EAP in the MCUXpresso SDK

Application Note

Released 4.0 – ***2021-sept-21*** **PUBLIC**

Document information	
Title	Essential Audio Processing Application Note
Status	Released
Version	4.0
Date	2021-sept-21
Security	Public
Usage	**NXP**

Change History

Version	Description	Date
1.0	Initial	2020/04/08
1.1	1st Review	2020/04/09
1.2	Add i.MX RT600 platform integration information	2020/04/09
2.0	Update to match EAP 2.0.2 release Update for i.MX RT600 and i.MX RT500 platform Update MusicEnhancer with RMS Limiter Update simulator tool chapter	2020/08/21
2.1	Update to be compliant with EAP release 2.4	2020/09/11
3.0	Add Crossover module description for EAP 3.0.0	2021/04/06
3.1	Add details about output buffers management	2021/04/14
3.2	Add I.MXRT1170 in NXP devices list	2021/06/11
3.3	Add a reference to 1.7.2 in Crossover presets	2021/06/30
4.0	Update on 24-32bits/96kHz	2021/09/21

TABLE OF CONTENT

1 DOCUMENT DESCRIPTION

1.1 Purpose

This document presents the Essential Audio Processing (EAP) package elements available in NXP's MCUXpresso SDK.

This document is compliant with :

- the package **EAP16_3.0.x_FPx**
- The package **EAP32_1.0.x_FPx**

It includes EAP library to support following platform:

- i.MX RT600 (Cadence® Tensilica® HiFi 4) – **EAP16_3.0.x_FPx / EAP32_1.0.x_FPx**
- i.MX RT500 (Cadence® Tensilica® Fusion F1) – **EAP16_3.0.x_FPx**
- i.MX RT1050 (Cortex-M7) – EAP 3.0.0 – **EAP16_3.0.x_FPx / EAP32_1.0.x_FPx**
- i.MX RT1062 (Cortex-M7) – EAP 3.0.0– **EAP16_3.0.x_FPx / EAP32_1.0.x_FPx**
- i.MX RT1170 (Cortex-M7) – EAP 3.0.0– **EAP16_3.0.x_FPx / EAP32_1.0.x_FPx**
- LPC55S69 (Cortex-M33) – EAP 3.0.1– **EAP16_3.0.x_FPx**

Refer to chapter *1.5.11* for release details.

1.2 Overview

The EAP package in the MCUXpresso SDK includes:

- An EAP user guide
- A pre-compiled library for a platform
- A tuning tool simulator
- An EAP integration example based on simulation (different IDE is proposed)
- An EAP integration example based on a hardware platform integration

1.2.1 What to do with this EAP package?

Learn about the EAP feature and EAP tuning capability for each processing block:

- Read the EAP user guide.
- Parse the parameter text file preset (or C code header preset).

Evaluate EAP and play with the parameters on a PC (no hardware required):

- Read Tuning tool simulator chapter.
- Install the EAP tuning tool simulator on your PC.
- Run and listen demo.
- Select a parameter preset and apply it to the soundtrack of your choice.
- Listen and compare the results thanks to the EAP tuning tool simulator.

Learn about the integration of the EAP library:

- Read the EAP user guide.

- Parse or Run the EAP integration example based on platform simulation.
  Full EAP API is used as example.
- Parse or Run the EAP integration example based on a hardware platform integration.
  Essential EAP API is used.

Enjoy the EAP on the EAP SDK board

- Run the EAP integration example based on a hardware platform integration.
- Select a parameter preset with the Uart command.
- Update the custom preset (header file) based on your requirements.
- Recompile and listen to the results.
- Compare with original file.

1.3 EAP User guide

The EAP user guide (EAP_userGuide.pdf) provides information to understand, tune and integrate the Essential Audio Processing solution in your product.

For each processing block, it provides:

- A description of the behavior.
- A description of the tuning parameters to perform classic tuning.

An additional chapter explains how to perform the integration.

1.4 Tuning tool simulator

The tuning tool simulator permits:

- To control each EAP parameters.
- To simulate the EAP processing behavior on a Windows PC environment.

Simulator provide same output audio file (bit exact) than the library and can be used to find the right tuning parameter setting.

Tuning tool simulator also provides:

- A player to listen and compare A/B files.
- A tool to generate a C code header parameter file

1.5 Audio preset

The EAP package is coming with audio preset parameters.

- Audio preset header file (.h) is dedicated to C code.

User can modify them manually with the help of the comments included in the preset file and the EAP user guide.

Audio tuning is an important point and must be considered to obtain great audio behavior. It permits to adapt the EAP processing feature to the product elements:

- the audio chains.
- the speakers.
- the casing of the speaker.

The preset effect can't be optimal for your design but are a good start of point.

1.5.1 AllEffectOff

All Effect are off.

Only volume control and balance are applied.

1.5.2 VoiceEnhancer

Equalizer with 3 bands are set to enhance voice frequency part.

Loudness maximiser with 2dB gain in Gentle mode is used.

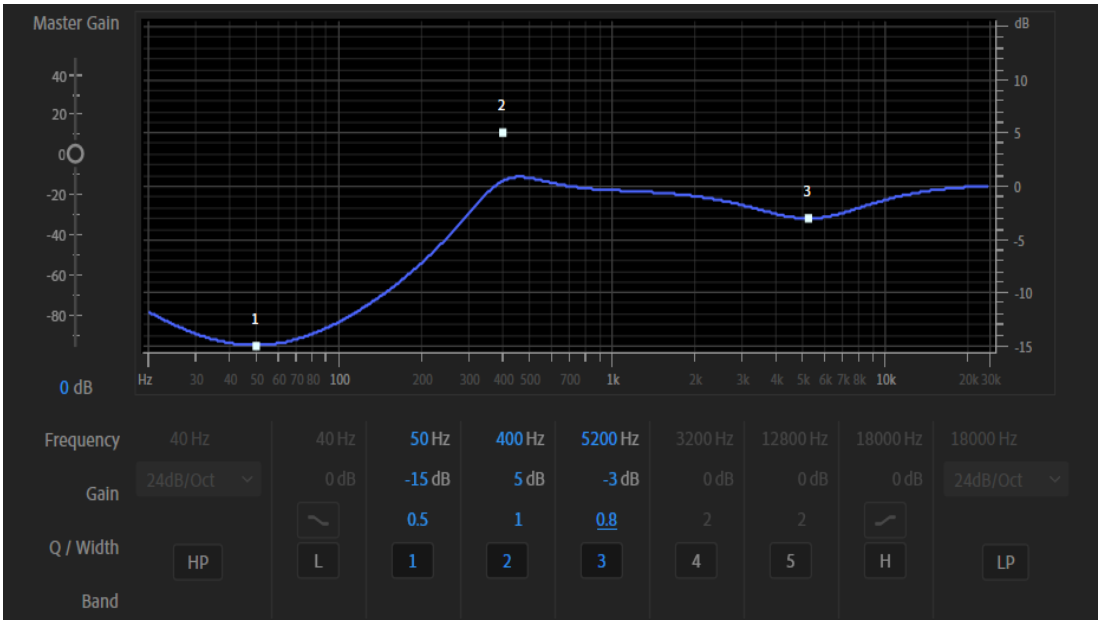

Figure 1 Voice parametric 3 bands equalizer curve (from adobe audition)

Effect:

- Voices are more present
- Soundtrack is less muffled

1.5.3 MusicEnhancer + RMS Limiter

Equalizer with 4 bands are set to increase frequency part up to 1KHz.
Treble enhancer with 4dB gain curve is used to boost the high frequency
Dynamic bass enhancer is added to increase bass dynamically. (G=9dB, Fcenter=90Hz)
RMS limiter with input signal reference is set to -3dB.

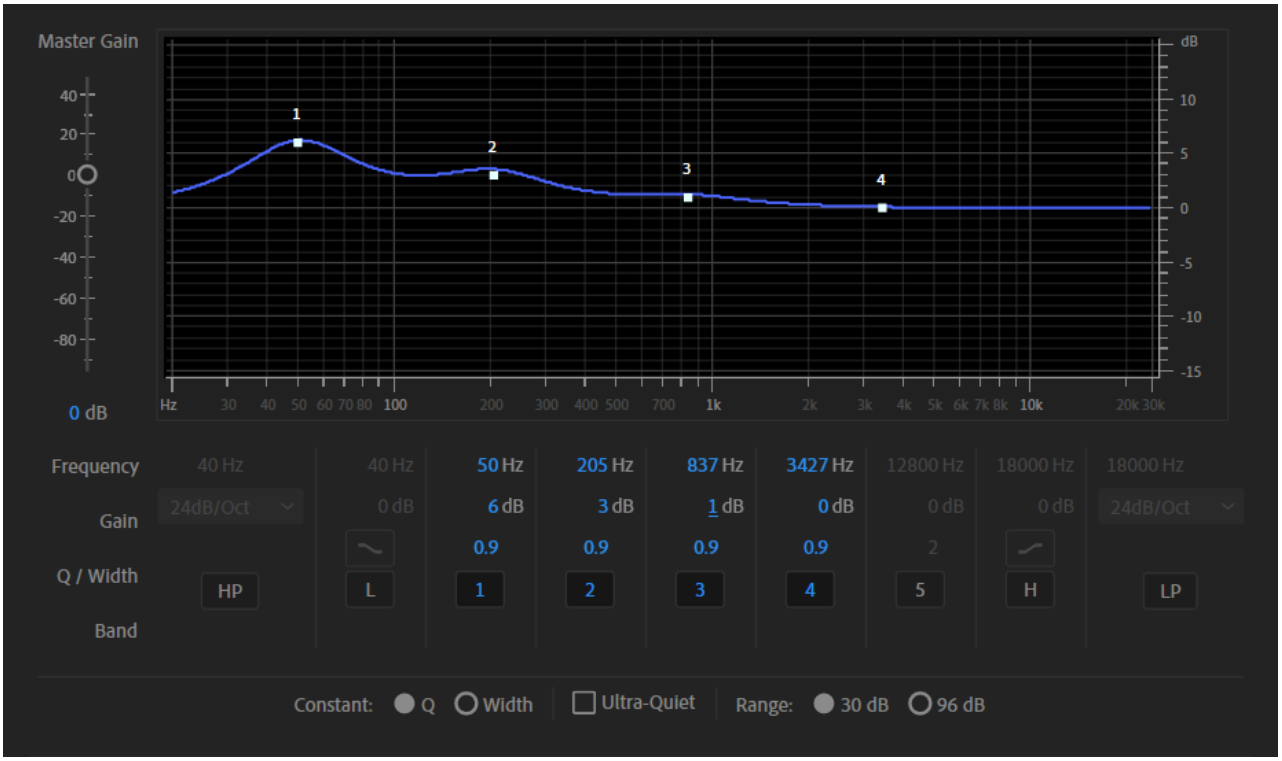

Figure 2 Music parametric 4 bands equalizer curve (from adobe audition)

Effect:

- Bass frequencies are more present and deeper
- Treble frequencies are more present.
- RMS value is 3 dB lower than the input signal.

1.5.4 AutoVolumeLeveler

Auto volume leveler is turned On.
High pass filter of the dynamic bass enhancer is used to avoid any residual DC offset.
Volume parameter (VC_EffectLevel) is set to -9dB.
Effect:

- Volume between soundtracks is constant
- Low volume part of a soundtrack is boosted

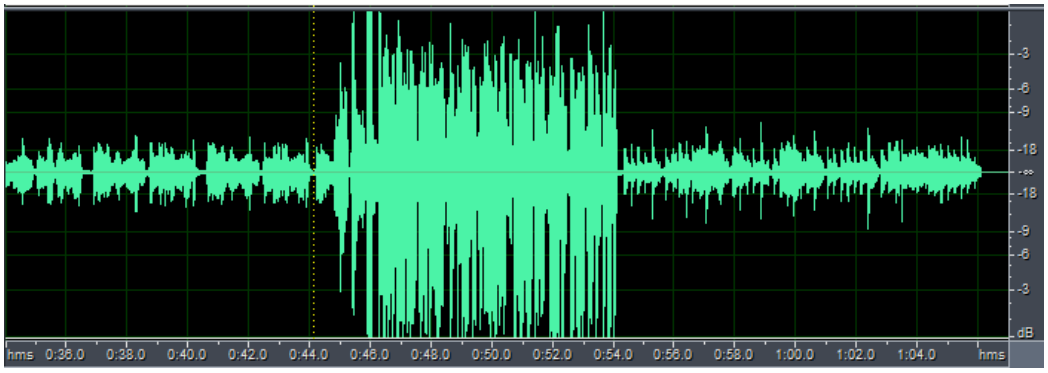

Figure 3 Input audio file

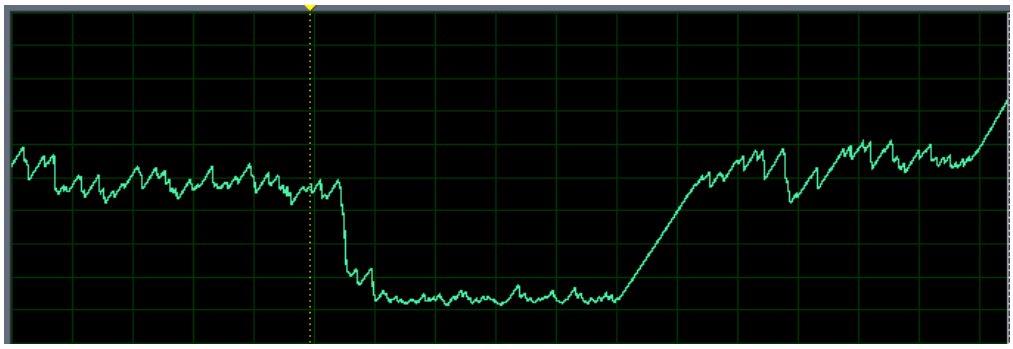

Figure 4 Input audio file AVL gain applied (see Read AVL Gain API chapter)

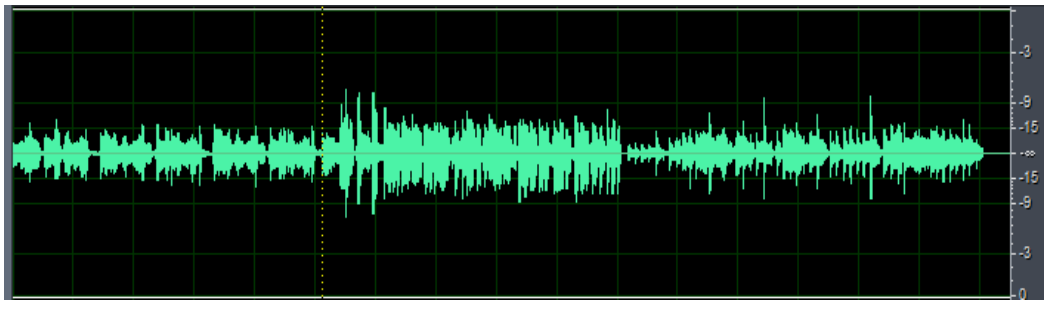

Figure 5 Output audio file

1.5.5 ConcertSound

3D Virtualizer is turned ON and the virtualizer type is set to concertSound.

Effect:

- Music appears more distant
- Some reverberation is present and provide large room feeling

1.5.6 LoudnessMaximiser

Volume control is set to -3dB.
Loudness maximiser with 6dB gain in Medium mode is used.

Effect:

- Volume boost of 6dB apply to the soundtrack whereas peaks values stay the same
- Get the maximum sound level of a speaker without damage it. It is generally used with small speakers.

1.5.7 Custom

Equalizer with 4 bands are set to increase frequency part up to 6KHz.
Treble enhancer with 9dB gain curve is used to boost the high frequency
Dynamic bass enhancer is added to increase bass dynamically. (G=6dB, Fcenter=55Hz)
3D Virtualizer is turn ON and the type is set to concertSound.

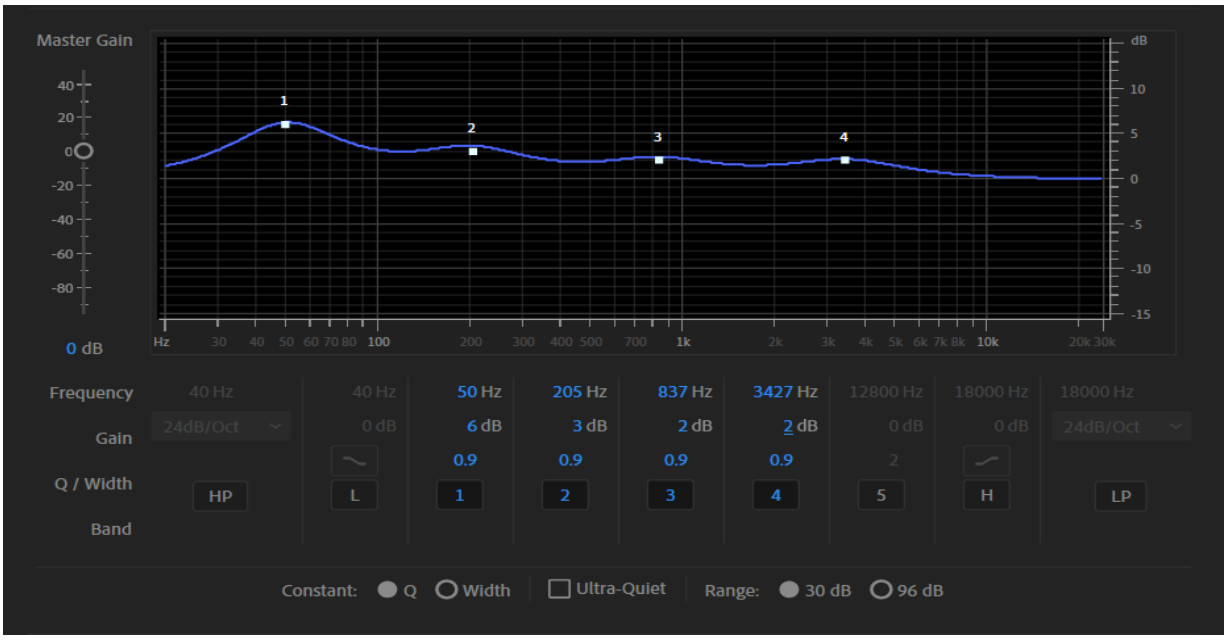

Figure 6 Music parametric 4 bands equalizer curve (from adobe audition)

Note: In the example based on the RT600 platform integration, the custom setting is your setting. Don't hesitate to modify it to adapt to your hardware.

1.5.8 Tone Generator

Sweep tone generator is turned On:

- 1 shot of 10seconds.
- Sweep linear mode

Other processing block are turned off.

Effect:

- At the beginning of the process, a sweep tone generator of 10 seconds is generated
  It is a tone starting from 20Hz to 22KHz.
  The start amplitude is -12dB
  The stop amplitude is -3dB
  It permits to execute some measures. Per example, measure of the spectral response of the speaker.

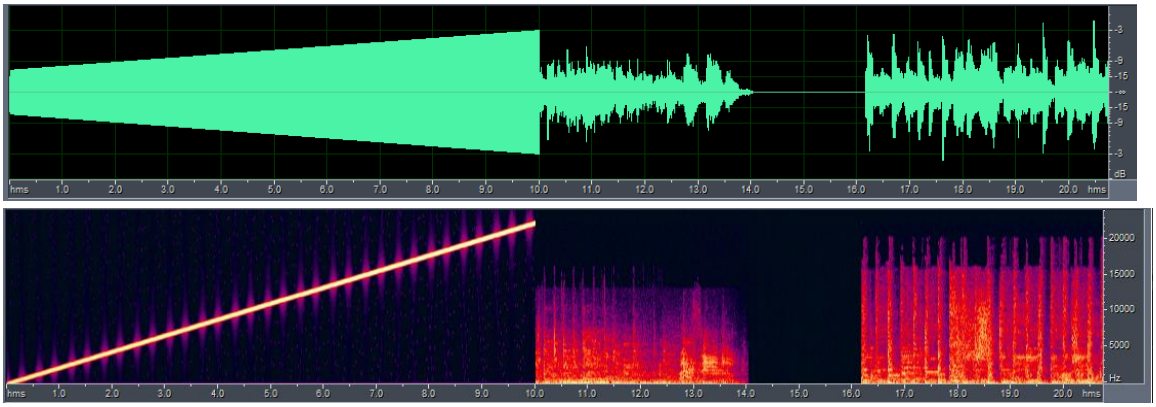

Figure 7 Sweep Tone Temporal & Spectrum (from adobe audition)

Note: Take care to keep low amplitude sweep tone to not damage your speaker.

1.5.9 Crossover for subwoofer

In this case, Crossover is the only feature enable. We want to separate the frequency range of our input audio signal in two bands in order to distribute the low band to a subwoofer. The high band will be redirected to a medium speaker. The cut-off frequency of our subwoofer is around 100Hz (in our example case), so the cut-off frequency of the Crossover will also be 100Hz.

Effect:

- The low band range = [0 Hz – 100Hz]
- The high band range = [100 Hz – Fs/2 Hz]

See 1.7.2 for output management on board

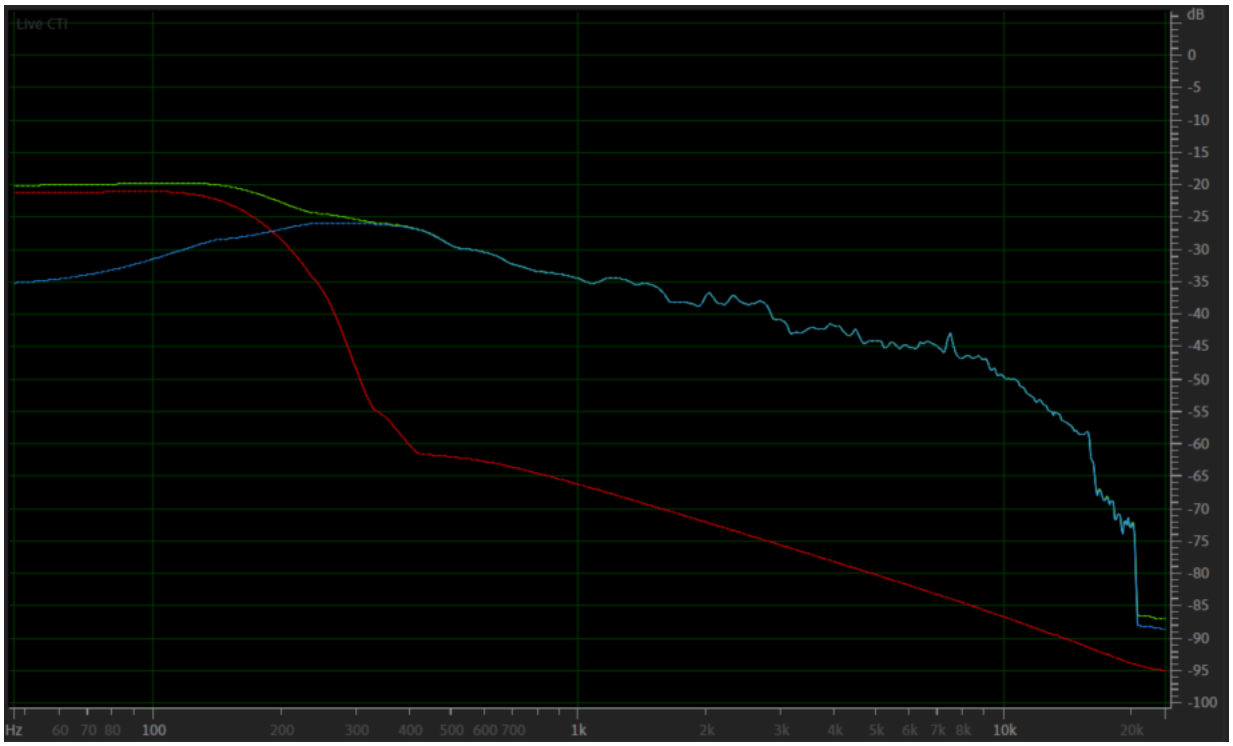

Figure 8: Frequency diagram of the signal after Crossover filtering

Note: green curve = input signal; red curve = Low band of the signal; blue curve = High band of the signal;(green and blue curve are confused for frequencies >300kHz)

1.5.10 Crossover 2 way speaker

In this configuration, the crossover is the only feature enable. Our example audio device is composed of two speakers, the first one has a low /medium range and a cut-off frequency of 3.5kHz. The second one is a tweeter for high frequencies. We can set the Crossover cut-off frequency at 3.5kHz.

Effect:

- The low band range = [0 – 3500 Hz]
- The high band range = [3500 Hz – Fs/2 Hz]

See 1.7.2 for Output management on board

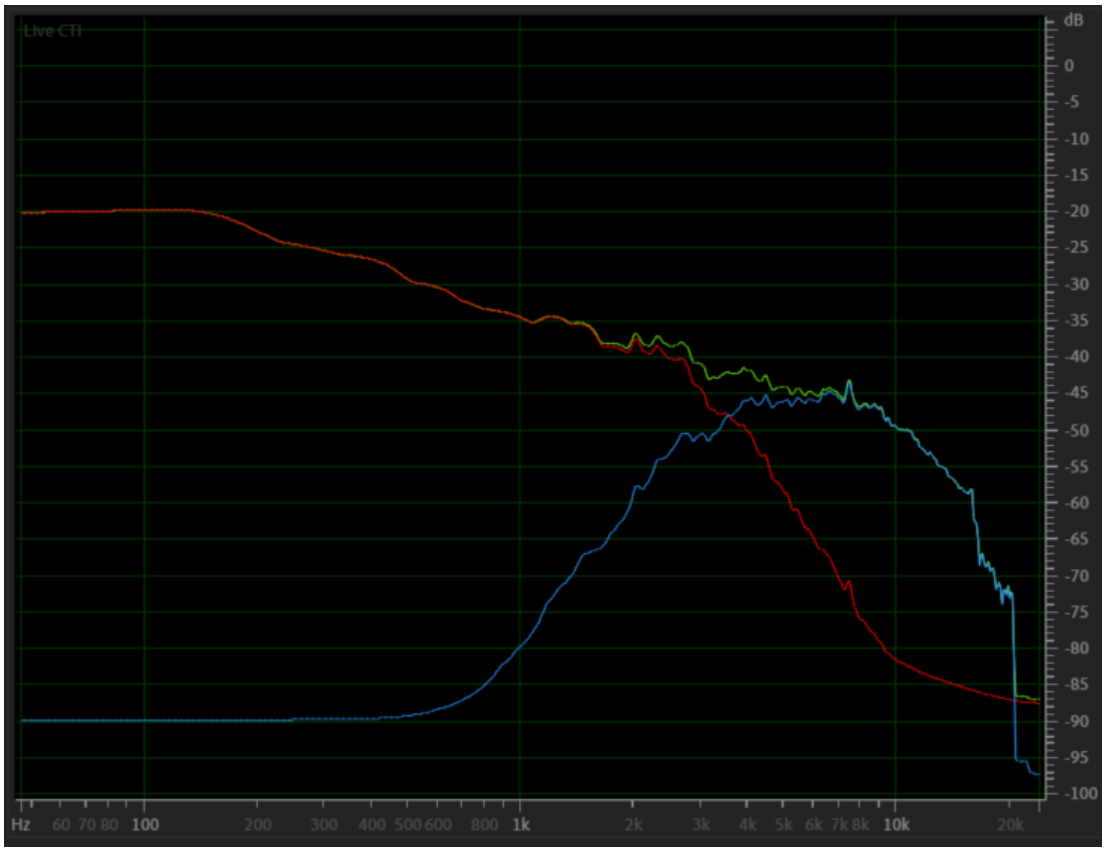

Figure 9: Frequency diagram of the signal after Crossover filtering

Note: green curve = input signal; red curve = low band of the output signal; blue curve = high band of the output signal;(red and green curves are confused for frequencies <2kHz. Green and blue curves are confused for frequency >6kHz).

1.5.11 Pre-compiled library

According to the compilation symbol option, a selection of the processing blocks is available in the library.

When programmer include this library, he must declare the same symbol definition in his project.

Read release note “libEAP_releaseNote.txt” to know:

- Compilation flag.
- Symbol definition.
- Platform supported.
- Algorithm supported.

1.6 EAP integration example

The EAP example integration file utilizes the entire EAP library API. It shows how to:

- Initialize the EAP library
  - Get EAP library memory requirement
  - Create an instance of the EAP library
- Set a preset parameter
- Update a parameter
- Volume update with no smoothing
- Get the Power Spectrum Analysis
- Get the current AVL gain

1.6.1 Initialize the EAP library

Parse the EAP_ExApp.c code as an example.

1.6.2 Set a preset parameters

Parse the EAP_ExApp.c code as an example.

A define permits to choose which preset to apply at initialization.

- EAP_PARAM_ALL_EFFECT_OFF
- EAP_PARAM_VOICE_ENHANCER
- EAP_PARAM_MUSIC_ENHANCER_RMSLIMITER
- EAP_PARAM_AUTO_VOLUME_LEVELER
- EAP_PARAM_CONCERTSOUND
- EAP_PARAM_LOUDNESS_MAXIMISER
- EAP_PARAM_TONE_GENERATOR
- EAP_PARAM_CUSTOM
- EAP_PARAM_CROSSOVER_FOR_SUBWOOFER
- EAP_PARAM_2WAY_SPEAKER
- EAP_PARAM_TEST_ALL_ON

1.6.3 Update a parameter

This example shows how to update one or multiple EAP parameters.

Parse the EAP_ExApp.c code as an example with EXAMPLE_PARAMETER_UPDATE defined.

1.6.4 Volume update no smoothing

When volume parameter (VC_EffectLevel) is updated, a volume smoothing between old and new volume is applied to avoid volume gap artifact.

This example shows how to update volume parameter without any smoothing.

Parse the EAP_ExApp.c code as an example with EXAMPLE_VOLUME_UPDATE_NO_SMOOTHING defined.

1.6.5 Power Spectrum Analysis

Parse the EAP_ExApp.c code as an example and focus on section code defined by ALGORITHM_PSA.

This example read the PSA and write the value in a .dat files.

1.6.6 Read AVL Gain

API permits to read the gain used by the Auto Volume Leveler.

Parse the EAP_ExApp.c code as an example with EXAMPLE_AVL_READ_GAIN defined.

In this example the gain is duplicated to fill a full frame size and is written as it is a pcm file. It permits to analyse the AVL behavior (see Read AVL Gain chapter).

1.7 EAP integration example based on a hardware platform integration

This EAP integration example :

- Runs on a NXP SDK evaluation kit.
- Provides full audio solution based on Xtensa Audio Framework from Cadence Design Systems for i.MX RT500 & 600 .
- Provides full audio solution based on Maestro Audio Framework from NXP for i.MX RT1062,RT1050, RT1170 & LPC55S69.
- Shows essential EAP library integration requirements
  (for full API example refer to EAP_ExApp.c)
- Permits to select EAP preset parameters
- Permits to increase or decrease EAP volume parameter
- Permits to control left/right audio EAP balance parameter

1.7.1 Control the EAP preset parameters

A UART command permits to:

- select an EAP preset parameters
- increase or decrease EAP volume parameter
- control left/right audio EAP balance parameter

Others EAP parameters are not accessible to the UART command and need header file update + project re-compilation.

A custom preset parameter (EAP_Parameter_Custom.h) is dedicated for the user parameters and can be updated.

A simple command explanation is displayed in the application using command ‘help’ in the shell. For further information, a readme.txt attached to the example application is present.

1.7.2 Crossover use case specification

(For more details about Crossover module, please see EAP_User_guide.pdf)

When the Crossover module is enable, the output will be split in two parts : the low band and the high band. This is the basic function provided by the crossover module.

If Crossover Disable:

If the signal is in MONO, the EAP output will be composed of 1 output buffer :

- EAP mono buffer

If the signal is in STEREO, the EAP output will be composed of 2 output buffers :

- EAP right channel buffer
- EAP left channel buffer

If Crossover enable:

If the signal is in MONO, the Crossover output will be composed of 2 output buffers :

- EAP low band mono buffer
- EAP high band mono buffer

If the signal is in STEREO, the Crossover output will be composed of 4 buffers:

- EAP low band right channel buffer
- EAP low band left channel buffer
- EAP high band right channel buffer
- EAP high band left channel buffer.

Due to hardware limitations, we only have 2 physical output wires on our NXP SDK evaluation kit. We can outputted 2 buffers.

The following table explain the EAP output management for MONO & STEREO when Crossover is enable and disable.

Crossover module	Mono/Stereo	Physical output 1	Physical output 2
DISABLE	MONO	EAP mono buffer	Not used
DISABLE	STEREO	EAP right channel buffer	EAP left channel buffer
ENABLE	MONO	EAP low band mono buffer	EAP high band mono buffer
ENABLE	STEREO	EAP low band right channel buffer	EAP high band right channel buffer

For the last case , **EAP low band left channel buffer** & **EAP high band left channel buffer** are not outputted because of the hardware limitations.

ABBREVIATIONS AND REFERENCES

Abbreviations	
API	Application Programmers Interface
AVL	Auto Volume Leveler
BE	Bass Enhancement, either PureBass or DBE whichever is included in the bundle release
Block Size	Equal Frame Size
Buffer Size	The size of a buffer in Bytes. For a mono stream this the Block Size times the size of one sample in Bytes, for a stereo stream this is twice the Block Size times the size of one sample in Bytes.
CS	ConcertSound, 3D widening
DBE	Dynamic Bass Enhancement
dBFS	dB relative to full scale signal
EQNB	N-Band Equalizer
Frame Duration	The duration of a buffer of samples in seconds. This is given by the Frame Size divided by the Sample Rate.
Frame Size	The number of samples per channel to be processed in one call to the LVM_Process function.
Inplace	The name for processing data where the input and output buffers are at the same physical address in memory
Interleaved	The arrangement of samples in memory where the samples are alternately for the Left channel and the Right channel
LIMP	Peak limiter
LIMR	RMS limiter
LM	Loudness Maximiser
MIPS	Million Instructions Per Seconds
Non-Interleaved	The arrangement of samples in memory where the samples for each channel follow one another,
Nyquist	Half the sample rate
Outplace	The name for processing data where the input and output buffers are at different physical addresses in memory
PB	PureBass
PSA	Parametric Spectrum Analyzer.
Sample Rate	The number of samples per second.
TE	Treble Enhancement
TG	Tone Generator
VC	Volume control
XO	Crossover

References	
EAP user guide	*EAP_UserGuide.pdf*