



Essential Audio Processing

Users Guide

Released 4.0 - 2021-Sep-28

PUBLIC

Document information

Title	Essential Audio Processing Users Guide
--------------	---

Status	Released
---------------	----------

Version	4.0
----------------	-----

Date	21-sept-2021
-------------	--------------

Document ID	
--------------------	--

Security	PUBLIC
-----------------	--------

Usage	NXP
--------------	-----

TABLE OF CONTENT

1	DOCUMENT DESCRIPTION	5
1.1	Purpose	5
1.2	Audio Tuning	5
1.3	Terminology	6
2	EAP DESCRIPTION.....	7
2.1	General	7
2.1.1	Overview.....	7
2.1.2	Operating mode.....	9
2.1.3	Sample rate	10
2.1.4	EAP Libraries.....	10
2.1.5	Feature Pack	11
2.1.6	High sample rate	11
2.1.7	Bit Depth.....	12
2.1.8	Input/Output format	12
2.1.9	Output device	14
2.1.10	Block size	15
2.1.11	NXP devices software protection	15
2.2	Tone Generator.....	16
2.2.1	Description.....	16
2.2.2	Definition C code	19
2.3	Parametric Equalizer.....	20
2.3.1	Description.....	20
2.3.2	Definition C code	24
2.4	3D Widening	25
2.4.1	Description.....	25
2.4.2	Definition C code	25
2.5	Volume Control	26
2.5.1	Description.....	26
2.5.2	Definition C code	26
2.6	Bass Enhancement.....	27
2.6.1	Description.....	27
2.6.2	Definition C code	27
2.7	Audio Volume Leveler.....	28
2.7.1	Description.....	28
2.7.2	Definition C code	28
2.8	Loudness Maximiser	29
2.8.1	Description.....	29
2.8.2	Definition C code	29
2.9	Treble Enhancement.....	31
2.9.1	Description.....	31
2.9.2	Definition C code	31
2.10	Peak Limiter	32
2.10.1	Description.....	32
2.10.2	Definition C code	32
2.11	RMS Limiter	33
2.11.1	Description.....	33

Users Guide

2.11.2	Definition C code	33
2.12	Parametric Spectrum Analyzer	34
2.12.1	Description.....	34
2.12.2	Definition C code	34
2.13	Crossover 2-Bands	35
2.13.1	Description.....	35
2.13.2	Definition C code	36
2.14	Headroom Management	37
2.14.1	Headroom computation:	37
2.14.2	API.....	37
2.14.3	Operating Mode.....	37
2.14.4	Band definition.....	38
2.14.5	Configuration example	38
2.14.6	Conclusion.....	40
3	EAP INTEGRATION.....	41
3.1	Sequence & Description	41
3.2	Special Functions.....	43
3.2.1	LVM_GetVersionInfo	43
3.2.2	LVM_ClearAudioBuffers.....	43
3.2.3	LVM_GetAVLGain	43
3.2.4	LVM_SetHeadroomParams	43
3.2.5	LVM_GetHeadroomParams	43
3.2.6	LVM_GetSpectrum	43
3.2.7	LVM_SetVolumeNoSmoothing.....	43
3.3	Memory Placement	44
4	MIPS & MEMORY.....	45

1 DOCUMENT DESCRIPTION

1.1 Purpose

This document provides the required information to understand, deploy and tune the Essential Audio Processing library (EAP).

For each processing block, this document 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 of EAP into an existing application.

1.2 Audio Tuning

Audio tuning is an important aspect of application development and must be considered. It permits to adapt the algorithm to your audio chain and your speaker including casing speaker.

EAP is provided with the Audio Tuning Tool to exercise the algorithms. However, you need to plan tuning sessions with the actual hardware.

Classic audio tuning is simple and it is possible to produce good results with little audio processing knowledge and NXP support.

If expert audio tuning is required to reach better acoustic experience, then you will need:

- An audio laboratory to be able to perform dedicated measure.
- An acoustic engineer to understand and reproduce the tuning procedure.

NXP can organize tuning session with your device in its certified audio laboratory, shown in Figure 1, with an acoustic engineer who supports the EAP block.

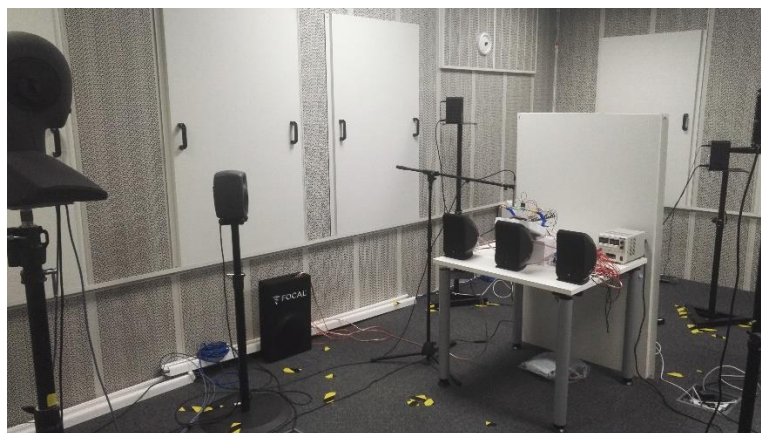
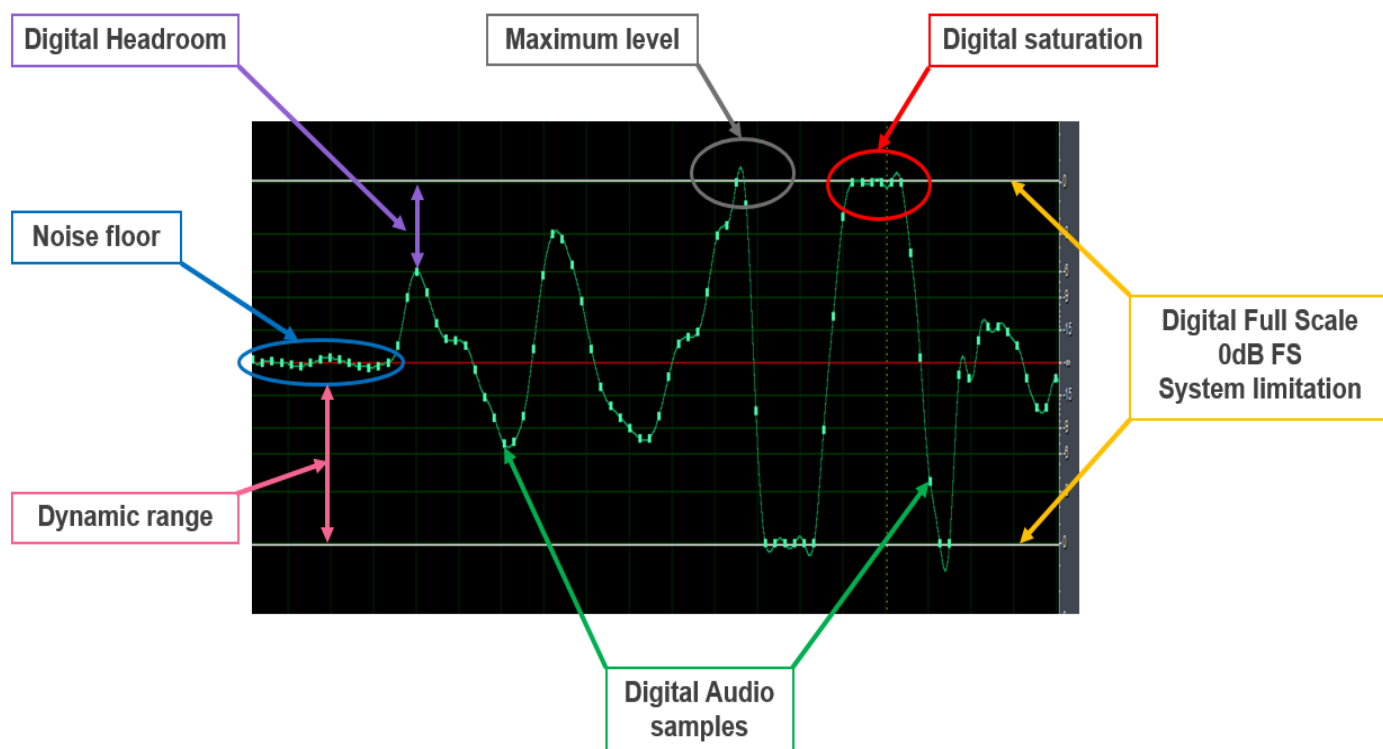


Figure 1 – NXP Audio Laboratory

1.3 Terminology

Below is an overview of several terms that are used throughout this document.



2 EAP DESCRIPTION

This chapter includes audio block descriptions and tuning information.

2.1 General

2.1.1 Overview

The EAP software is a bundle of audio processing blocks which can be ordered as required at library compilation time. Then the EAP software can be placed anywhere in the audio chain after the audio decoder and before the output driver.

EAP supports mono or stereo raw audio data in 16bits, 24bits, 32bits at multiple sample rate 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100 48000 or 96000kHz.

EAP includes the following sound processing algorithms:

- 3D Virtualization: ConcertSound
- Speaker Equalizer
- User Equalizer
- Bass Enhancement (Pure Bass or Digital Bass Enhancement)
- Volume Control
- Treble Enhancement
- Loudness Maximiser
- Auto Volume Leveler
- Tone Generator
- Peak Limiter
- RMS Limiter
- Power Spectrum Analyzer
- Crossover 2-bands

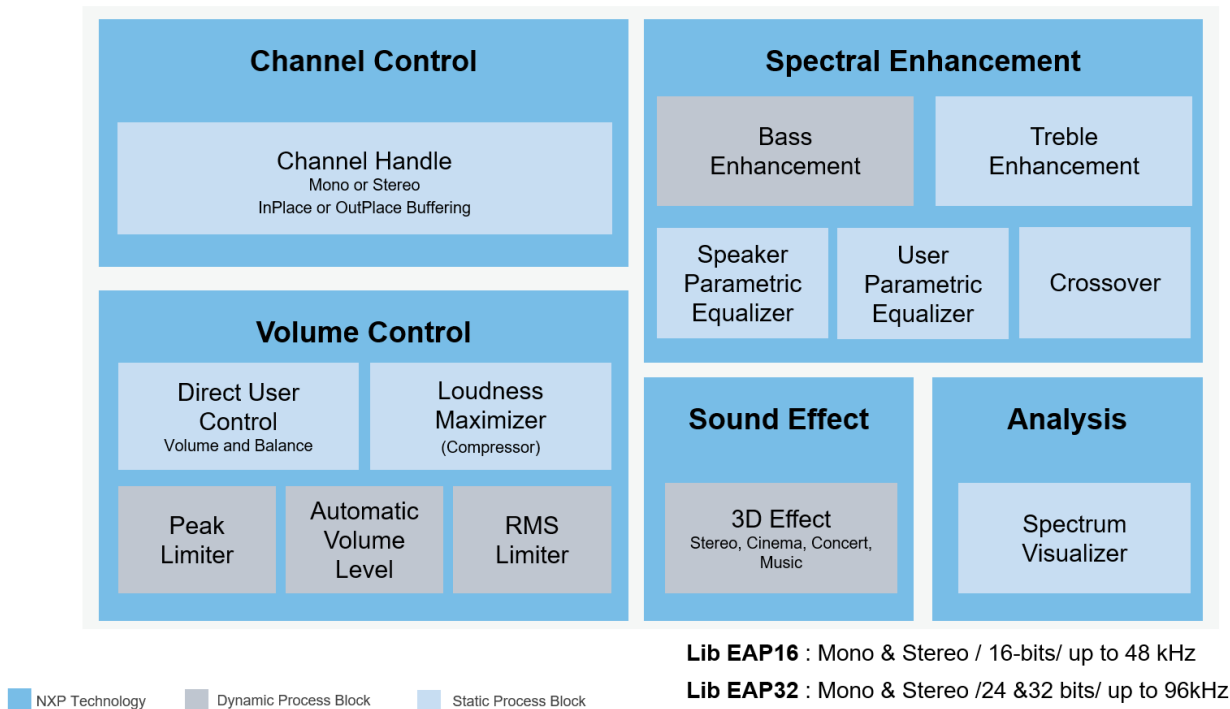


Figure 2 - EAP software block diagram

The process can be performed in-place (input and output buffer are same) or with separate input and output buffer.

Parameters update can happen at any time. EAP will save them and apply them.

This combination of audio features results in an impressive effect that enhances the tonal perception of the sound and improves the ‘spatialness’ of the audio, resulting in an enjoyable and relaxing listening experience.

Users Guide

EAP audio blocks can be re-ordered if required (please contact NXP for new library generation). The below figure shows the default and recommended audio chain order.

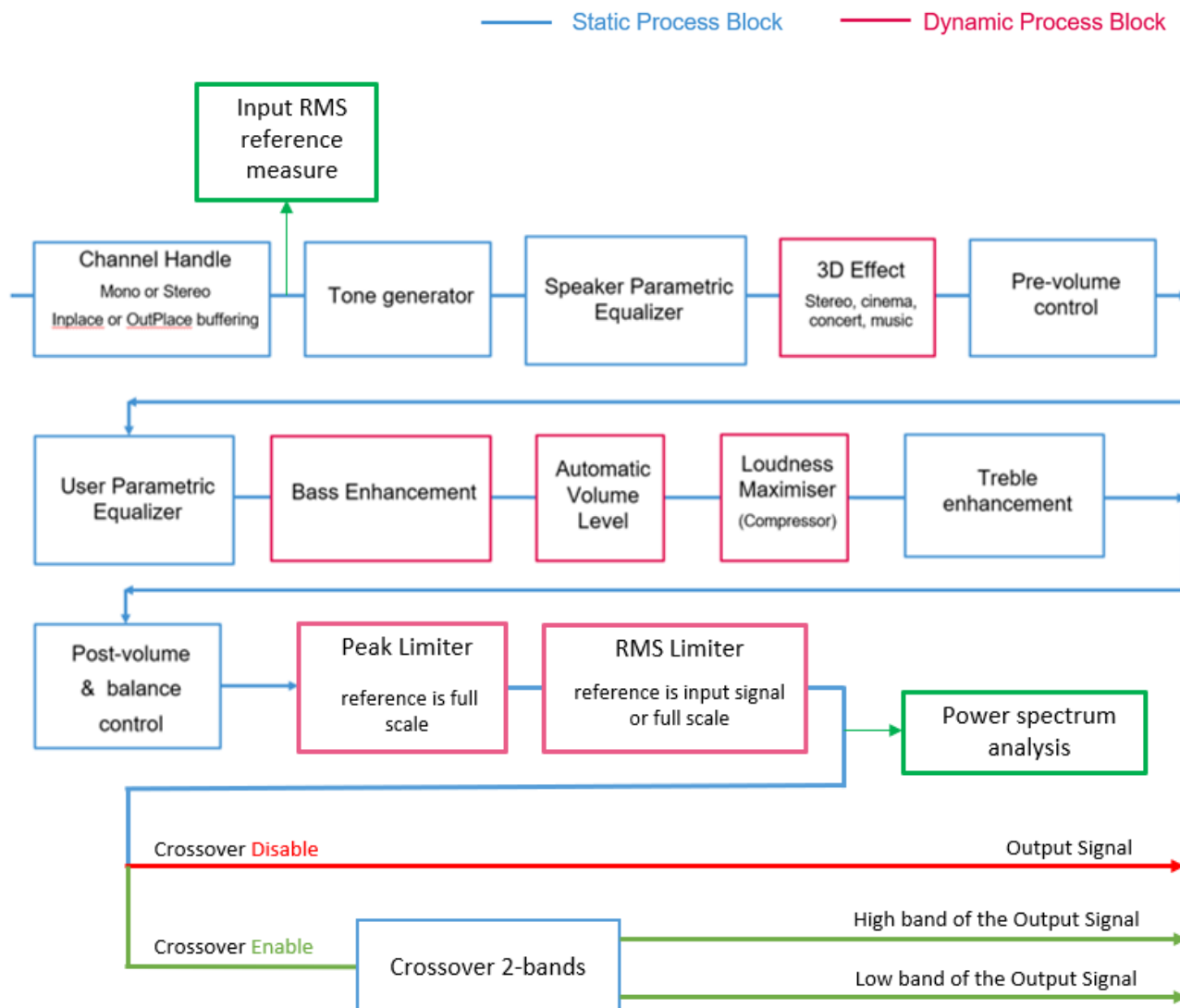


Figure 3 - EAP default audio chain order

2.1.2 Operating mode

The entire EAP audio chain or individual EAP blocks can be activated or bypassed through operating Mode parameters. Volume control and balance are always enabled.

2.1.2.1 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

LVM_Mode_en	OperatingMode;	/* Bundle operating mode */
LVM_Mode_en	VirtualizerOperatingMode;	/* Virtualizer operating mode */
LVM_EQNB_Mode_en	EQNB_OperatingMode;	/* N-Band Equaliser operating mode */
LVM_BE_Mode_en	BE_OperatingMode;	/* Bass Enhancement operating mode */

LVM_TE_Mode_en	TE_OperatingMode;	/* Treeble Enhancement operating mode */
LVM_AVL_Mode_en	AVL_OperatingMode;	/* AVL operating mode */
LVM_TG_Mode_en	TG_OperatingMode;	/* Tone generator operating mode */
LVM_PSA_Mode_en	PSA_Enable;	/* PSA mode operating mode */

See dedicated enum definition in LVM.h for correct syntax of each operating mode.

2.1.3 Sample rate

2.1.3.1 Description

It describes the sample rate of the input audio. It can be one of the following values:

- High sample rates : 96kHz (only supported by the EAP32 libraries)
- Normal sample rates: 32kHz, 44.1kHz, 48kHz,
- Half the sample rates: 16kHz, 22.05kHz and 24kHz,
- Quarter the sample rates: 8kHz, 11.025kHz and 12kHz.

2.1.3.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_Fs_en SampleRate    // LVM_FS_8000, LVM_FS_11025, LVM_FS_12000,
                        LVM_FS_16000, LVM_FS_22050, LVM_FS_24000,
                        LVM_FS_32000, LVM_FS_44100, LVM_FS_48000,
                        LVM_FS_96000
```

2.1.4 EAP Libraries

There is two EAP library available :

- *Lib_EAP16_x_y_z_FPw* is supporting up to 32 bits input audio sample and up to 96KHz sampling rate.
- *Lib_EAP32_x_y_z_FPw* is optimized to consume less MIPS and memory than the EAP 32 bits library. However EAP 16 library is limited to 16 bits input audio sample and 48KHz sampling rate.

x : reflects API changes;

y : reflects major changes;

z : reflects minor changes;

w : reflects the different features pack (see Feature Pack for further information)

x.y.z of 16 or 32 bits library are not linked.

2.1.5 Feature Pack

On each library name there is a number following “_FP”. It represents the feature pack compiled within the library. **If an algorithm is not compiled within the library, it cannot be usable.**

	Feature Pack 1: _FP1	Feature Pack 2: _FP2
Concert Sound	X	
Speaker Equalizer	X	X
User Equalizer	X	X
Bass Enhancement	X	X
Volume Control	X	X
Treble Enhancement	X	X
Loudness Maximiser	X	X
Auto Volume Leveler	X	X
Tone Generator	X	X
Peak Limiter	X	X
RMS Limiter	X	X
Power Spectrum Analyzer	X	X
Crossover 2-bands	X	

Table 1: Algorithm compiled within the library depending on the Feature Pack

2.1.6 High sample rate

For the high sample rate (sampling rate > 48KHz) the EAP block is handling the input audio signal in 2 frequency parts.

EAP is splitting the audio signal in a low band and a high band.

The low bands [0 to 24KHz] is processed as usual.

The high band [24kHz to 48kHz] is not processed by the bundle of algorithm.

It only :

- get a delay to stay synchronized with the low band
- receive a gain to keep same amplitude than the low band.

The following schematic show the processing parts.

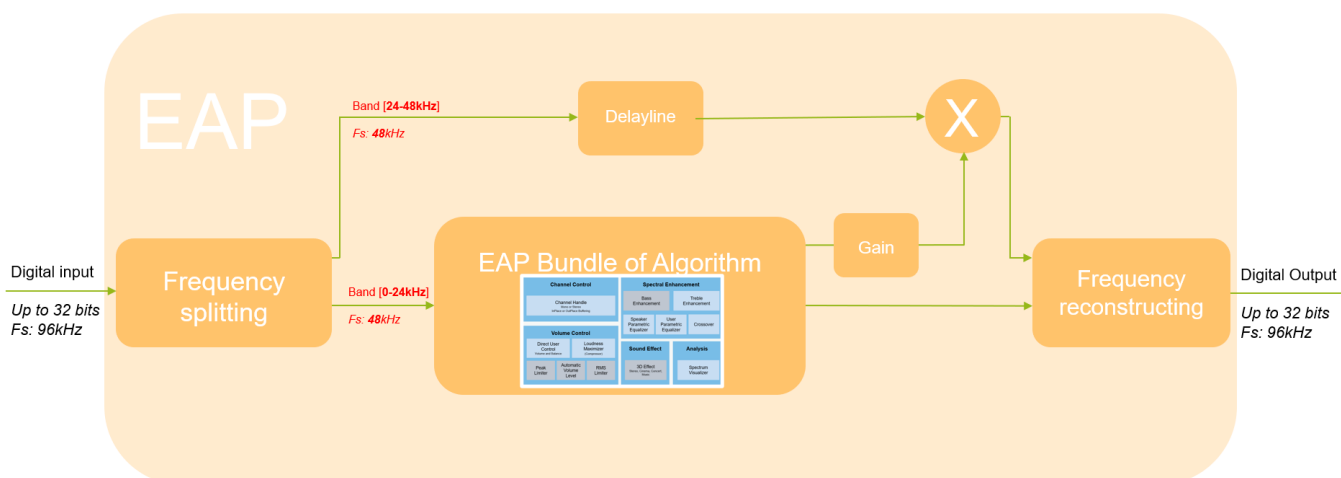


Figure 4: processing part of EAP for high sample rates

2.1.7 Bit Depth

Maximum bit depth value depends of the type of library (see EAP Libraries for further informations)

2.1.7.1 16 bits library

Input bit depth is maximum 16 bits. It can be lower but MSB must be align to 16 bit data MSB.

Output audio sample data will be 16 bits.

2.1.7.2 32 bits library

Input bit depth is maximum 32 bits. It can be lower (16,24,32 bits) but MSB must be align to 32 bit data MSB.

Output audio sample data will be 32 bits.

It is the user code responsibility to adapt audio sample input and output to the container used by the library (16 or 32 bits).

2.1.8 Input/Output format

2.1.8.1 Description

The LVM_Format_en enumerated type is used to set the value of the bundle data format.

The bundle supports input data in 2 formats: Mono or Stereo.

For an input buffer of sample number (N sample pairs for Stereo or N samples for Mono) the format of data in the In/Out buffer depends of your configuration. There is 3 cases :

2.1.8.1.1 Crossover Disable

Sample Number	Input in stereo	Input in mono
0	Left(0)	Mono(0)
1	Right(0)	Mono(1)
2	Left(1)	Mono(2)
3	Right(1)	Mono(3)
4	Left(2)	Mono(4)
"	"	"
"	"	"
N-2	Left(N/2-1)	Mono(N-2)
N-1	Right(N/2-1)	Mono(N-1)
N	Left(N/2)	Not Used
N+1	Right(N/2)	Not Used
N+2	Left(N/2+1)	Not Used
N+3	Right(N/2+1)	Not Used
"	"	Not Used
"	"	Not Used
2*N-2	Left(N-1)	Not Used

2*N-1	Right(N-1)	Not Used
-------	------------	----------

Table 2: Audio buffer format when Crossover is Disable

The output format is the same than the input format.

2.1.8.1.2 Crossover Enable & input/output in MONO

MONO			
Sample Number	Input buffer	Output buffer LB <i>pOutData[0]</i>	Output buffer HB <i>pOutData[1]</i>
0	Mono(0)	Mono LB(0)	Mono HB(0)
1	Mono(1)	Mono LB(1)	Mono HB(1)
2	Mono(2)	Mono LB(2)	Mono HB(2)
3	Mono(3)	Mono LB(3)	Mono HB(3)
4	Mono(4)	Mono LB(4)	Mono HB(4)
"	"	"	"
"	"	"	"
N-2	Mono(N-2)	Mono LB (N-2)	Mono HB (N-2)
N-1	Mono(N-1)	Mono LB (N-1)	Mono HB (N-1)

Table 3 : Audio buffer format when Crossover Enable & input/output are in mono

LB : Low band ; HB : High band (see 2.13 for further informations)

2.1.8.1.3 Crossover Enable & input/output in STEREO

STEREO			
Sample Number	Input buffer	Output buffer LB <i>pOutData[0]</i>	Output buffer HB <i>pOutData[1]</i>
0	Left(0)	Left LB(0)	Left HB(0)
1	Right(0)	Right LB(0)	Right HB(0)
2	Left(1)	Left LB(1)	Left HB(1)
3	Right(1)	Right LB(1)	Right HB(1)
4	Left(2)	Left LB(2)	Left HB(2)
"	"	"	"
"	"	"	"
N-2	Left(N/2-1)	Left LB (N/2-1)	Left HB (N/2-1)
N-1	Right(N/2-1)	Right LB (N/2-1)	Right HB (N/2-1)
N	Left(N/2)	Left LB(N/2)	Left HB(N/2)
N+1	Right(N/2)	Right LB(N/2)	Right HB(N/2)
N+2	Left(N/2+1)	Left LB (N/2+1)	Left HB (N/2+1)
N+3	Right(N/2+1)	Right LB (N/2+1)	Right HB (N/2+1)
"	"	"	"

"	"	"	"
2*N-2	Left(N-1)	Left LB (N-1)	Left HB (N-1)
2*N-1	Right(N-1)	Right LB (N-1)	Right HB (N-1)

Table 4: Audio buffer format when Crossover Enable & input/output are in Stereo

LB : Low band ; HB : High band (see 2.13 for further informations)

2.1.8.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_Format_en SourceFormat; // Input data format LVM_STEREO, LVM_MONO
```

LVM_MONOINSTEREO permits to read a mono file and output a stereo files.

2.1.9 Output device

2.1.9.1 Description

The output device may be headphones or speakers. If it is a speaker, 3 speaker types (small, medium or large) can be selected. Speaker type is only used by the 3D Virtualizer if the processing block is enabled.

2.1.9.2 Speaker type identification

Table 5: Speaker Types			
	Small Speaker	Medium Speaker	Large Speaker
Low 3dB Frequency	Above 1000Hz	500Hz to 1000Hz	Below 500Hz

The low 3dB frequency can be estimated by looking at the frequency response of the loudspeaker when mounted in the target device and finding the frequency at which the output is approximately 3dB below the plateau. This frequency may be different from that measured on the speaker when it is not mounted in the target device.

2.1.9.3 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_OutputDeviceType_en SpeakerType; // Output device type: LVM_HEADPHONES,
LVM_MOBILE_SPEAKERS_SMALL,
LVM_MOBILE_SPEAKERS_MEDIUM,
LVM_MOBILE_SPEAKERS_LARGE
```

2.1.10 Block size

Block size is the number of samples which will be process when EAP process function is called. We advise to use fix block size which represent a 10ms or 20ms audio buffer.

Example: For Sampling rate = 44.1KHz, block size is 441 sample for a 10ms audio buffer or 882 samples for a 20ms buffer.

Input format as mono or stereo doesn't affect this value.

If you choose 96kHz as sample rate, **blocksize must be even**.

2.1.10.1 Definition C code

Block size is not a part of the EAP tuning parameters. However, the application must provide the block size in number of samples when the EAP process function is called.

2.1.11 NXP devices software protection

A software protection is used to ensure EAP library can be used only on an NXP device.

To successfully pass the test:

- 1- The *Platform* parameter of the *LVM_InstParams_t* structure must be set to the correct device used.
- 2- The synchronous audio interface transmitter (SAI TX) module must be enabled when *LVM_getInstanceHandle* function is called during the EAP initialization phase.

If software protection test failed, then the *LVM_getInstanceHandle* function return *LVM_INVALIDNXPPLATFORM*.

If SAI TX is no more needed by the software then it can be disable after the *LVM_getInstanceHandle()* call.

2.1.11.1 Definition in C code

LVM_InstParams_t structure parameter into LVM.h:

```
EAP_NXPPlatform_en Platform; /* NXP Platform where EAP is playing on
                               (LVM_IMXRT1050,LVM_IMXRT1060,
                               LVM_IMXRT1064, LVM_IMXRT1170, LVM_LPC55,
                               LVM_IMXRT500, LVM_IMXRT600)*/
```

2.2 Tone Generator

2.2.1 Description

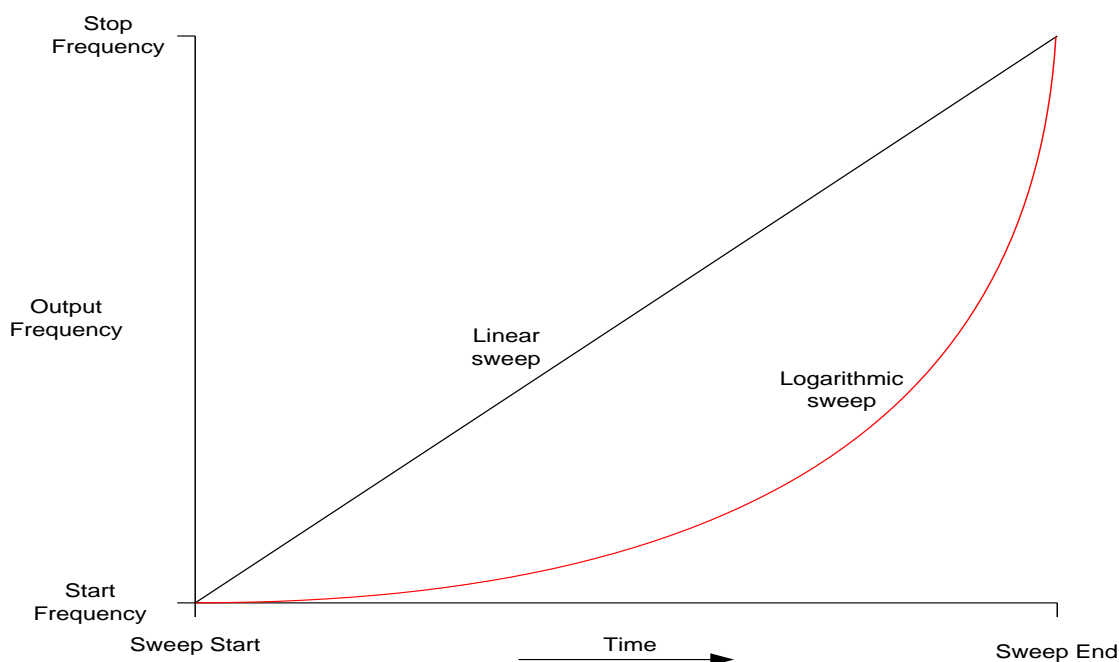
The tone generator can be configured to create fixed frequency tones and a variety of frequency and amplitude sweeps. This algorithm is made available on request while compiling the library, for use during development, to help tuning the music algorithms and to aid measurement of the system audio output quality.

Figure 5:EAP process diagram in case of 96kHz as sample rate

The tone generator module is not used during music playback and is not intended to be included as part of the final release version of the library.

The tone generator may be used during tuning to help measuring the audio quality of the system under development. The tone is generated inside the bundle before any of the other algorithms are applied. In general, all other algorithms should be disabled when the tone generator is in use.

The sweep can be either linear or logarithmic. If logarithmic sweep is selected the rate of frequency update is based on the ratio of the stop and start frequencies. The following graph shows the linear sweep output and a logarithmic sweep output for a high ratio of stop frequency to start frequency. Linear sweep should be used when the ratio of the start and stop frequencies is close to unity.



The use of the generator is best shown by a series of examples:

Continuous Fixed tone

To generate a continuous fixed tone the following parameters should be set:


```
Params.TG_OperatingMode = LVM_TG_CONTINUOUS;  
Params.TG_SweepMode     = LVM_TG_SWEEPLIN;  
Params.TG_StartFrequency = RequiredFrequency;  
Params.TG_StartAmplitude = RequiredAmplitude;  
Params.TG_StopFrequency  = Params.TG_StartFrequency;  
Params.TG_StopAmplitude  = Params.TG_StartAmplitude;  
Params.TG_SweepDuration  = 0;  
Params.pTG_Callback      = LVM_NULL;    /* No callback */
```

Where:

RequiredFrequency is the tone frequency in Hz

RequiredAmplitude is the tone output level in dBr, i.e. relative to the maximum peak signal level. 0dBr is the maximum amplitude.

No callback function can be used in this mode as the tone never ends.

Fixed tone for a set duration

To generate a fixed tone for a set duration the following parameters should be set:

```
Params.TG_OperatingMode = LVM_TG_ONESHOT;  
Params.TG_SweepMode     = LVM_TG_SWEEPLIN;  
Params.TG_StartFrequency = RequiredFrequency;  
Params.TG_StartAmplitude = RequiredAmplitude;  
Params.TG_StopFrequency  = Params.TG_StartFrequency;  
Params.TG_StopAmplitude  = Params.TG_StartAmplitude;  
Params.TG_SweepDuration  = RequiredDuration;  
Params.pTG_Callback      = LVM_NULL;    /* No callback */
```

Where:

RequiredFrequency is the tone frequency in Hz

RequiredAmplitude is the tone output level in dBr, i.e. relative to the maximum peak signal level. 0dBr is the maximum amplitude.

RequiredDuration is the duration of the tone in seconds

At the end of the tone the Tone Generator will be automatically placed in LVM_TG_OFF mode and its output disabled. A callback function can be used; this will be called at the end of the tone.

Frequency Sweep

To generate a frequency sweep, the following parameters should be set:

```
Params.TG_OperatingMode = LVM_TG_ONESHOT;  
Params.TG_SweepMode     = LVM_TG_SWEEPLOG;  
Params.TG_StartFrequency = RequiredStartFrequency;  
Params.TG_StartAmplitude = RequiredAmplitude;  
Params.TG_StopFrequency  = RequiredStopFrequency;  
Params.TG_StopAmplitude  = Params.TG_StartAmplitude;  
Params.TG_SweepDuration  = RequiredDuration;  
Params.pTG_Callback      = LVM_NULL;    /* No callback */
```

Where:

RequiredStartFrequency	is the initial tone frequency in Hz
RequiredAmplitude	is the tone output level in dBr, i.e. relative to the maximum peak signal level. 0dBr is the maximum amplitude.
RequiredStopFrequency	is the final tone frequency in Hz
RequiredDuration	is the duration of the tone in seconds

At the end of the tone the Tone Generator will be automatically placed in LVM_TG_OFF mode and its output disabled. A callback function can be used; this will be called at the end of the tone.

Amplitude Sweep

To generate an amplitude sweep, the following parameters should be set:

```
Params.TG_OperatingMode = LVM_TG_ONESHOT;  
Params.TG_SweepMode     = LVM_TG_SWEEPLIN;  
Params.TG_StartFrequency = RequiredFrequency;  
Params.TG_StartAmplitude = RequiredStartAmplitude;  
Params.TG_StopFrequency  = Params.TG_StartFrequency;  
Params.TG_StopAmplitude  = RequiredStopAmplitude;  
Params.TG_SweepDuration  = RequiredDuration;  
Params.pTG_CallBack      = LVM_NULL;    /* No callback */
```

Where:

RequiredFrequency	is the tone frequency in Hz
RequiredStartAmplitude	is the initial tone output level in dBr, i.e. relative to the maximum peak signal level. 0dBr is the maximum amplitude.
RequiredStopAmplitude	is the final tone output level in dBr
RequiredDuration	is the duration of the tone in seconds

At the end of the tone the Tone Generator will be automatically placed in LVM_TG_OFF mode and its output disabled. A callback function can be used; this will be called at the end of the tone.

Repeating Amplitude Sweep

To generate a fixed tone the following parameters should be set:

```
Params.TG_OperatingMode = LVM_TG_CONTINUOUS;  
Params.TG_SweepMode     = LVM_TG_SWEEPLIN;  
Params.TG_StartFrequency = RequiredFrequency;  
Params.TG_StartAmplitude = RequiredStartAmplitude;  
Params.TG_StopFrequency  = Params.TG_StartFrequency;  
Params.TG_StopAmplitude  = RequiredStopAmplitude;  
Params.TG_SweepDuration  = RequiredDuration;  
Params.pTG_CallBack      = LVM_NULL;    /* No callback */
```

Where:

RequiredFrequency	is the tone frequency in Hz
RequiredStartAmplitude	is the initial tone output level in dBr, i.e. relative to the maximum peak signal level. 0dBr is the maximum amplitude.

Users Guide

RequiredStopAmplitude	is the final tone output level in dBr
RequiredDuration	is the duration of the tone in seconds

No callback function can be used in this mode as the tone never ends.

2.2.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_TG_Mode_en TG_OperatingMode;    // LVM_TG_OFF, LVM_TG_CONTINUOUS, LVM_TG_ONESHOT
LVM_TG_SweepMode_en TG_SweepMode;    // LVM_TG_SWEEPLIN or LVM_TG_SWEEPLOG
LVM_UINT16 TG_StartFrequency;         // Tone Generator Sweep Start Frequency in Hz
LVM_INT16 TG_StartAmplitude;         // Tone Generator Sweep Start Amplitude in dB
LVM_UINT16 TG_StopFrequency;         // Tone Generator Sweep Stop Frequency in Hz
LVM_INT16 TG_StopAmplitude           // Tone Generator Sweep Stop Amplitude in dB
LVM_UINT16 TG_SweepDuration;         // Tone Generator Sweep Duration; Sweep duration in seconds, 0 for
                                     // infinite duration tone
LVM_Callback TG_CallBack;            // End of sweep callback
LVM_INT16 TG_CallBackID;             // Callback ID
void *pTGAppMemSpace;               // Application instance handle or memory area
```

2.3 Parametric Equalizer

2.3.1 Description

This audio process is a Nband equalizer plus a low & high pass filter.

Two parametric equalizers are present in the bundle:

The User parametric equalizer:

This equalizer is used to provide frequency effect like voice enhancement, bass boost, pop or others.

The Speaker parametric equalizer:

This equalizer is used to control the speaker or headphone output can be adjusted through equalization. This allows imperfections in the headphone or speaker and housing design of the device to be reduced.

This gives a better overall sound quality from the device.

This audio process is a Nband equalizer plus a high pass filter.

2.3.1.1 N Bands

The N-Band equalizer algorithm can be used to provide signal equalization using up to 15 bands.

Each band is defined by :

- Its center frequency, controlled in 1Hz step.
- The gain, controlled over the range –15dB to +15dB in 1dB steps for **User parametric equalizer**.
- The gain, controlled over the range –15dB to +3dB in 1dB steps for **Speaker parametric equalizer**. It is advice to not set positive gain to avoid any digital saturation in the band.
- The Q factor, controlled over the range 0.25 to 12.00 in steps of 0.01.

Note 1: Why a different gain range?

- User parametric equalizer works with headroom management block to avoid saturations. It controls the maximal allowed volume control settings to guarantee an acceptable risk of saturations over the band definition. With this approach it is allowed to set band gain superior to 0 dB.
- Speaker parametric equalizer doesn't get any mechanism to avoid saturation. This is why it is recommended to stay below 0dB gain. +3dB gain per band can be partially set in a limited band frequency with acceptable risk of saturations.

Each band has 3 parameters:

Gain	the gain of the band in 1dB step. If the gain is set to 0dB the band will be disabled
Frequency	the frequency is in Hz, from 20Hz to Nyquist frequency (half the sample rate)
Q-factor	the filter Q, from 0.25 to 12.00 in steps of 01.

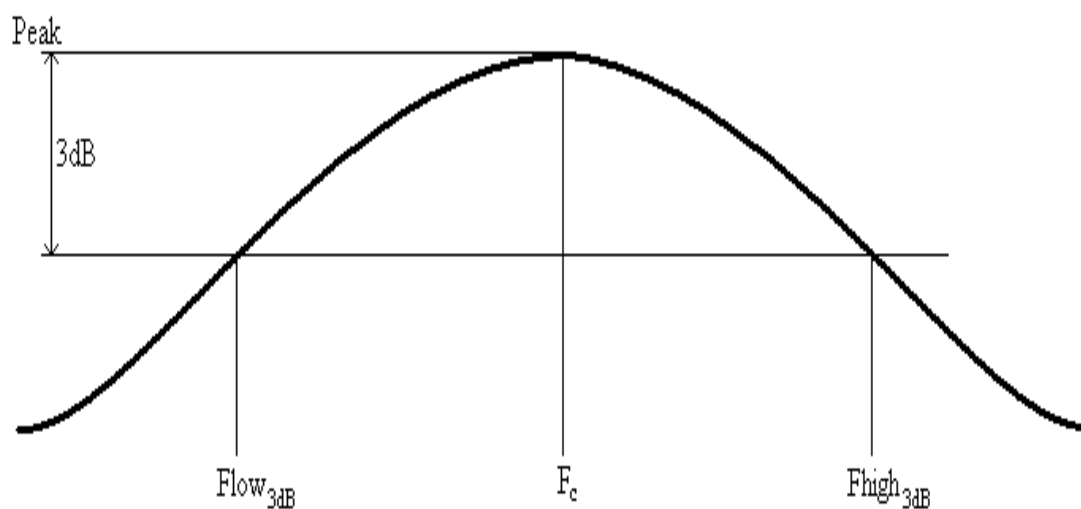
WARNING: The desired Q value is the parameter value divided by 100. So a Q of 0.25 is given by the value 25 The Q-factor is defined by the following equation:

$$Q\text{-factor} = F_c / (F_{\text{high}3\text{dB}} - F_{\text{low}3\text{dB}})$$

Where:

Users Guide

- F_c - Filter center frequency (the Frequency setting)
- $F_{low_{3dB}}$ - The lower 3dB cut-off frequency of the filter
- $F_{high_{3dB}}$ - The upper 3dB cut-off frequency of the filter



The maximum number of bands to be used can be define:

```
#define MAX_BANDS 15
```

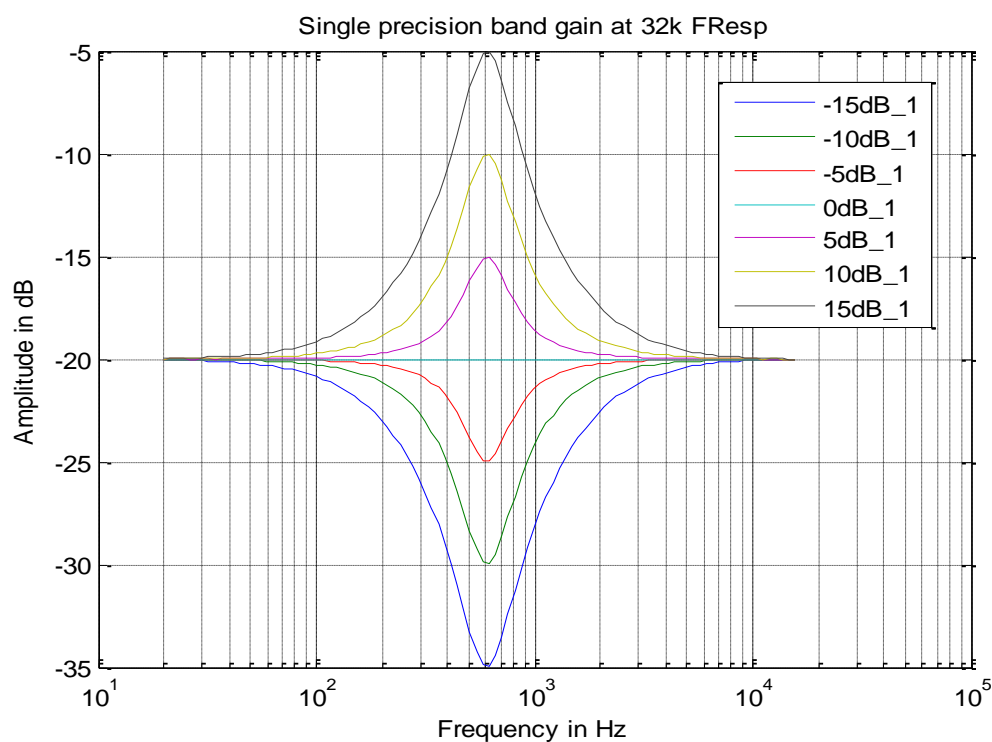
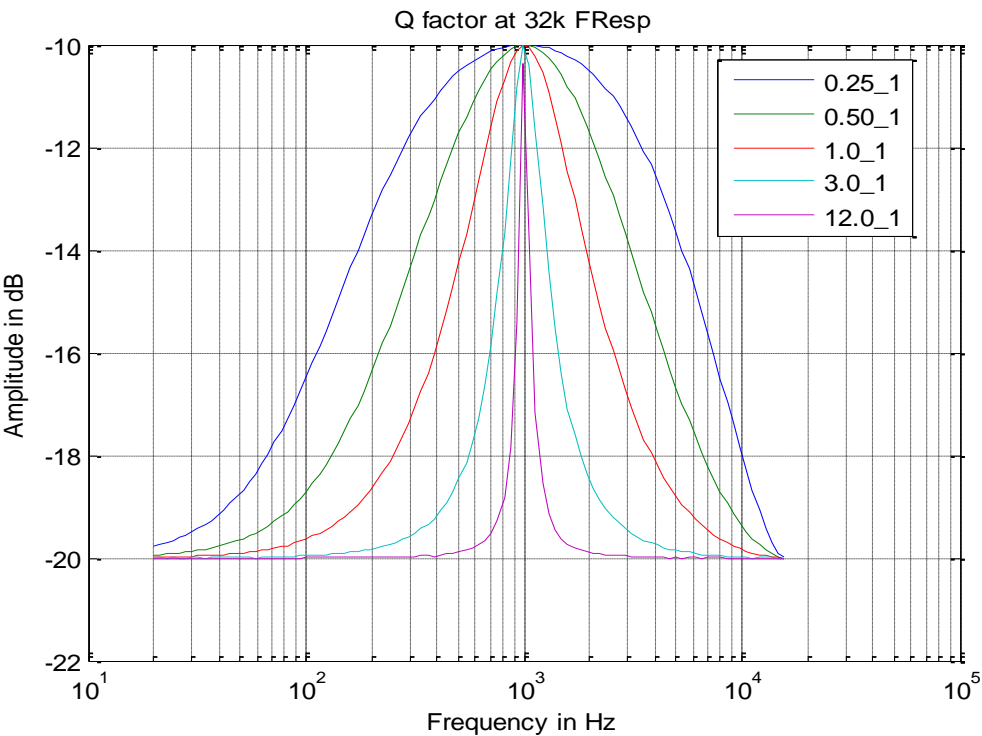


Figure 6 – Equalizer filter gain settings



The filters used in the equalizer include a user defined center or corner frequency. If this frequency is greater than Nyquist (half the sample rate) then the filter will be automatically disabled. Hence the filter definitions do not need to be modified when changing the sample rate.

2.3.1.2 High Pass and Low Pass Filters

The equalizer also includes two additional filters for low pass and high pass filtering. Each filter has a separate On/Off control and user defined corner frequency. The filters have the following characteristics:

	High Pass Filter	Low Pass Filter
Minimum Frequency	20Hz	1Hz
Maximum Frequency	1Hz	Nyquist
Order	2 nd order	1 st order
dB/Octave	12dB	6dB

If the corner frequency of the high pass filter is above Nyquist then the filter will be automatically disabled in the equalizer.

2.3.1.3 Equalizer standart effect

These following table regroup an example of standard user parametric equalizer setting for dedicated effect:

Band	Frequency	Q Factor	Gain (dB)
1	50Hz	0.96	See gain table
2	205Hz	0.96	See gain table
3	837Hz	0.96	See gain table
4	3427Hz	0.96	See gain table
5	14027Hz	0.96	See gain table

Gain Table

Preset Name	Band				
	#1	#2	#3	#4	#5
Normal	3	0	0	0	3
Bass Booster	6	3	1	0	0
Classical	5	3	-2	4	4
Dance	8	2	4	6	3
Flat	0	0	0	0	0
Folk	6	3	3	5	2
Heavy Metal	4	1	9	3	0
Hip Hop	5	3	0	1	3
Jazz	4	2	-2	2	5
Piano	3	2	3	5	4
Pop	-1	2	5	1	-2
Rock	5	3	-1	3	5
Spoken Word	-2	2	5	5	2
Symphony	7	0	-2	-4	3
Theater	3	0	5	-1	2
Treble Booster	0	0	2	4	6
Latin	4	0	-2	1	5
Vocal Booster	-4	2	5	3	-1
Bass Reducer	-6	-4	0	0	0
Treble Reducer	0	0	-1	-4	-7

Table 6 Equalizer effect coefficient for 5 band filter

2.3.1.4 Working with other algorithms:

When Bass Enhancement or Treble Enhancement audio block are used, it provides superior performance, but we need to adapt the coefficient as follow:

In the example describe in Table 6 (see above).

When the bass enhancement is enabled at the same time of the equalizer:

- The low frequency band (band 1) in the equalizer should not be used to provide boost.
- The other frequencies remain same

When the treble enhancement is enabled at the same time of the equalizer:

- The high frequency band (band 5) in the equalizer should not be used to provide boost.
- The other frequencies remain same

2.3.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

The band definitions are held in an array:

```
LVM_EQNBBandDef_t BandDefs [MAX_BANDS]; /* Band definitions */
```

```
/* N-Band Equaliser band definition */
```

```
typedef struct
```

```
{
```

```
    LVM_INT16      Gain;
```

```
    LVM_UINT16     Frequency;
```

```
    LVM_UINT16     QFactor;
```

```
} LVM_EQNB_BandDef_t;
```

```
LVM_EQNB_FilterMode_en EQNB_LPF_Mode;      // Low pass filter ON/OFF
LVM_INT16               EQNB_LPF_CornerFreq; // low pass frequency
LVM_EQNB_FilterMode_en EQNB_HPF_Mode;      // High pass filter ON/OFF
LVM_INT16               EQNB_HPF_CornerFreq; // High pass frequency
```


2.4 3D Widening

2.4.1 Description

The ConcertSound audio effect increases the perceived distance and depth of the audio, providing an enhanced listening experience.

With headphones it offers highly effective sound enrichment. These algorithms reproduce audio with the frequency balance intended in the studio, creating a natural sound field outside the listener's head. It works on all types of music and film sound tracks in mono or stereo formats.

With closely spaced loudspeakers, it provides a sound with widened stereo (also sometimes called 3D widening). It works on music and film sound tracks and leaves the voice natural.

The Cinema and Concert sound algorithms digitally process the music signal on an audio device's sound processor using HRTF (Head-Related Transfer Function) positioning data. This accentuates inter-aural differences, increases the perceived distance and depth of the sound, and recreates natural cross talk between the ears. Such elements are lost with conventional headphone playback.

2.4.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_UINT16 VirtualizerReverbLevel; // reverberation level in % 0 no effect to 100 maximum effect
LVM_INT16 CS_EffectLevel; // Concert Sound effect level 0 no effect to 32767 maximum effect
```

Additional advanced parameter:

```
CS_AP_Mode = LVM_AP_DEFAULT, // concert sound advanced parameter mode:
                                     LVM_AP_DEFAULT or LVM_AP_MANUAL
```

Select LVM_AP_MANUAL to enable the following parameter else keep LVM_AP_DEFAULT.

```
CS_AP_MidGain          = 0, // MidChannelGain: -10 to 10 dB
CS_AP_MidCornerFreq    = 500, // Shelving Filter Corner Frequency: 20 to 24000 Hz
CS_AP_SideHighPassCutoff = 600, // SideBoost HighPassFilter Corner Frequency: 20 to 24000 Hz
CS_AP_SideLowPassCutoff = 1544, // SideBoost LowPassFilter Corner Frequency: 20 to 24000 Hz
CS_AP_SideGain         = 10, // Side Channel Gain: 0 to 15 dB
```

2.5 Volume Control

2.5.1 Description

The Volume Control is a permanent feature of the EAP solution.

The volume is adjustable from 0dB (full volume) to –96dB (silence) in 1dB steps. This is a soft volume control; it smoothly changes from one volume setting to another.

The volume control is split in 2 parts:

- Pre volume control is located at the front end and permits to create headroom for the following process.
- Post volume control is located at the back end of the chain to apply standard volume after the process.

The repartition of the volume in pre and post volume is automatic and follow this rule.

For general volume = 0dB to -15dB:

- Pre volume = General volume
- Post Volume = 0dB

For general volume = -15dB to -96dB:

- Pre volume = -15dB
- Post volume = general volume – 15 dB

2.5.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_INT16 VC_EffectLevel; // Volume setting in dBs (-96dB to 0dB)  
LVM_INT16 VC_Balance;    // Left Right balance in dB (-96 to 96 dB)
```

2.6 Bass Enhancement

2.6.1 Description

Two different bass enhancement algorithms can be delivered within the software bundle:

- Dynamic Bass Enhancement (DBE) – the algorithm exploits the acoustics system characteristics and aims to enhance the bass sensation in the target system by maximizing the bass enhancement within the available headroom.
- Pure Bass (PB) – this second bass enhancement option differs from the Dynamic Bass Enhancement in such a way that it can also deliver a bass enhancement for input signals at full range level. For this product implementation an unique patented NXP technology is used, that makes sure a deep and rich bass enhancement is delivered without any distortion to the audio and without the need to take any acoustical headroom on the input signal even with full scale input signals.

Warning: DBE or PB is chosen at library compilation time.

The Bass Enhancement is adjustable between 0dB to 15dB in 1dB steps.

Its center frequency can be adjusted in 4 different settings.

- LVM_BE_CENTER_55Hz for a center frequency of 55Hz,
- LVM_BE_CENTER_66Hz for a center frequency of 66Hz,
- LVM_BE_CENTER_78Hz for a center frequency of 78Hz,
- LVM_BE_CENTER_90Hz for a center frequency of 90Hz.

For high quality headphones, with good low frequency reproduction, all four settings may be used and the lowest, 55Hz is recommended.

For some headphone types the quality of reproduction at the lowest frequencies is not good enough to support the 55Hz center frequency. In these cases, another setting should be selected to give an optimum balance between bass response and audio quality.

The effect of changing the bass enhancement center frequency can be subjective but in general the lowest acceptable setting should be used.

A high pass filter is available to remove residual frequency lower than the center frequency. With Pure bass algorithm the HPF is always enable even if parameter is set to OFF.

2.6.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_BE_CentreFreq_en BE_CentreFreq // LVM_BE_CENTRE_55Hz, LVM_BE_CENTRE_66Hz,
                                     LVM_BE_CENTRE_78Hz, LVM_BE_CENTRE_90Hz
```

```
LVM_INT16 BE_EffectLevel; // 0 to 15 dB in 1dB steps
```

```
LVM_BE_FilterSelect_en BE_HPF; // high pass filter selector LVM_BE_HPF_OFF
                                     LVM_BE_HPF_ON
```

2.7 Audio Volume Leveler

2.7.1 Description

The volume level may require re-adjustment for each song when selecting music from a variety of sources and artists. This is because the volume setting defined for the previous song is either too low or too high for the next song.

The Auto Volume Leveler overcomes this problem in music players. It automatically adjusts the volume for each track to maintain the desired output volume level. It removes the need to adjust volume for each track.

Typical use cases are:

- Having a constant volume between different music track.
- Minimise the higher volume of advertisements.
- Increase volume of low-level soundtrack part. Some details may become audible

2.7.2 Definition C code

No parameters.

2.8 Loudness Maximiser

2.8.1 Description

The Loudness Maximiser significantly increases the perceived output volume for all types of music. It is intended for use when the output volume is already set to maximum.

It works with all speaker types from small to large but is particularly effective with small speakers where the maximum output volume is typically very low.

The Loudness Maximiser should only be used once the system volume control is already at its maximal undistorted level, it should be disabled otherwise.

With high level input signals there is still an output volume increase but at the expense of linearity.

If the processing before or after the bundle includes a compressor or other non-linear processing, the combination with the Loudness Maximiser can create some unexpected audio effects. Other compressor or non-linear processing blocks should be disabled when the Loudness Maximiser is enabled.

2.8.1.1 Effect level

Effect level can be:

- Gentle
- Medium
- Extreme

The non-linearities introduced:

- Are not generally obvious for Gentle and Medium effect levels
- Can be heard on some speaker types with Extreme mode.

2.8.1.2 Gain

It represents the target gain of the compressor.

Higher is the gain more the volume increase effect is perceptible but more non linearity are present.

2.8.1.3 Attenuation

If input data already get non linearity, adding the LM generates unacceptable distortion.

In these cases, the only solution is to reduce the general output gain of the LM thanks to attenuation parameter.

2.8.1.4 Speaker cutoff

It represents the frequency response low knee of the speaker.

If not available, then for initial tuning the following values can be used:

Speaker Size	High Pass Filter Frequency
Small	750Hz
Medium	500Hz
Large	250Hz or lower

2.8.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

Users Guide

```
LVM_LM_Effect_en LM_EffectLevel ; // LVM_LM_GENTLE or LVM_LM_MEDIUM or LVM_LM_EXTREME  
LVM_UINT16 LM_Attenuation;      // Output attenuation 0 to 6 dB  
LVM_UINT16 LM_CompressorGain;   // Target compressor gain 0 to 6 dB  
LVM_UINT16 LM_SpeakerCutOff;    // Device speaker cut off frequency 150Hz to 1100Hz
```

2.9 Treble Enhancement

2.9.1 Description

The Treble Enhancement algorithm adds more brilliance to the sound. It applies a filter to amplifies the higher audio frequencies.

Two operating modes are available:

- In the normal mode, an adjustable treble enhancement effect is used. This requires extra MIPS to operate. The amount of boost is set in 15 levels of 1 dB steps. The effect has a corner frequency of 8kHz and so no boost can be applied at sample rates of 16kHz or less.
- In the MIPS-saving mode, the bundle applies a fix curve going up to 6dB of treble enhancement without additional MIPS.

The following graph shows the frequency response of the treble boost at different gain settings for a signal of -20dB input level.

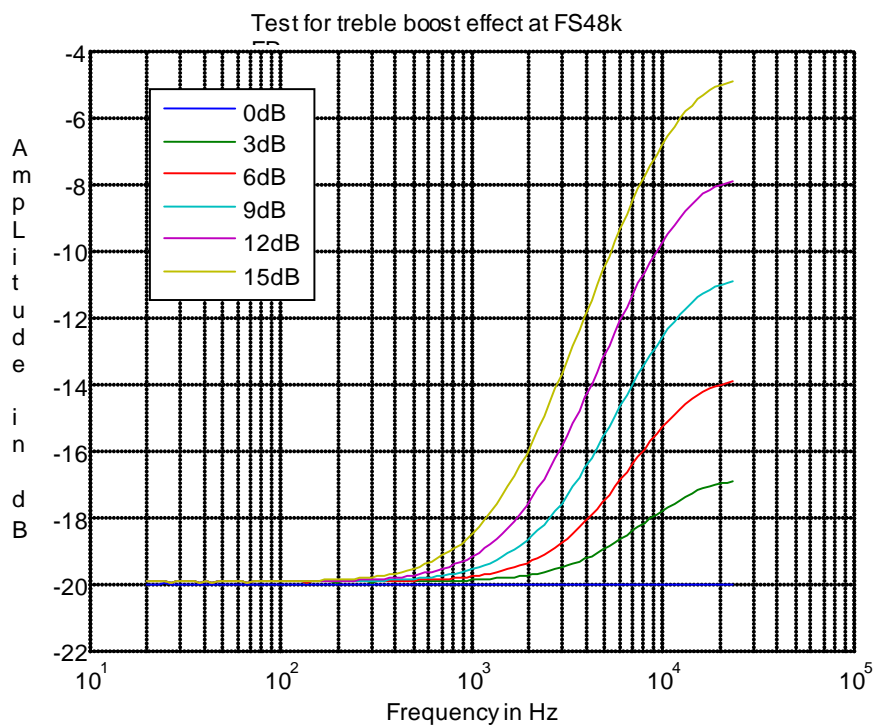


Figure 7 Treble Enhancement at different effect levels

2.9.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_INT16 TE_EffectLevel;
// Treble Enhancement gain in dB (0 to 15) or LVM_TE_LOW_MIPS for saving MIPS
```

2.10 Peak Limiter

2.10.1 Description

The peak limiter permits to limit peak amplitude of the processed signal.

It keeps audio signal amplitude below a threshold parameter value relative to 0dBFs. Algorithm include a look ahead buffer to guarantee a low distortion level at output.

It can be used for acoustic feeling purpose or material protection (ex: protection of small speaker).

2.10.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LIMP_Threshold = -2, // LIMP threshold in dB: -24dB to 0dB
```


2.11 RMS Limiter

2.11.1 Description

The RMS limiter permits to limit the average RMS value of the processed signal.

RMS value used by EAP is the average RMS power (FS square wave) with a window width equal to the block size.

It keeps RMS audio signal value below a threshold parameter relative to a reference. Reference can be 0dBFS or the input average RMS power of the signal.

It can be used for acoustic feeling purpose or material protection (ex: protection of large speaker).

2.11.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LIMR_Reference = LVM_LIMR_REF_INPUT, // LIMR reference input: LVM_LIMR_REF_INPUT or  
LVM_LIMR_REF_0DBFS  
LIMR_Threshold = 0, // LIMR threshold in dB: -24dB to 0dB
```

2.12 Parametric Spectrum Analyzer

2.12.1 Description

The parametric spectrum analyser (PSA) generates the spectral information of the output signal. This spectral information is generally used to spectral display purpose.

The PSA spectral amplitude is not affected by the EAP volume control.

User can define the number of band and the decay rate value for each band.

2.12.2 Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

```
LVM_PSA_DdecaySpeed_en PSA_PeakDecayRate; // Peak value decay rate
LVM_UINT16 PSA_NumBands; // Number of Bands
```

LVM_InstParams_t structure parameter into LVM.h::

```
LVM_UINT16 PSA_HistorySize; // PSA History size in ms: 200 to 5000
LVM_UINT16 PSA_MaxBands; // Maximum number of bands: 6 to 64
LVM_UINT16 PSA_SpectrumUpdateRate; // Spectrum update rate : 10 to 25
```

2.13 Crossover 2-Bands

2.13.1 Description

The goal of the crossover is to split the digital EAP output signal in 2 spectral bands. One dedicated to the low frequencies and another dedicated to the high frequencies. This permit to address 2-way speakers for a better sound experience.

As an example:

- Crossover cutoff frequency is around 100-250Hz according subwoofer spectral characteristic
- The low band output signal is sent to a subwoofer.
- The high band output signal is sent to a medium speaker.
- This could provide a better sound experience than send the full band to a medium.

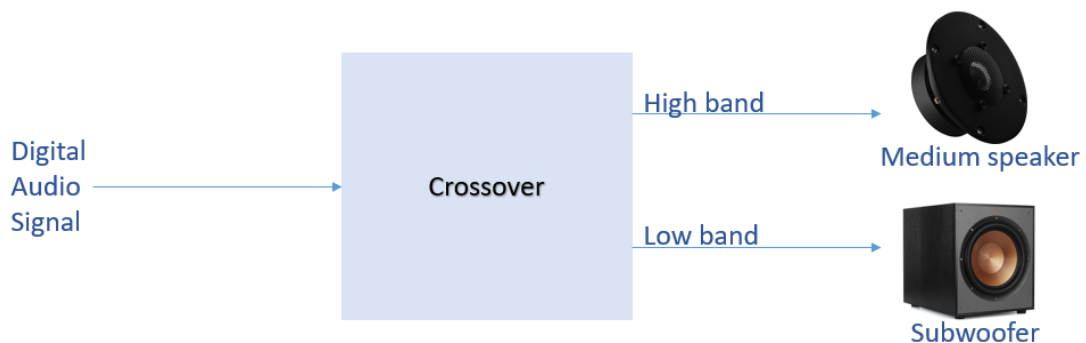


Figure 8: Crossover 2-Bands product application example

EAP crossover cutoff frequency (F_c) is configurable from 60Hz to 6KHz. EAP deliver digital audio signal in 2 bands:

- o The low band in range $(0 - F_c)$
- o The high band in range $(F_c - F_s/2)$

Each speaker has got its own frequency specifications. For example, a subwoofer is used to play the low frequencies only whereas a tweeter is used to play higher frequencies. Please refer to speaker characteristic to determine the better cutoff frequency to applied.

At library API level, digital signal output low and high bands are stored in two separate buffers. One dedicated to the low band, the other to the high band. See code API description for further details.

Users Guide

Audio reconstruction:

The audio reconstruction (Sum = low band + high band) is occurring in acoustic area. However, if we sum the low band and the high band at digital level, the EAP crossover reconstruction is almost perfect as shown in the following figure:

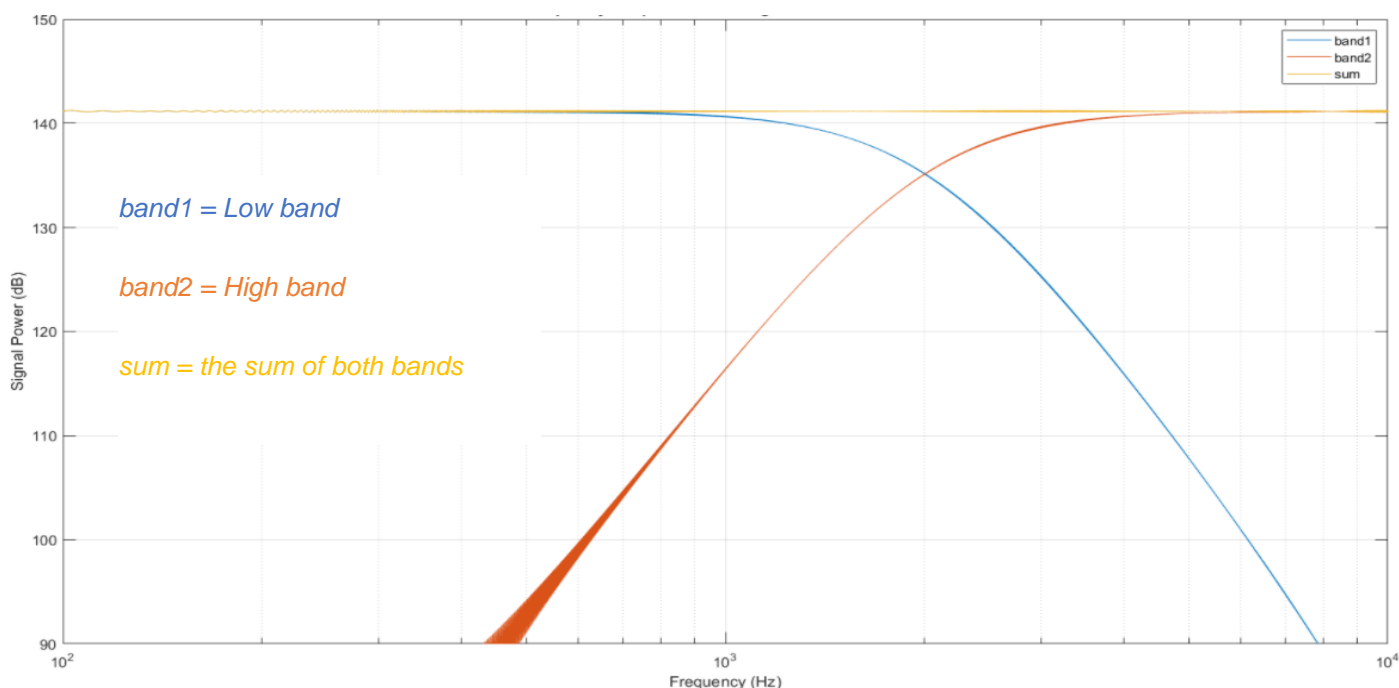


Figure 9: Frequency response of a Sweep signal

2.13.2

Definition C code

LVM_ControlParams_t structure parameter into LVM.h:

LVM_UINT16 XO_cutoffFrequency //Cutoff frequency in Hz (range = [60 Hz – 6000Hz])

2.14 Headroom Management

The EAP library uses headroom management to minimize the signal saturation risks while ensure the maximum allowed volume is possible.

It is an optional software part and can be disable. Headroom management works in pair with the user equalizer gain band definition and the automatic volume leveler if enabled.

The music may saturate when the user equalizer band is configured with a high level of boost and the signal is also at a high level.

To avoid saturations, the headroom management software acts as a watchdog. It analyses the volume control, the user equalizer band gain and the headroom management settings. Then it has the capability to internally limit the general volume.

It works as the volume control parameter is limited to a maximal value computed by the headroom management.

According to headroom allowed gain settings:

- Less the band gain equalizer is set, more the volume control value can be high.
- More the band gain equalizer is set, less the volume control can be high.

The headroom management is not signal dependent, it is a static algorithm.

Read **Note 1: Why a different gain range?** For more information about user equalization interaction with headroom management.

2.14.1 Headroom computation:

The headroom requirements are specified for different frequency bands.

The calculation of the headroom is based on the headroom management parameters and the EAP user equalizer settings (volume control and equalizer settings).

The headroom management compute an allowed gain to be applied per band frequency. It saves only the worst case and then applied this limit to all the frequency.

The allowed gain computation does not depend on the music content.

2.14.2 API

An API extension allows the user to control the headroom parameters and switch the management ON and OFF.

Headroom parameters may be changed at any time during processing using the LVM_SetHeadroomParameters function. They will take effect at the next LVM_Process call. LVM_GetHeadroomParameters permits to read the configuration. Read Special Function chapter.

2.14.3 Operating Mode

The LVM_Headroom_Mode_en enumerated type is used to enable and disable the headroom management.

2.14.3.1 Definition C code

LVM_HeadroomParams_t structure parameter into LVM.h.

```
LVM_Headroom_Mode_en Headroom_OperatingMode; // Headroom Control
```

```
LVM_HEADROOM_OFF  
or LVM_HEADROOM_ON
```

2.14.4 Band definition

It define the required headroom per frequency band.

2.14.4.1 Definition C code

```
LVM_HeadroomBandDef_t *pHeadroomDefinition; // Pointer to headroom bands definition
LVM_UINT16 NHeadroomBands;                 // Number of headroom bands, 0 to 5
```

The LVM_HeadroomBandDef_t type is used for defining the headroom band characteristics.

```
/* Headroom band definition */
typedef struct
{
    LVM_UINT16 Limit_Low;          // in Hz, range 20 to 24000 Hz
    LVM_UINT16 Limit_High;        // in Hz, range 20 to 24000 Hz
    LVM_INT16 Headroom_Offset;    // in dB range -15 to 15 dB
} LVM_HeadroomBandDef_t;
```

2.14.5 Configuration example

These examples have been formulated to help understanding of the headroom management. It may not be representative of a real device. If necessary, NXP can help you to determine suitable headroom bands settings in the scope of an integration project.

2.14.5.1 Example 1

For headroom management settings are as follows:

```
Headroom_OperatingMode      = LVM_HEADROOM_ON;
NHeadroomBands              = 2;
pHeadroomDefinition[0].Limit_Low      = 20;
pHeadroomDefinition[0].Limit_High     = 4999;
pHeadroomDefinition[0].Headroom_Offset = 3;
pHeadroomDefinition[1].Limit_Low      = 5000;
pHeadroomDefinition[1].Limit_High     = 24000;
pHeadroomDefinition[1].Headroom_Offset = 4;
```

The aim here is to achieve a trade-off between the risk of saturation and good volume.

The assumption made in this definition is:

- It is possible to apply up to 3dB of boost in the band 20Hz to 4999Hz without serious risk of saturation.
- It is possible to apply up to 4dB of boost in the band 5000Hz to 24000Hz without serious risk of saturation.

When saturation occurs, it should not be audible under normal listening conditions.

Volume control will be limited to -3dB or -4dB only if user equalizer band gain is superior to 3dB or 4dB in the corresponding frequency range.

It can be default parameters for most of the device.

2.14.5.2 Example 2

In this example, we want to control the headroom of the output signal amplitude like this:

- In the first band, 20Hz to 999Hz, the signal can reach full scale 0dBFS.
- In the second band, 1000Hz to 24000Hz, the signal must stay below -3dB to ensure the saturations are not so present.

To realize that, we configure the headroom management as follows:

```
/* Headroom parameters declaration */
LVM_HeadroomParams_t  HeadroomParams;
LVM_HeadroomBandDef_t HeadroomBandDef[2];

/* Headroom bands definition */
HeadroomBandDef[0].Limit_Low      = 20;
HeadroomBandDef[0].Limit_High     = 999;
HeadroomBandDef[0].Headroom_Offset = 0;
HeadroomBandDef[1].Limit_Low      = 1000;
HeadroomBandDef[1].Limit_High     = 24000;
HeadroomBandDef[1].Headroom_Offset = -3;

/* Headroom parameters setting */
HeadroomParams.pHeadroomDefinition = &HeadroomBandDef[0];
HeadroomParams.Headroom_OperatingMode = LVM_HEADROOM_ON;
HeadroomParams.NHeadroomBands = 2;

/* Apply changes */
LVM_SetHeadroomParams(hInstance, &HeadroomParams);
```

We need to consider two cases: with and without the equalizer enabled.

2.14.5.2.1 Without the equalizer

The EAP headroom control software checks the headroom parameters and the EAP settings.

It sees that for frequencies above 1kHz the output gain should never be greater than -3dB but for other frequencies it can be as high as 0dB.

The equalizer is disabled so this does not affect the calculation.

The calculation chooses a maximum output level of -3dB and applies this over the entire music spectrum.

For an input signal at 0dBFS, the output would be at -3dBFS.

The side effect of the setting is that the maximum volume is reduced by 3dB. It is a relatively small loss. On the other hand, the mid to higher frequency signals get the -3dB saturation protection we would like to guarantee.

Note: If this rule was applied only above 1kHz, it would introduce attenuation at higher frequencies, changing the tonal characteristics of the music.

2.14.5.2.2 With the equalizer

The equalizer is set for the treble boost preset, this has the following parameters:

Users Guide

- 5 bands
- Band #1: Freq = 50Hz, Q= 0.96, Gain = 0dB
- Band #2: Freq = 205Hz, Q= 0.96, Gain = 0dB
- Band #3: Freq = 837Hz, Q= 0.96, Gain = 2dB
- Band #4: Freq = 3427Hz, Q= 0.96, Gain = 4dB
- Band #5: Freq = 14027Hz, Q= 0.96, Gain = 6dB

The headroom band limit are still:

- 20Hz to 1kHz, offset = 0dB
- 1kHz to 24kHz, offset = -3dB

The EAP headroom control software checks the headroom parameters and the EAP settings

It combines the headroom band limit value with the highest equalizer boost settings (in the band):

- 20Hz to 1kHz, offset = 0dB, maximum equalizer boost = 2dB
- 1kHz to 24kHz, offset = -3dB, maximum equalizer boost = 6dB

The calculation determines that it is possible to boost up to 6dB in the 1kHz to 24kHz band and the output must stay below -3dBFS. So, system required 9dB of headroom for all the frequencies.

The internal volume setting will be limited to -9dB whatever the volume control parameter is.

2.14.6

Conclusion

To control the amount of potential saturation the maximum volume will be reduced.

While tuning phase, the headroom manager permits to deal between amount of acceptable saturations at maximum level, the maximal level allowed and the user equalizer gain.

Headroom manager adapt automatically the maximal volume gain allowed for an equalization parameters settings.

3 EAP INTEGRATION

This chapter illustrates an application code example which is distributed with the EAP library.

An example application is provided in the release to show classical integration of the EAP(see the file EAP_ExApp.c).

3.1 Sequence & Description

To initialize the algorithm and run the process. It requires the following steps:

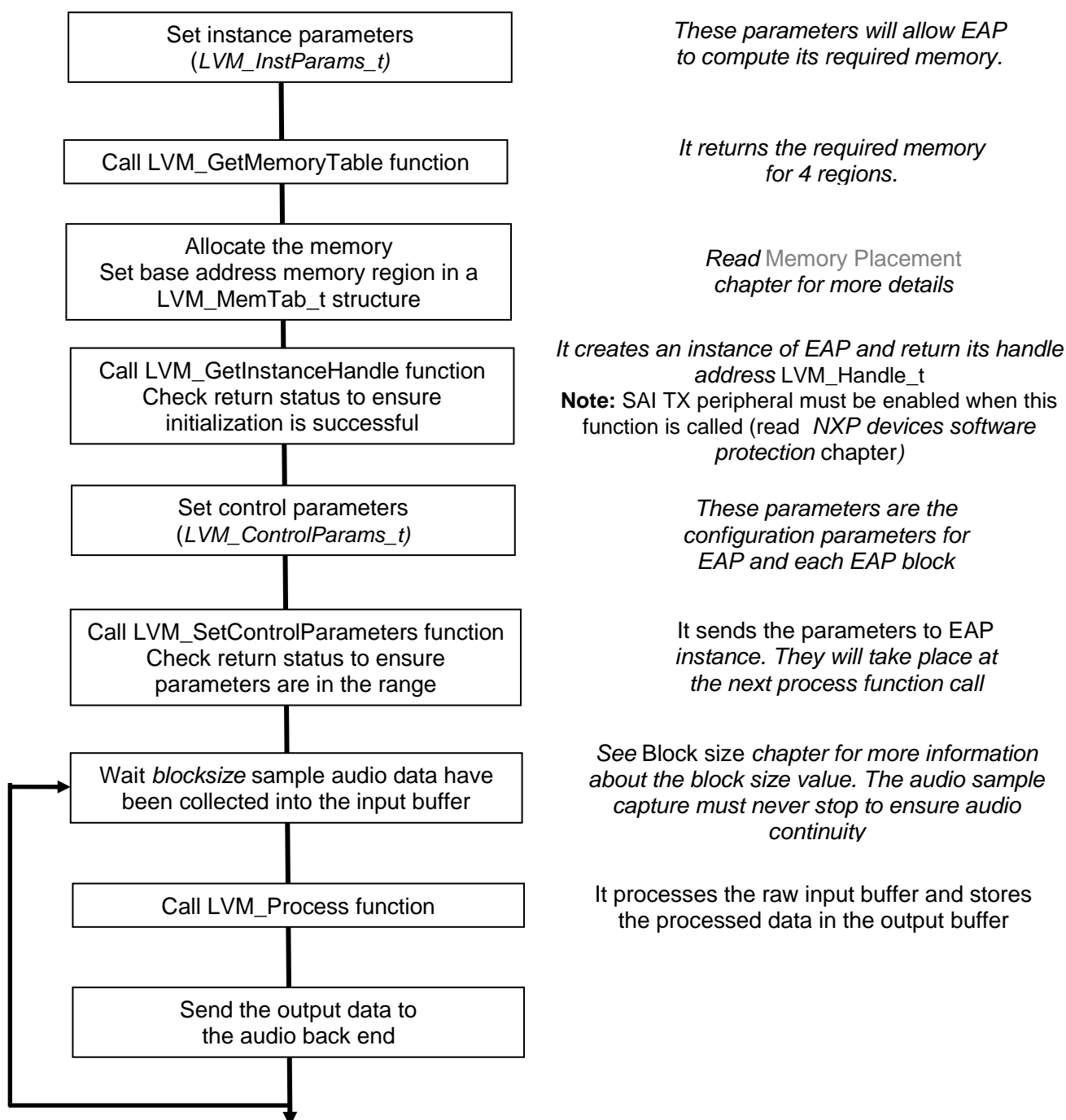


Figure 10: Integration - Initialization & process *sequence*

The parameters may also be updated after initialization. This procedure may be used at any time while running the algorithm:

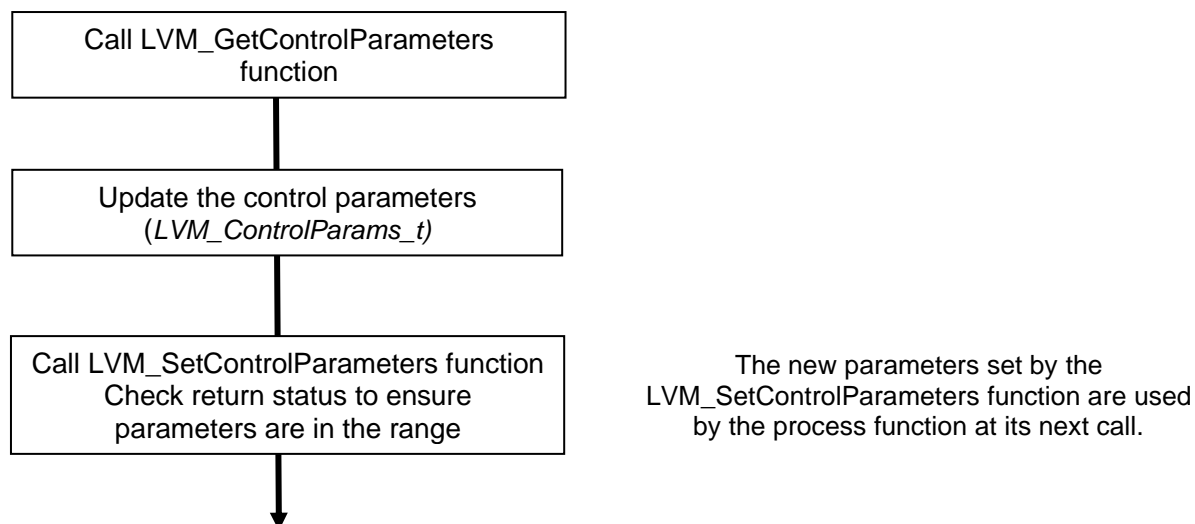


Figure 11: Integration – Update control parameters

3.2 Special Functions

In addition of the classical functions presented in the *Sequence & Description* chapter there are special functions which offer more flexibility and control.

3.2.1 LVM_GetVersionInfo

This function is used to retrieve information about the library's version.

3.2.2 LVM_ClearAudioBuffers

This function is used to clear the internal audio buffers of the bundle.

3.2.3 LVM_GetAVLGain

This function is used to retrieve the AVL last generated gain.

3.2.4 LVM_SetHeadroomParams

Available only If library is compiled with Equalizer block

This function is used to set the headroom management parameters.

3.2.5 LVM_GetHeadroomParams

Available only If library is compiled with Equalizer block

This function is used to get the headroom management parameters.

3.2.6 LVM_GetSpectrum

Available only If library is compiled with PSA block

This function is used to retrieve Spectral information at a given Audio time for display usage.

3.2.7 LVM_SetVolumeNoSmoothing

This function is used to set output volume without any smoothing.

3.3 Memory Placement

The EAP library uses four main memory regions. The placement of these regions in the processor memory impacts the MIPS consumption.

LVM_TEMPORARY_FAST

- Used for storage of intermediate results
- Used to store a copy of the input buffer

LVM_PERSISTENT_SLOW_DATA

- Used for storage of instance parameters
- Used for storage of control parameters
- Used for storage of control variables
- Used for storage of delay line data
- Used for storage of a copy of the equalizer filter definitions

LVM_PERSISTENT_FAST_DATA

- Used for storage of filter data history

LVM_PERSISTENT_FAST_COEF

Used for storage of filter coefficients

Both the LVM_PERSISTENT_FAST_DATA and LVM_PERSISTENT_FAST_COEF memory regions are used very intensively and should be given the top priority in placement. On some processors this can be in:

- dedicated separate memory spaces
- dual access memory
- zero wait state memory

The LVM_TEMPORARY_FAST memory is also used intensively but can be released after function call. This should be placed in fast memory when possible, slow memory will significantly lower performance.

LVM_PERSISTENT_SLOW_DATA is less intensively accessed. Only used during initialization or only concern few variables at each run. This can be placed in a slow memory area without significant effect on the overall MIPS consumption.

4 MIPS & MEMORY

Mips are measured on SDK platform with Segger system view tool.

According available resource for EAP on the final product, you can disable some algorithms to save MIPS. Note that all EAP algorithm can't be enabled at the same time on a LPC55 platform.

Memory size is compilation dependent and can't be adjustable by enabling or disabling algorithm.

Algorithm	FOOT PRINT (K Byte)			MIPS (MHz)	
	Scratch	RAM	code size	Cortex M7 (RT1060) (Segger system view)	Cortex M33 (LPC55) (Segger system view)
Equalizer (EQ) 2 bands	3.6	1.7	1.0	14.4	60.0
Equalizer (EQ) 4 bands				19.8	72.0
Equalizer (EQ) 6 bands				25.2	88.0
EQ 10 bands + LPF + HPF				61.2	Na
Pre-Equalizer (EQ) 2 bands	3.6	1.7	1.0	14.4	60.0
Pre-Equalizer (EQ) 4 bands				19.8	72.0
Pre-Equalizer (EQ) 6 bands				25.2	88.0
Pre-EQ 10 bands + LPF + HPF				61.2	Na
3D Concert Sound (CS)	72.7	25.2	3.0	8.4	Na
Treble enhancer (TE)	0.0	0.0	4.0	3.6	4.5
Bass enhancement (DBE)	5.4	0.6	5.0	22.8	18.0
Loudness maximiser (LM)	3.6	0.4	7.0	7.2	19.5
Auto Volume leveler (AVL)	0.3	0.2	8.0	7.2	7.5
Tone generator (TG)	0.1	0.3	9.0	3.6	7.5
Power spectrum analysis (PSA)	2.1	4.5	10.0	0.6	0.3
Peak Limiter (LIMP)	0.0	12.0	11.0	22.8	69.0
RMS Limiter (LIMR)	0.0	12.0	11.0	6.6	6.0
Crossover 2-bands(XO)	0.9	1.5	11.0	48.0	Na

ABBREVIATIONS AND REFERENCES

Abbreviations	
API	Application Programmers Interface
AVL	Auto Volume Leveler
BE	Bass Enhancement, either PureBass or DBE which ever 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
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
MSB	Most significant bit

References	

CHANGE HISTORY

Version	Description	Date
1.0	Initial	2020/03/05
1.1	1 st Review	2020/03/11
1.2	Add source format information	2020/04/09
2.0	Add peak limiter Add RMS limiter Add speaker equalization Add concert sound advanced parameter Update IN / OUT format Remove custom tuning API function	2020/08/05
2.1	Update HeadRoom management chapter	2020/12/16
3.0	Add Crossover module description	2021/04/06
3.1	Update input/output buffer table	2021/04/12
3.2	Add chapter about MIPS & Memory	2021/04/27
3.3	Updates Mips with simulation values	2021/06/02
3.4	Add NXP devices software protection description	2021/06/11
3.5	SW protection detail update	2021/06/15
3.6	Update MIPS value	2021/06/22
4.0	Update for 32 bits, 96kHz, feature pack	2021/09/21