Essential Audio Processing

Users Guide

Released 1.2 - ***2020-Apr-09*** **CONFIDENTIAL**

Document information	
Title	Essential Audio Processing Users Guide
Status	Released
Version	1.2
Date	2020-Apr-099
Author	Christophe Boulant
Reviewers	LP
Approved by	Laurent Pilati
Document ID	NXPSW_SO_04100
Security	CONFIDENTIAL The attached material and the information contained therein are proprietary to NXP Software, and are issued only under strict confidentiality arrangements. Without a separate written authorization, it shall not be used for any purpose whatsoever, reproduced, copied in whole or in part, adapted, modified, or disseminated. It must be returned to NXP Software upon its first request.
Usage	**NXP**

Change History

Version	Status	Description	Author	Date
1.0	Proposed	Initial	Christophe Boulant	2020/03/05
1.1	Reviewed	1st Review	Laurent Pilati	2020/03/11
1.2	Reviewed	Add source format information	Christophe Boulant	2020/04/09

TABLE OF CONTENT

1 DOCUMENT DESCRIPTION

1.1 Purpose

This document provides required information to understand, deploy and tune the Essential Audio Processing solution.

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.

1.2 Audio Tuning:

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

The Essential Audio Processing (aka EAP) is provided with Windows simulator to exercise the algorithms. But you need to plan tuning session with the actual hardware.

Classic audio tuning is simple and permits to obtain good results with few audio processing knowledges 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 with acoustic engineer who support the EAP block:

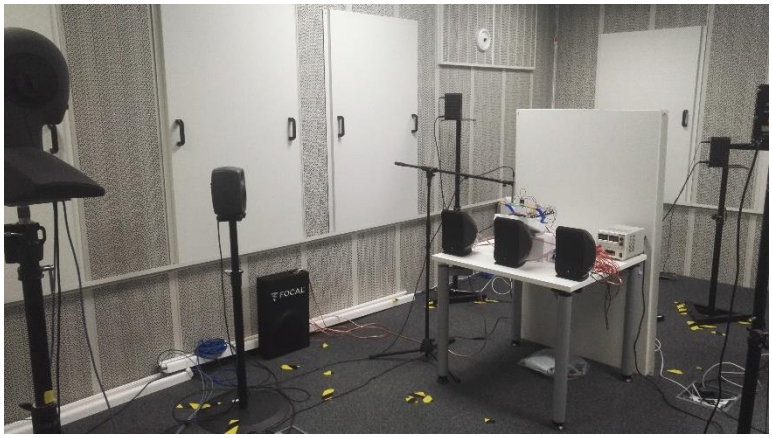
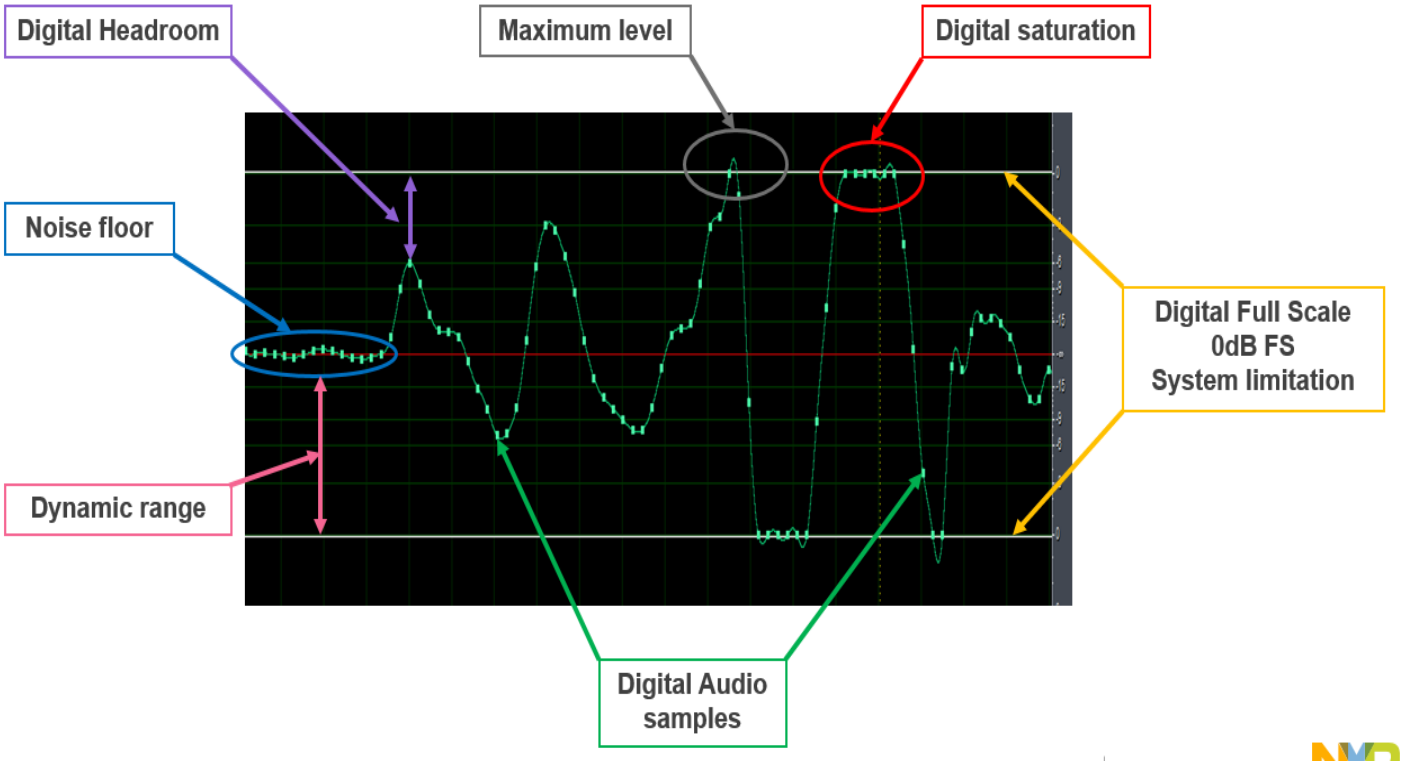


Figure 1 – NXP Audio Laboratory

1.3 Some terminology

Here are some terminologies used in t his document.



2 EAP DESCRIPTION

This chapter regroups audio block description and tuning information.

2.1 General

2.1.1 Overview

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

It supports mono or stereo raw audio data in 16bits at multiple sample rate 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100 or 48000 Hz.

The EAP includes the following sound processing algorithms:

- 3D Virtualization: ConcertSound or CinemaSound
- Speaker equalizer
- User Equalizer
- Bass Enhancement (Pure Bass or Digital Bass Enhancement)
- Volume Control
- Treble Enhancement
- Loudness Maximiser
- Auto Volume Leveler
- Tone Generator
- Parametric Spectrum Analyzer

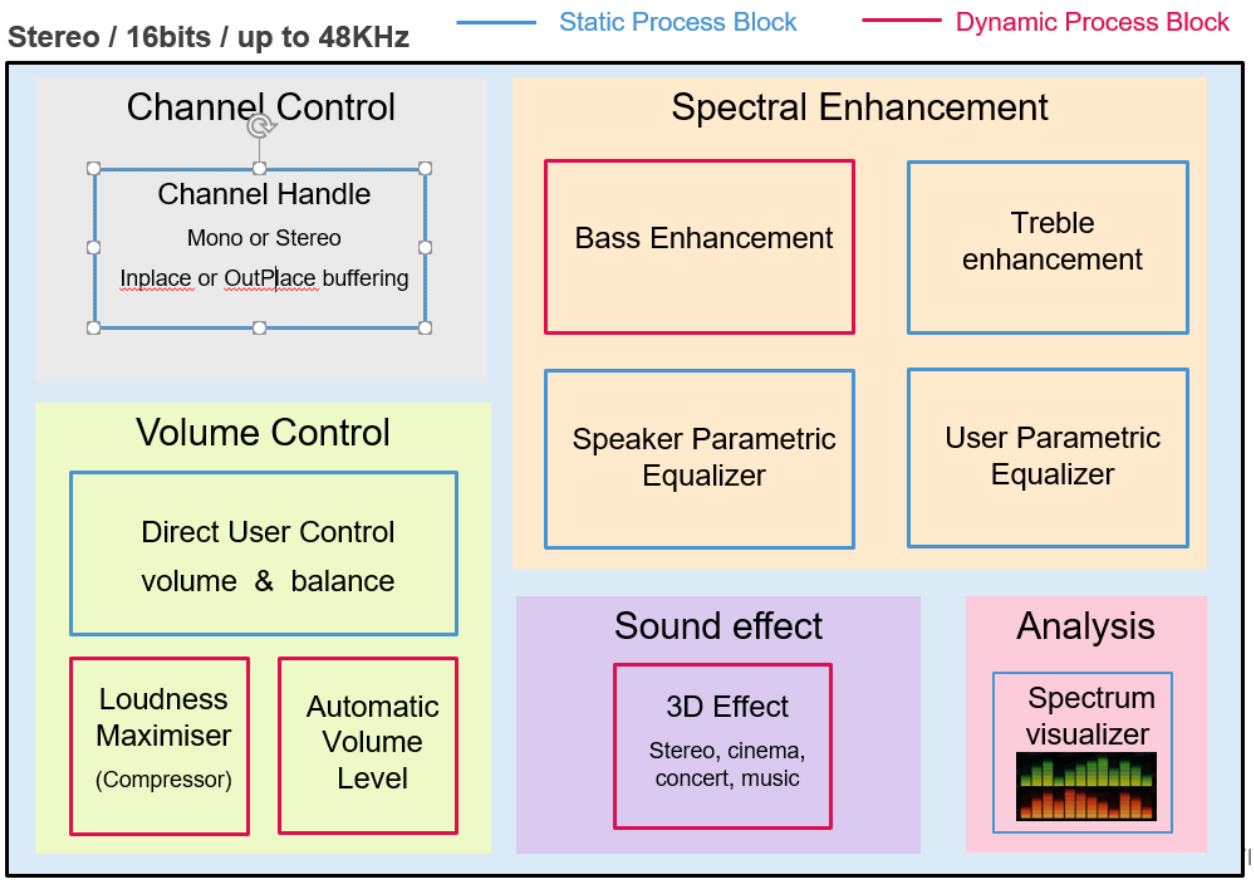


Figure 2 - EAP software audio block

The process can be performed in-place (input and output buffer are same) or not in-place (input and output buffer are different).

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 but also improves the 'spatialness' of the audio, resulting in an enjoyable and relaxing listening experience.

EAP audio block can be re-ordered if required.
As an example, the bellow picture shows the default and recommended audio chain order.

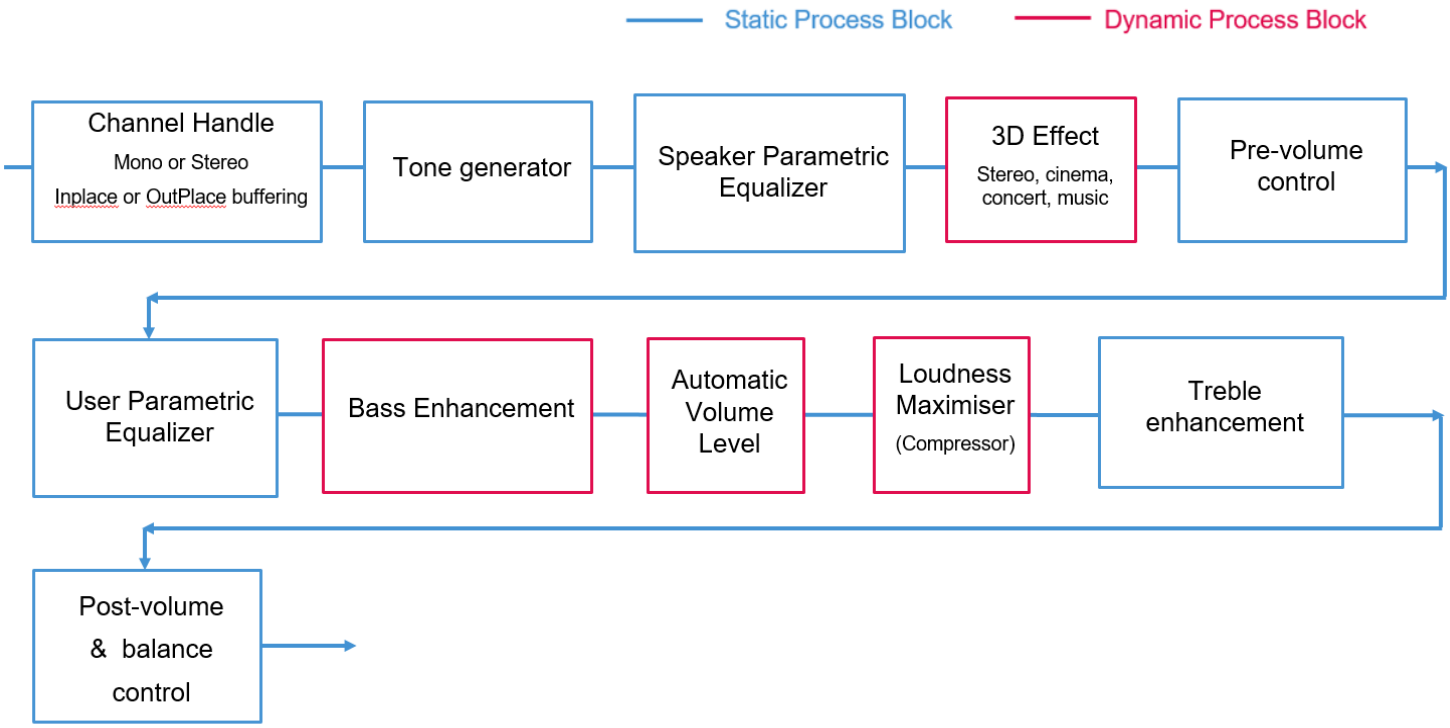


Figure 3 - EAP default audio chain order

2.1.2 Operating mode

Entire EAP or Each EAP block 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.

2.1.2.2 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
ON                  // Bundle operating mode: ON or BYPASS
```

```
OFF             // Virtualizer operating mode: ON or OFF
BE_OFF          // BE operating mode: BE_ON or BE_OFF
TREBLE_OFF      // Treble Boost operating mode: TREBLE_ON or TREBLE_OFF
LM_OFF          // Loudness Maximiser operating mode: LM_ON or LM_OFF
AVL_OFF         // AVL operating mode: AVL_OFF or AVL_ON
TG_OFF          // TG operating mode: TG_OFF, TG_CONTINUOUS or TG_ONESHOT
OFF             // PSA Enable (ON/OFF)
EQNB_ON         // N-Band equaliser operating mode: EQNB_ON or EQNB_OFF
```

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:

- Normal sample rates: 32kHz, 44.1kHz and 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
```

2.1.3.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
44100  // Sample rate: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100 or 48000
```

2.1.4 Source format

2.1.4.1 Description

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

The LifeVibes bundle supports input data in the five formats: Mono, MonoInStereo, Stereo, 5.1 and 7.1. For an input buffer of NumSamples = N (meaning N sample pairs for Stereo and MonoInStereo or N samples for Mono) the format of data in the buffer will be as follows:

Sample Number	Stereo	MonoInStereo	Mono
0	Left(0)	Mono(0)	Mono(0)
1	Right(0)	Mono(0)	Mono(1)
2	Left(1)	Mono(1)	Mono(2)
3	Right(1)	Mono(1)	Mono(3)
4	Left(2)	Mono(2)	Mono(4)
“	“	“	“
“	“	“	“
N-2	Left(N/2-1)	Mono(N/2-1)	Mono(N-2)
N-1	Right(N/2-1)	Mono(N/2-1)	Mono(N-1)
N	Left(N/2)	Mono(N/2)	Not Used
N+1	Right(N/2)	Mono(N/2)	Not Used
N+2	Left(N/2+1)	Mono(N/2+1)	Not Used
N+3	Right(N/2+1)	Mono(N/2+1)	Not Used
“	“	“	Not Used
“	“	“	Not Used
2*N-2	Left(N-1)	Mono(N-1)	Not Used
2*N-1	Right(N-1)	Mono(N-1)	Not Used

Sample Number	5.1	7.1
0	Front Left	Front Left
1	Front Right	Front Right
2	Centre	Centre
3	LFE	LFE
4	Rear Left	Surround Left
5	Rear Right	Surround Right
6	Front Left	Surround Back Left
7	Front Right	Surround Back Right

Whatever the input format the output

2.1.4.2 Definition C code

`LVM_ControlParams_t` structure parameter into LVM.h:

```
LVM_Format_en  SourceFormat; // Input data format LVM_STEREO, LVM_MONOINSTEREO,
                             LVM_MONO LVM_5DOT1 or LVM_7DOT1
```

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

2.1.4.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
STEREO                  // Input file format: MONO, MONOINSTEREO, STEREO, 5DOT1 or 7DOT1
```

2.1.5 Output device

2.1.5.1 Description

The output device may be headphones or speakers.

If speaker, 3 speaker types (small, medium or large) can be selected. Speaker type is only used by the 3D Virtualizer if enabled.

2.1.5.2 Speaker type identification

Table 1: 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.

2.1.5.3 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
```

2.1.5.4 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
HEADPHONE    // output device: HEADPHONE, MS_SMALL, MS_MEDIUM or MS_LARGE
```

2.1.6 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.

2.1.6.1 Definition C code

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

2.1.6.2 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
441                 // Block size, maximum 1024
```

2.2 Tone Generator

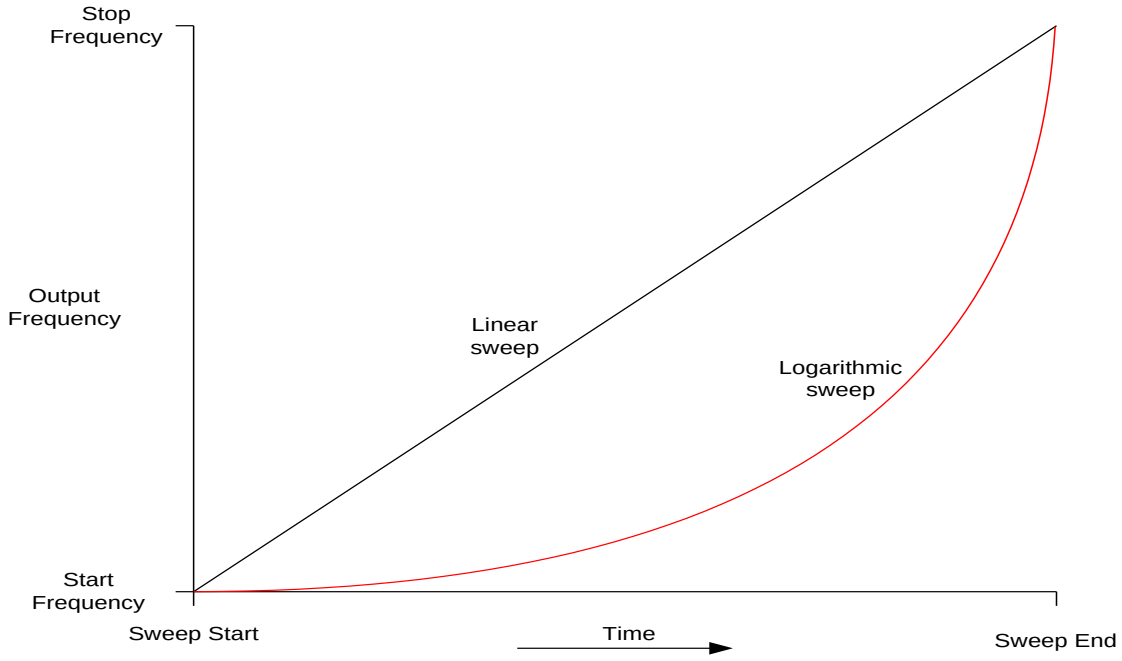
2.2.1 Description

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.

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.
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
```

2.2.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
TG_ONESHOT          // TG operating mode: TG_OFF, TG_CONTINUOUS or TG_ONESHOT
TG_LOG              // Sweep mode: TG_LIN or TG_LOG
20                  // Start frequency in Hz
0                   // Start amplitude in dBr
22000               // Stop frequency in Hz
0                   // Stop amplitude in dBr
60                  // Sweep duration in seconds, 0 for infinite duration tone
```

2.3 Parametric Equalizer

2.3.1 Description

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.

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

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 equalization is controlled through the custom tuning parameters function.

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.
- The Q factor, controlled over the range 0.25 to 12.00 in steps of 0.01.

Each band has 3 parameters:

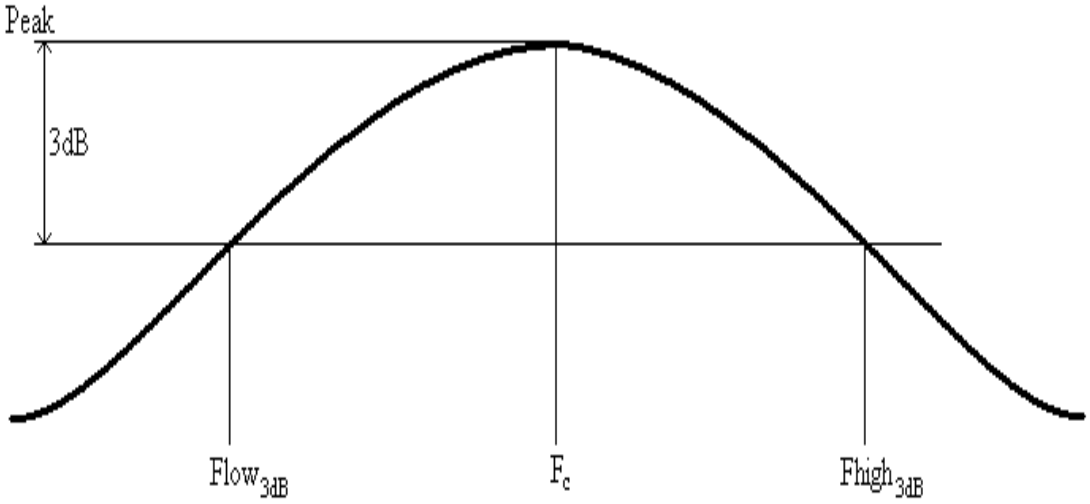
Gain	the gain of the band in dB, from –15dB to +15dB in 1dB step. If the gain is set to 0dB the band will be disabled
Frequency	the frequency is in Hz, from 20Hz to Nyquist frequency (half the sample rate)
Q-factor	the filter Q, from 0.25 to 12.00 in steps of 01. **WARNING:** The desired Q value is the 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: $Q\text{-factor} = F_c \ / \ (Fhigh_{3dB} - Flow_{3dB})$

Where:

F_c - Filter center frequency (the Frequency setting)

$Flow_{3dB}$ - The lower 3dB cut-off frequency of the filter

$Fhigh_{3dB}$ - The upper 3dB cut-off frequency of the filter



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

```
#define MAX_BANDS        15
```

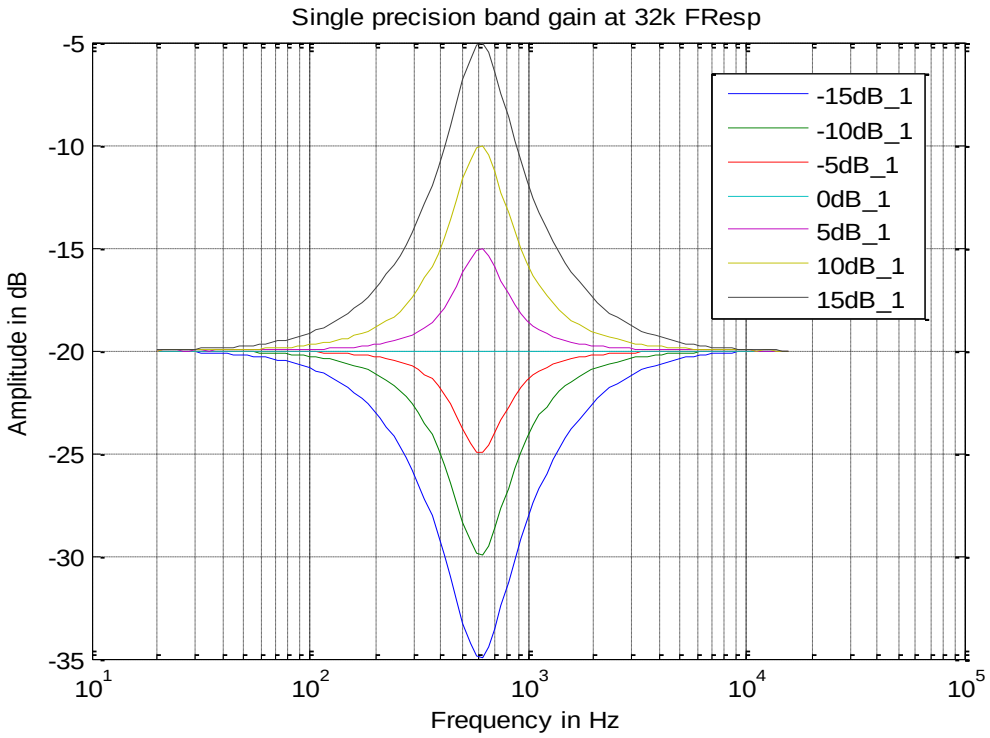
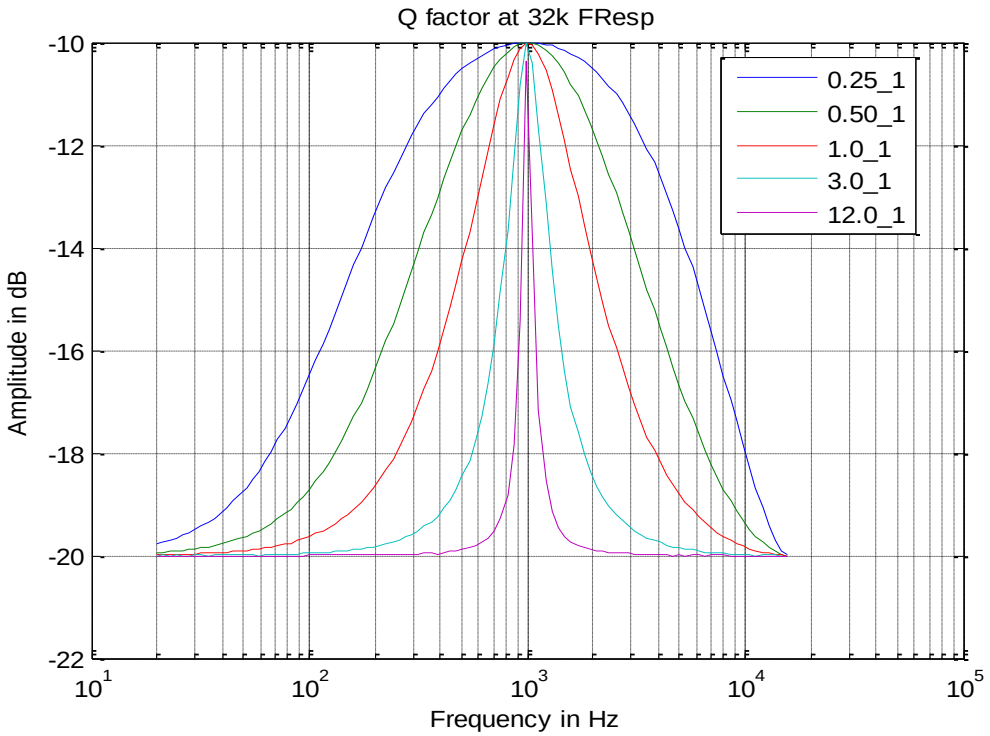


Figure 4 – Equalizer filter gain settings



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	2nd order	1st 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	Band				
Preset Name	**#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
Theather	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 2 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 2 (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
```

2.3.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
EQNB_ON          // N-Band equaliser operating mode: EQNB_ON or EQNB_OFF
LPF_ON           // LPF operating mode: LPF_OFF or LPF_ON
22000            // LPF corner frequency in Hz: 1kHz to Nyquist
HPF_ON           // HPF operating mode: HPF_OFF or HPF_ON
20               // HPF corner frequency in Hz: 20Hz to 1kHz
5                // Number of bands required
-1               // Band gain in dB
50               // Band center frequency
96               // Band Q (x100)
-1               // Band gain in dB
205              // Band center frequency
96               // Band Q (x100)
-1               // Band gain in dB
837              // Band center frequency
96               // Band Q (x100)
-1               // Band gain in dB
```

```
3427            // Band center frequency
96              // Band Q (x100)
-1              // Band gain in dB
14027           // Band center frequency
96              // Band Q (x100)
```

2.4 3D Widening

2.4.1 Description

The CinemaSound and ConcertSound ambience provides a cost-effective, enjoyable and relaxing listening experience.

With headphones it is offering 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 an enjoyable sound with widened stereo (also sometimes called 3D widening). It works on music and film sound tracks and leaves the voice natural.

Cinema and Concert sound 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
```

2.4.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
MUSIC       // Virtualizer media type: CONCERTSOUND, MUSIC or MOVIE
100         // Virtualizer reverb level: from 0% to 100%
32767       // Effect level 0 to 32767: low=16384, medium=24576 or high=32767
```

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)
```

2.5.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
0               // Volume in dB: -96 to 0
0               // Volume Balance -96 to 96
```

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
```

2.6.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
15                  // BE effect level: 0 to 15 in dB
BE_CENTRE_90        // BE center frequency: BE_CENTRE_55, BE_CENTRE_66, BE_CENTRE_78 or BE_CENTRE_90
BE_HPF_OFF          // BE high pass filter: BE_HPF_ON or BE_HPF_OFF
```

2.7 Audio Volume Leveler

2.7.1 Description

The volume level may requires 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.

2.7.3 Definition simulator

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:

```
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
```

2.8.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
EXTREME     // Loudness Maximiser effect level: GENTLE, MEDIUM or EXTREME
0           // Loudness Maximiser output attenuation: 0 to 6 in dB
6           // Loudness Maximiser output compressor gain: 0 to 6 in dB
500         // Loudness Maximiser speaker cutoff 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.

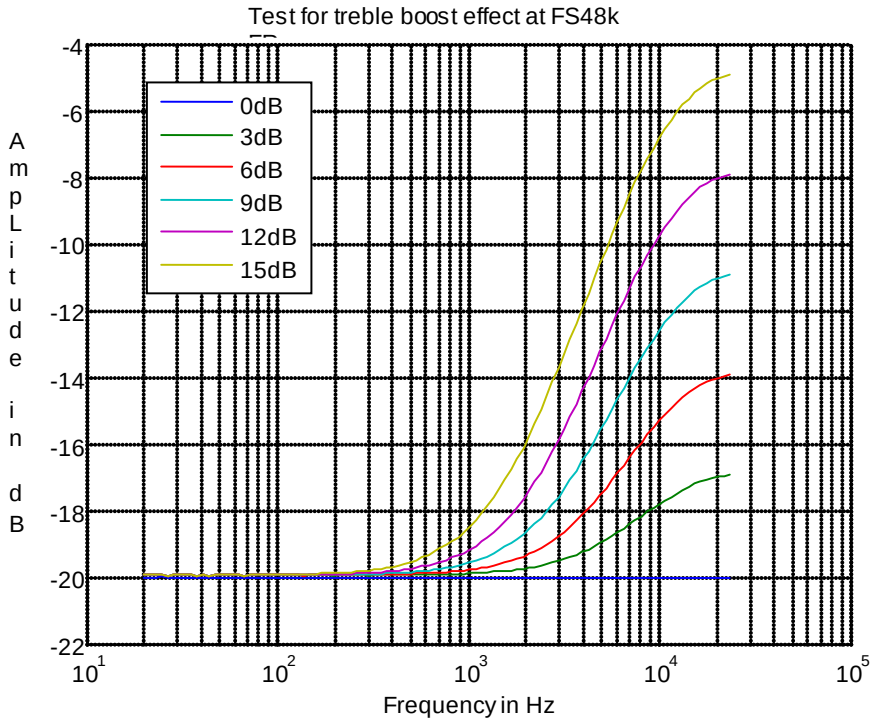


Figure 5 Treble Enhancement at different effect levels

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
```

2.9.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
0                   // Treble boost level, 0dB  to 15dB or LVM_TE_LOW_MIPS
```

2.10 Parametric Spectrum Analyzer

2.10.1 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.

2.10.2 Definition C code

`LVM_ControlParams_t` structure parameter into LVM.h:

```
LVM_PSA_DecaySpeed_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
```

2.10.3 Definition simulator

EAP simulator for windows is using parameters located into the TAL_args.txt:

```
PSATestOutput.dat      // Relative test output file path and name
SLOW                   // PSA Decay Rate (SLOW / MEDIUM / FAST)
32                     // PSA Number of Bands (6 to 64)
```

```
1000            // PSA History size (200-5000)ms
64              // PSA maximum number of Bands (6 to 64)
25              // PSA Update rate (10 to 25)
ON              // PSA Include in memory allocation ON/OFF
```

2.11 Headroom management

The EAP library uses headroom management to minimize the risk of the signal saturating and ensure maximum allowed volume is possible.

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

To avoid saturations, a headroom is applied to the music before any processing. This headroom is the same for all frequencies.

2.11.1 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 and the Volume Control of the bundle work together. In case the volume level is lowered using the Volume Control, the amount of headroom is automatically reduced.

The amount of headroom does not depend on the music content.

2.11.2 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.

2.11.3 Operating Mode

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

2.11.3.1 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
```

2.11.3.2 Definition simulator:

Not tunable in simulator. A fix headroom configuration can be embedded at compilation.

2.11.4 Band definition

It define the requiered headroom per frequency band.

2.11.4.1 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;
```

2.11.5 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.

2.11.5.1 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.

It can be default parameters for most of the device.

2.11.5.2 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.

2.11.5.2.1 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.

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

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, there has been an improvement in audio quality, especially for mid to higher frequency signals.

2.11.5.2.2 With the equalizer

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

- 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:

- 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 concludes 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.

For an input signal at 0dBFS, the output would be at -9dBFS at low frequencies, rising to -3dB at high frequencies. Saturation risks is still reduced.

The side effect of the setting is that the maximum volume allowed is reduced by up to 9dB, but only for low frequencies.

2.11.6 Conclusion:

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

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

Headroom manager adapt automatically with volume control setting and equalization parameters settings.

3 EAP INTEGRATION

This chapter is showing an application code example which should be distributed with the EAP library. An example application SW is provided in the release to show classical integration of the EAP.

3.1 Sequence & description

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

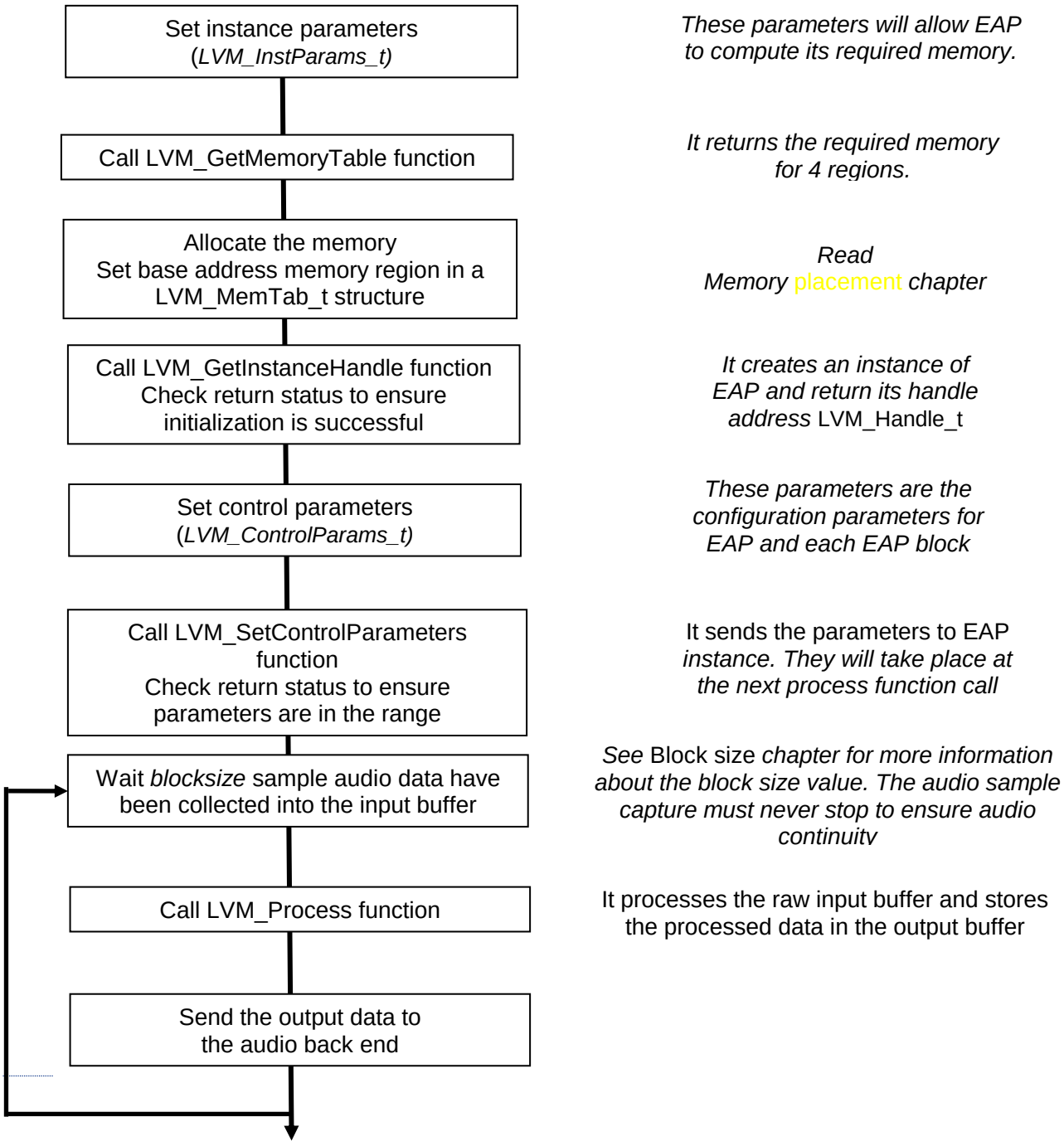


Figure 6: Integration - Initialization & process *sequence*

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

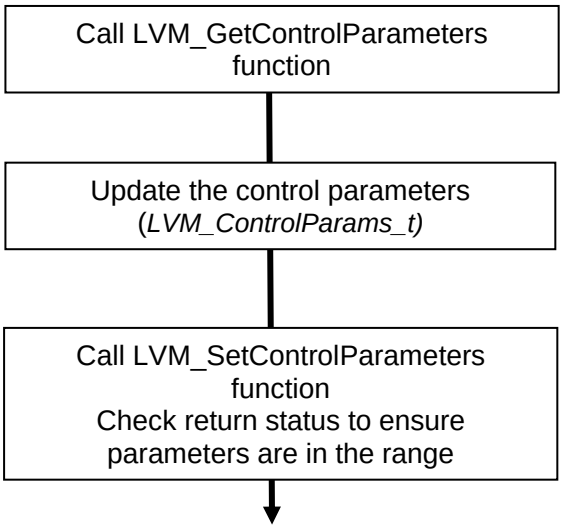


Figure 7: Integration – Update control parameters

3.2 Special function

In addition of the classical function presented in the Sequence & description chapter there is special function which offers 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.

3.2.8 LVM_SetCustomTuning

This function is used to set custom tuning parameters.

3.2.9 LVM_GetCustomTuning

This function is used to retrieve custom tuning parameters.

3.3 Memory placement

The EAP uses four main memory regions. The placement of these regions in the processor memory impact 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.

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

References