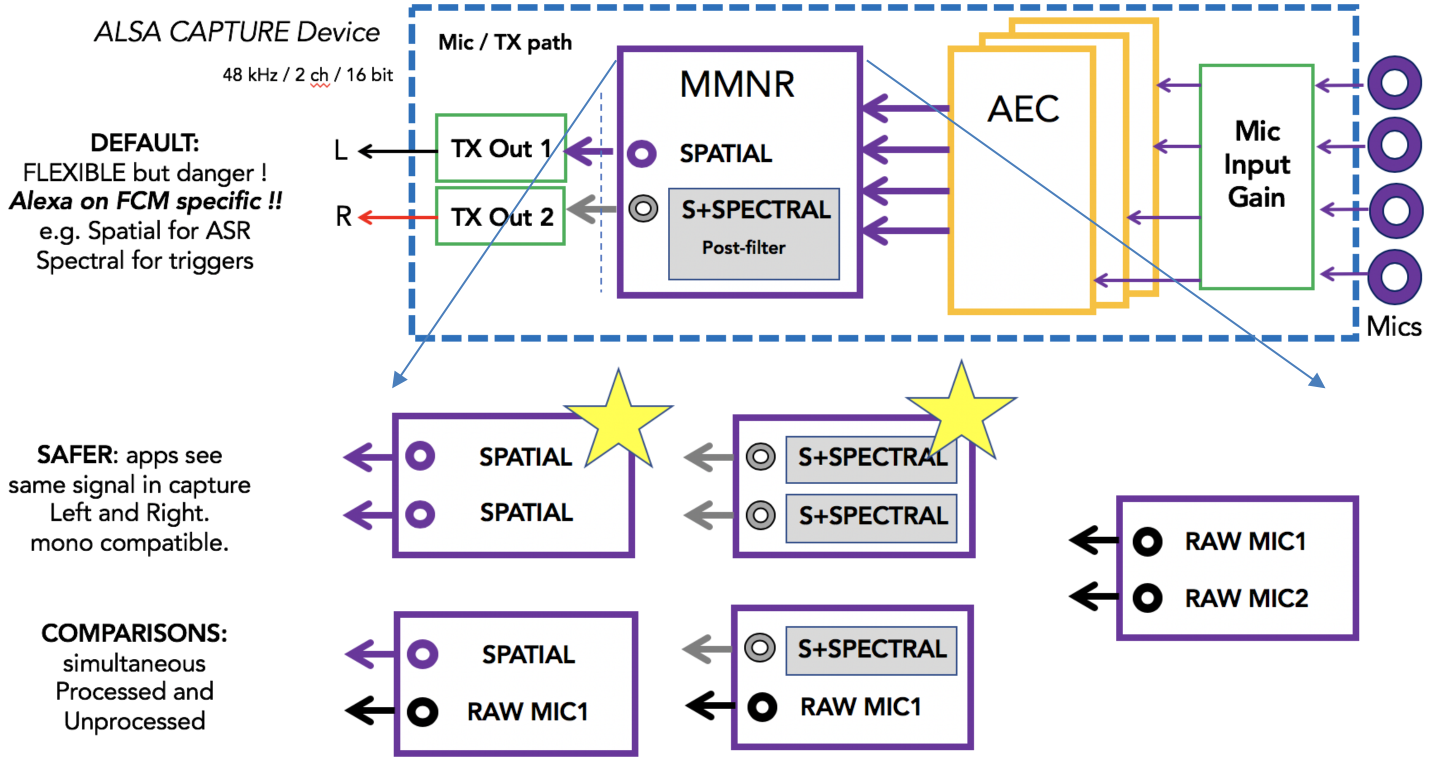
The SCFF algorithm produces three signals of interest: The raw microphone signal(s), and two processed mic signals, known as SPATIAL and+ SPECTRAL. Please read Section 3, SCFF algorithm overview, before continuing.

Recall from S.3 that SPATIAL refers to the mono, processed mic signal that emerges from MMNR processing, the output of the adaptive beamformer. Thanks to the beamformer, this signal generally has a much improved SNR vs. raw microphone signals. A fixed-position noise source will be suppressed, and this improvement can be watched in real-time in the “Beamformer NR” diagnostics panel (S4.3.7). This signal is quite linear, and is good input to cloud ASR engines.

+SPECTRAL refers to a mono, processed mic signal which is derived from the SPATIAL, but which includes ADDITIONAL spectral subtraction (single-ended noise reduction) processing. Hence the SNR is further improved over the SPATIAL signal, if the noise sources are stationary in the spectral sense (e.g. white noise, fans, road noise). Some cloud ASR vendors do NOT want spectral noise reduction processing in their inputs, although the spectral output is generally preferred by human listeners in voice calls. Hence the +SPECTRAL output is generally intended for voice applications like Skype or Jitsi.

Figure 15 is a restatement of Fig.2, in which various options for the Capture_L and Capture_R signals seen by ALSA applications at the left hand side, are presented:

Figure 15 : Input / Mic / CAPTURE Routing



The simplest to understand are the “mono compatible” routings, in which the ALSA capture device sees the same signal in the left and right channels, e.g. { spatial, spatial }, { spectral, spectral }. These are marked with gold stars as being the safest, in the sense of least confusion about what applications are consuming. Some applications listen to only the left or the right input. Some applications mix the mic inputs into a mono sum. Some applications transmit stereo (e.g. Jitsi voice calls). The only way to guarantee that the application receives the signal you expect, is to transmit the identical signal in left and right capture channels.

The default routing (v1.0.1) is (spatial, +spectral). The left capture signal is different than the right capture channel. This gives maximum flexibility, as both outputs of the SCFF algorithm are available simultaneously over a two-channel interface. It allows a use case in which the Alexa application uses the right channel +spectral signal to feed a wakeword engine, and the left channel spatial signal to send up to the cloud, getting the best of both worlds. However, this is definitely confusing and leaves room for error. Using this default (spatial, +spectral) routing would give odd results if piped to a stereo VoIP application like Jitsi.

The remaining routings (raw mic1, raw mic2), (spatial, raw mic1) and (+spectral, raw mic1) are clearly for evaluation or comparison experiments. They allow one to record (using the arecord utility, S5.2) both the processed and unprocessed signals simultaneously. Note the (spatial, +spectral) routing is also useful for comparisons. Use arecord to capture audio as stereo/16bit/48kHz wav files (‘arecord -f DAT test.wav’), sftp the recorded wav files from the target to a PC running an analysis viewer like Adobe Audition, Audacity, etc. and listen/look at the differences.

```
kmac:~$ ssh root@192.168.2.75
root@pico-imx8mm:~# cd /home/root/utils
```

(mono compatible; same singnal in both left, right capture)

```
root@pico-imx8mm:~/utils# ./set_capture_spatial
root@pico-imx8mm:~/utils# arecord -f DAT spatial.wav

root@pico-imx8mm:~/utils# ./set_capture_spectral
root@pico-imx8mm:~/utils# arecord -f DAT spectral.wav
```

(be careful, different signals in left, right capture.)

```
root@pico-imx8mm:~/utils# ./set_capture_spatial_raw
root@pico-imx8mm:~/utils# arecord -f DAT spatialraw.wav

root@pico-imx8mm:~/utils# ./set_capture_spectral_raw
root@pico-imx8mm:~/utils# arecord -f DAT spectralraw.wav

root@pico-imx8mm:~/utils# ./set_capture_raw
root@pico-imx8mm:~/utils# arecord -f DAT raw.wav

root@pico-imx8mm:~/utils# ./set_capture_spatial_spectral
root@pico-imx8mm:~/utils# arecord -f DAT spatialspectral.wav


root@pico-imx8mm:~/utils# exit
kmac:~$ sftp root@192.168.2.75
root@pico-imx8mm:~# get utils/*wav
```

Note – the comparison routings are good for hearing the delta that SoundClear processing is making, after the fact. This can be heard in real-time, by the simple expedient of enable/disabling SoundClear processing or individual blocks (MMNR, AEC) in the web console.

Note – to change the default routing to one of these options, e.g. (spatial,,spatial), modify the FCM startup script in /home/root/AVSnix/runAvs.sh. Add a call to one of these scripts directly after the call to set_mic_pattern.sh, using the full path /home/root/utils/set_capture_xxx . Reboot, and verify that new routing has become the new default.

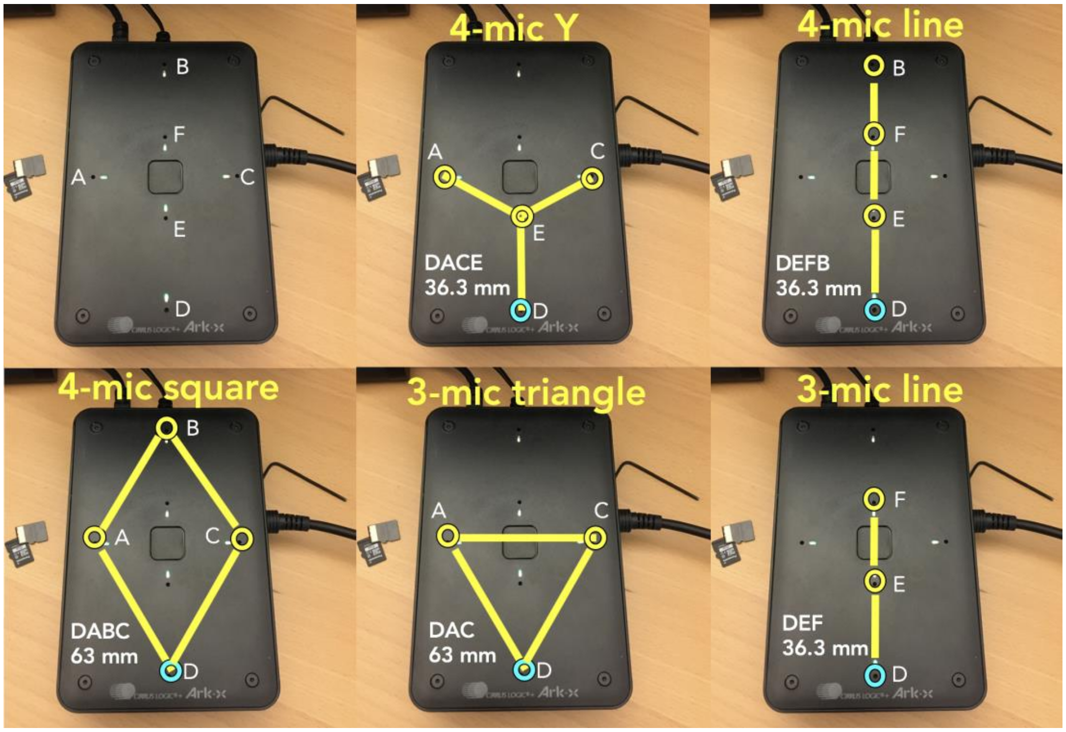
```
nano /home/root/AVSnix/runAvs.sh
…
# Set MIC PATTERN, which will replace the symlinks in /opt/tunings to point
# at the desired files in /home/root/AVSnix/cirrusACFs/
# then LOAD the new tuning and routing files into the Wright.
# USAGE:  set_mic_pattern.sh (dace, df, ca, def, pattern name)

# default pattern is Y-array D,A,C,E,  with 36mm mic spacing
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DACE
sh /home/root/utils/set_capture_spectral
```

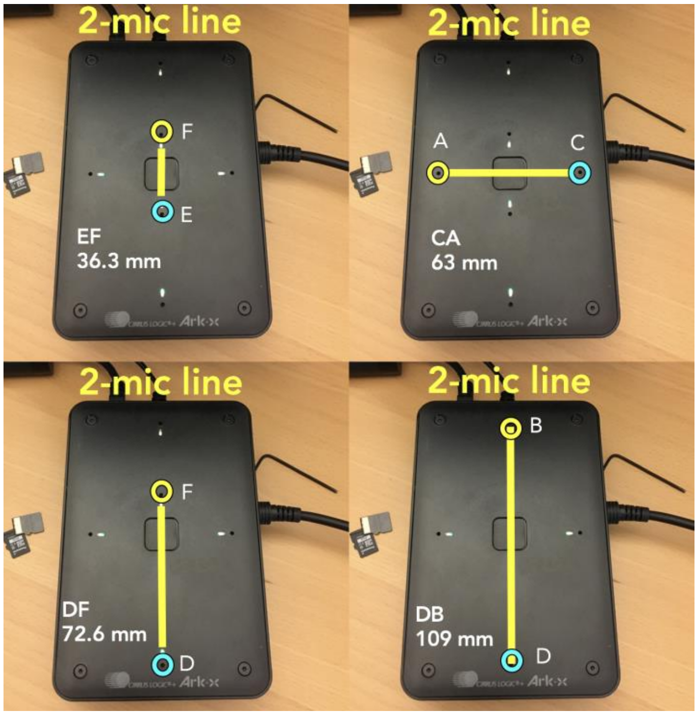
Note – the (x, raw) capture routings take the raw mic signal from Mic1, the PRIMARY mic (plus 10dB gain, modify the set_capture_xx script directly to change the mic gain). For example, if the mic geometry set in /home/root/AVSneix/runAvs.sh (above) is 'DACE', which is the default, then the PRIMARY mic is mic D and the SECONDARY is A. See the FCM User's Guide for photos of the mic labels.

The (raw mic 1, raw mic 2) capture routing is not simply the primary mic signal duplicated. Rather, it captures the PRIMARY and SECONDARY mics, the first two listed in the argument to set_mic_pattern.sh. In our example, it would capture mics D and A in left, right capture channels respectively.

5.5 Microphone Array configuration

Figure 16: FCM Microphone Array Patterns, DACE, DEFB, etc.





Sometimes, a picture is worth 1000 words. The SCFF algorithm supports 2,3 and 4 mic configurations. The default 4 mic configuration is the Y array, DACE, which has 36.3mm inter-mic spacing. The recommended 2-mic line array is DF, with 72.6mm spacing.

The PRIMARY mic is marked in blue in each diagram. The primary and secondary mics are simply the first two letters of the array name. Recall that the primary mic is captured (Section 5.4) in the (spatial,raw) or (spectral,raw) capture routings, and the the primary and secondary are captured in the (raw mic 1, raw mic 2) routing.

The primary and secondary mics are merely labels. All mics are equal to the SCFF algorithm. What is important is the shape, and the mic spacing. The algorithm is quite robust to changes in geometry, however, the automatic mic tuning tool in SoundClear Studio must be used to create a custom tuning for mic patterns that are different from these examples which are included with FCM. For example, if you have developed a product that uses a 4-mic line array with 20mm spacing, please ask your vendor to obtain the SoundClear Studio tuning tools.

5.5.1 Changing Default Microphone Array Pattern

The default mic array pattern is set in a single line, in the file /home/root/AVSnix/runAvs.sh, on the FCM filesystem. The file can be modified simply using the ‘nano’ text editor on the FCM, and the change will take effect after a reboot. This change will be persistent across reboots.

```
kmac:~$ ssh root@192.168.2.75
root@pico-imx8mm:~# nano /home/root/AVSnix/runAvs.sh

…
# Set MIC PATTERN, which will replace the symlinks in /opt/tunings to point
# at the desired files in /home/root/AVSnix/cirrusACFs/
# then LOAD the new tuning and routing files into the Wright.
# USAGE:  set_mic_pattern.sh (dace, df, ca, def, pattern name)

# default pattern is Y-array D,A,C,E,  with 36mm mic spacing
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DACE
```

Simply change the pattern ‘DACE’ to one of the patterns shown in Figure 16, e.g.

```
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DACE
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DEFB
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DABC
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DAC
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DEF
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh EF
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh CA
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DF
sh /home/root/AVSnix/cirrusACFs/set_mic_pattern.sh DB
```

and reboot. When you run alexa, you will note that the active mics turn blue.

6 Revision History

Revision History

Revision	Changes
1.0 MAY 20	• Initial Release. kevin.connor@cirrus.com
1.1	•

Contacting Cirrus Logic Support
For all product questions and inquiries, contact a Cirrus Logic Sales Representative.
To find one nearest you, go to www.cirrus.com.