Virtual Surround Design Specification

Abstract

This document describes the Virtual Surround component used to produce a 5.1 multichannel audio scene through virtualization in a pair of headphones. It includes details about the requirement, design, reference implementation and OMX IL interface specification.

Authors

Jonas Lundbäck (jonas.xj.lundback@stericsson.com)

William Glass (william.glass@st.com) added sections 4,5,6,7 (J uly 19, 2012)

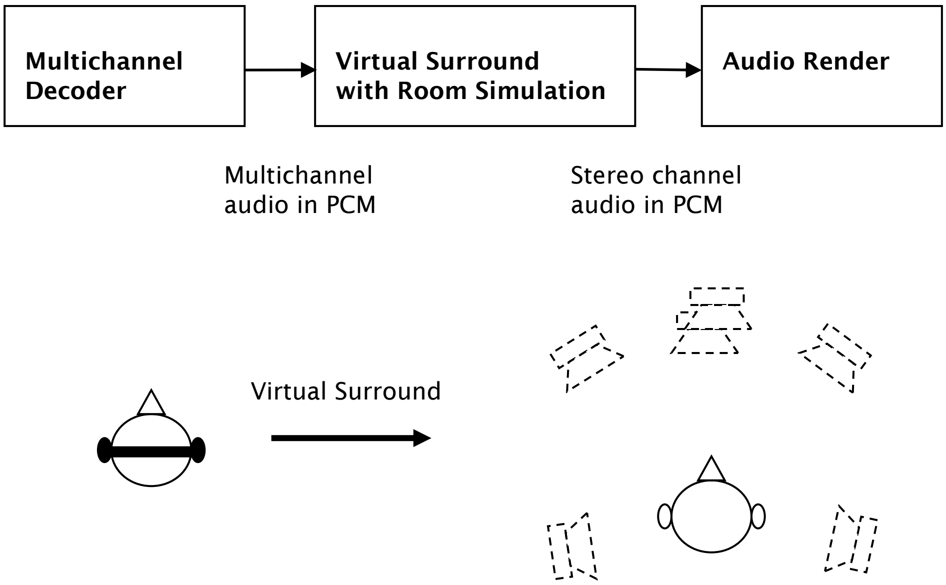
Additional Information

When this component is schedule for implementation I have reference code to use as a starting point.

Contents

1 Virtual Surround - Overview

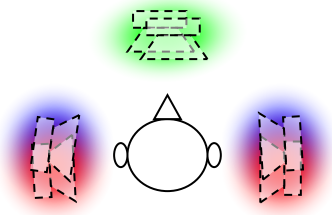
Virtual Surround is multichannel to binaural converter with the purpose to position and enhance the multichannel audio in such a way that the user listening (through headphones) experiences an externalization and a spatial localization similar to listening to a 5.1 surround system.



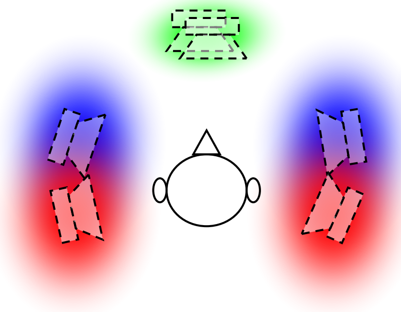
There are three different rendering modes of the virtual surround algorithm; each having a specific benefit over the others in terms of complexity or externalization and virtualization.

The room simulation is an extra feature that adds an extra spatial sensation to the virtualized audio while at the same time creating an audio scene where the listener can be part of a movie audience, a crowd at a concert or at home in the living room.

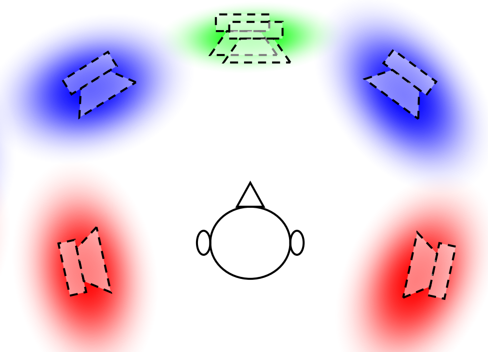
Virtual Surround Rendering Modes



Matrix Left/Right Downmix
No frequency coloration
Low complexity
No virtualization

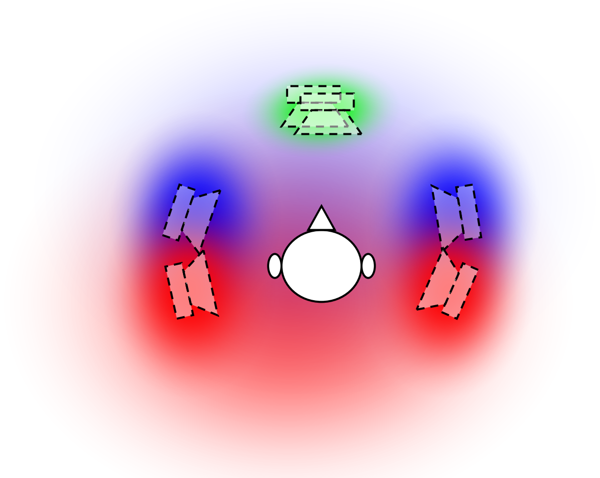


Matrix Surround Downmix
No frequency coloration
Low complexity
Low virtualization

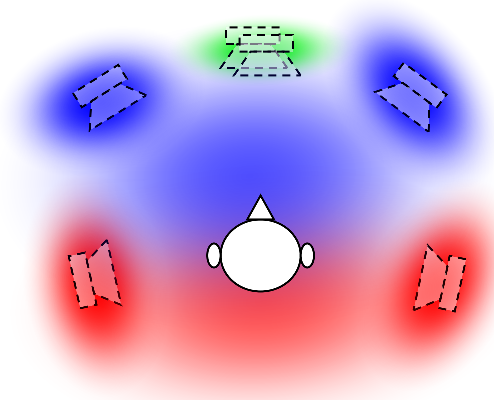


Surround Virtualization
Low frequency coloration
Medium complexity
High virtualization

Virtual Surround Room Simulation



Room Simulation Mixed Mode
High externalization
Diffuse source location



Room Simulation Virtualized Mode
High externalization
Front and surround separation

1.1 Requirement Description

Requirements:

- Supported sample rate is 48 kHz
- Component shall accept 6 channels in and provide 2 channels out. Channel order is according to OMX IL OMX_AUDIO_CHANNELTYPE and includes 0x0 to 0x6.
- The input and output bit-resolution of audio shall be 16 bit. Internal processing can be 16 and 32 bit.
- All parameters are changeable in runtime (OMX IL SetConfig) except sample rate and frame size.
- Arbitrary frame size is supported but not changeable unless component is reallocated (according to standard OMX IL)

2 Interface Specification

There is no standardized component in OMX IL for this type of implementation. There is a reference to 5.1 to stereo in Stereo Widening component in OMX IL. However, 5.1 to stereo conversion have nothing to do with stereo widening and therefore we shall define a proprietary component called Virtual Surround.

2.1 Parameter Description

The VSC can be at runtime configured by the following parameters. Names in the table below are only for this description. Different names are used in the OMX IL interface definition.

Parameter Name	**Description**
Rendering Mode	*Rendering mode is the selection of how the VSC should process the multichannel audio to produce either a stereo audio signal or a binaural audio signal. The choice of rendering mode will affect the user experience and the computational load.* Required parameter types: Enumeration values
MLRMD	Multichannel Left-Right Matrix Downmix. Matrix down-mix without left and right surround mixing.
MSMD	Multichannel Surround Matrix Downmix with left and right surround mixing. It can be considered as a low complexity version of
MSVD	Multichannel Surround Virtualization Downmix. This mode is also denoted Virtual Surround
Room Simulation	*Control the optional reverberation processing to simulate the room where audio is virtualized.* Required parameter types: Boolean values
RSIM_FRONT_OFFON	Room Simulation of front speakers OFF and ON. Controls if the reverberation should be executed and mixed with the downmixed multichannel audio.

RSIM_SURROUND_OFFON	Room Simulation of surround speakers OFF and ON. Controls if the reverberation should be executed and mixed with the downmixed multichannel audio.
Room Simulation Mode	*Control which channels that are used to perform the room simulation.* Required parameter types: Enumeration values
RSIM_MIXMODE	In this mode the unprocessed front and surround channels are inputs to the reverberation processing.
RSIM_VIZMODE	In this mode the processed front and surround channels are inputs to the reverberation processing.
Room Simulation Room type	*The Room type will completely configure the reverberation processing part of the virtual surround component.* Required parameter types: Enumeration values
ROOM_TYPE	Can be any of: Room Living Room Auditorium Concert Hall Arena Small Room Medium Room Large Room Medium Hall Large Hall.
Mixing Gain	*The mixing gain of each channel (after processing) can be adjusted before final mixing into left and right channel.* Required parameter types: sint16
LF_GAIN	Left Front Gain

	Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
RF_GAIN	Right Front Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
CF_GAIN	Center Front Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
LFE_GAIN	LFE Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
LS_GAIN	Left Surround Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
RS_GAIN	Right Surround Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
FR_GAIN	Front Reverberation Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where -7800 correspond to mute. Default value = 0 mB
SR_GAIN	Surround Reverberation Gain Gain in mB (1 milliBell = 0.01 dB) ranging from [–7800 1200] where

	-7800 correspond to mute. Default value = 0 mB

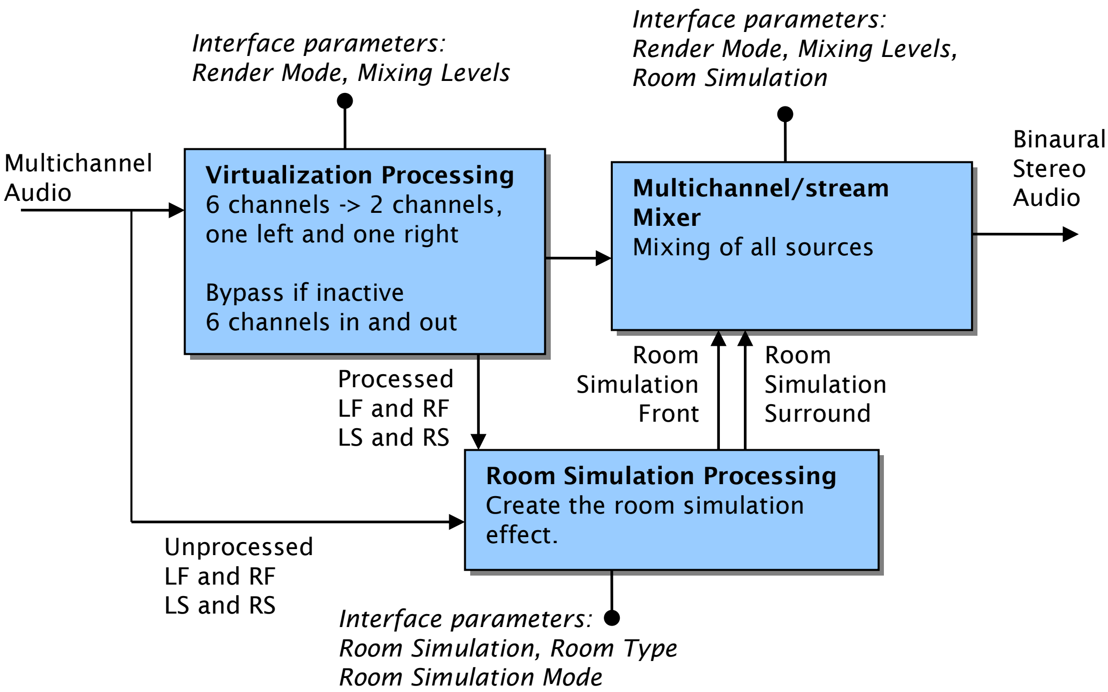
2.2 OMX IL Interface Description

```
typedef enum OMX_AUDIO_VIRTUALSURROUNDMODE
{
   OMX_AUDIO_VirtualSurroundStandardDownmix,    /**< Matrix Downmix without left and right surround mixing */
   OMX_AUDIO_ VirtualSurroundSurroundDownmix,  /**< Matrix Downmix with left and right surround mixing*/
   OMX_AUDIO_ VirtualSurroundSurroundVirtualization /**< 5.1 channel virtualization*/
   OMX_AUDIO_VirtualSurroundModeMax = 0x7FFFFFFF
} OMX_AUDIO_VIRTUALSURROUNDMODE;

typedef enum OMX_AUDIO_VIRTUALSURROUNDROOMSIMULATIONMODE
{
   OMX_AUDIO_VirtualSurroundRoomSimulationMixed,    /**< Room Simulation is based on non-processed front/surround
                                                         audio signals */
   OMX_AUDIO_ VirtualSurroundRoomSimulationVirtualized,  /**< Room Simulation is based on virtualized front/surround
                                                          audio signals*/
} OMX_AUDIO_ VIRTUALSURROUNDROOMSIMULATIONMODE;

typedef enum OMX_AUDIO_VIRTUALSURROUNDROOMSIMULATIONROOMTYPE
{
   OMX_AUDIO_ VirtualSurroundRoomTypeRoom,
   OMX_AUDIO_ VirtualSurroundRoomTypeLivingRoom,
   OMX_AUDIO_ VirtualSurroundRoomTypeAuditorium,
   OMX_AUDIO_ VirtualSurroundRoomTypeConcertHall,
   OMX_AUDIO_ VirtualSurroundRoomTypeArena,
   OMX_AUDIO_ VirtualSurroundRoomTypeSmallRoom,
   OMX_AUDIO_ VirtualSurroundRoomTypeMediumRoom,
   OMX_AUDIO_ VirtualSurroundRoomTypeLargeRoom,
   OMX_AUDIO_ VirtualSurroundRoomTypeMediumHall,
   OMX_AUDIO_ VirtualSurroundRoomTypeLargeHall
} OMX_AUDIO_ VIRTUALSURROUNDROOMSIMULATIONROOMTYPE;

typedef enum OMX_AUDIO_VIRTUALSURROUNDMIXGAIN
{
   OMX_AUDIO_ VirtualSurroundLeftFrontGain,
   OMX_AUDIO_ VirtualSurroundRightFrontGain,
   OMX_AUDIO_ VirtualSurroundCenterFrontGain,
   OMX_AUDIO_ VirtualSurroundLeftSurroundGain,
   OMX_AUDIO_ VirtualSurroundRightSurroundGain,
   OMX_AUDIO_ VirtualSurroundLFEGain,
   OMX_AUDIO_ VirtualSurroundFrontRoomSimulationGain,
   OMX_AUDIO_ VirtualSurroundSurroundRoomSimulationGain
} OMX_AUDIO_VIRTUALSURROUNDMIXGAIN;
```

```
typedef struct OMX_AUDIO_CONFIG_VIRTUALSURROUNDTYPE
{
   OMX_U32 nSize;            /**< size of the structure in bytes */
   OMX_VERSIONTYPE nVersion;  /**< OMX specification version information */
   OMX_U32 nPortIndex;        /**< port that this structure applies to */
   OMX_BOOL bEnable;          /**< Enable/disable for Virtual Surround control */
   OMX_AUDIO_VIRTUALSURROUNDMODE eVirtualSurroundMode; /**< Virtual Surround  algorithm type */
   OMX_BOOL bRoomSimulationFrontEnable /**< Enable/disable room simulation processing for the Front speaker pair (LF
                                                      and RF)*/
   OMX_BOOL bRoomSimulationSurroundEnable /**< Enable/disable room simulation processing for the Surround speaker
                                                         pair (LS and RS)*/
   OMX_AUDIO_ VIRTUALSURROUNDROOMSIMULATIONMODE eRoomSimulationMode; /**< Select the room simulation
                                                                                                    mode*/
   OMX_AUDIO_ VIRTUALSURROUNDROOMSIMULATIONROOMTYPE eRoomSimulationRoomType; /**< Select the room
                                                                                             type where audio is virtualized */
   OMX_AUDIO_VIRTUALSURROUNDMIXGAIN eMixGain; /** < Select which channel to adjust the gain on */
   OMX_S16 sGainValue; /**< Gain value in mB (1/100 dB) */
} OMX_AUDIO_CONFIG_ VIRTUALSURROUNDTYPE;
```

3 Algorithm Detailed Description

The figure below shows an overview of the complete algorithm and where parameters are required.



Consider the following matrix and table of how the algorithm will work according to the parameters.

- Render Mode = MLRMD or MSMD
  The Virtualization Processing block is skipped and multichannel audio is transferred directly into the multichannel mixer.

- Render Mode = MSVD
  The Virtualization Processing block will create 6 stereo audio streams that are input to the mixer.

Room Simulation Mode (Room Simulation = ON)

Render Mode		RSIM_MIXMODE	RSIM_VIZMODE
	MLRMD	Virtualization Processing OFF Reverberation is done using non-processed multichannel audio i.e. Front Reverberation using LF+RF Surround Reverberation using LS+RS	Virtualization Processing OFF Not possible
	MSMD	Virtualization Processing OFF Reverberation is done using non-processed multichannel audio i.e. Front Reverberation using LF+RF Surround Reverberation using LS+RS	Virtualization Processing OFF Not possible
	MSVD	Virtualization Processing ON Reverberation is done using non-processed multichannel audio i.e. Front Reverberation using LF+RF Surround Reverberation using LS+RS	Virtualization Processing ON Reverberation is done using processed multichannel audio i.e. Front Reverberation using Processed_LF+ Processed_RF Surround Reverberation using Processed_LS+ Processed_RS

3.1 Details - Multichannel Mixer

The multichannel mixer will apply gain and mix the virtualized audio signals from the Virtualization Processing block. It also handles the down-mix without virtualization when the Rendering Mode is such that Virtualization Processing block is inactive.

The following formula describe the operation of the multichannel mixer for Rendering Mode = MLRMD and MSMD

$$Out(n) = G_{LF} LF(n) + G_{RF} RF(n) + G_{CF} CF(n) + G_{LFE} LFE(n) +$$
$$G_{LS} LS(n) + G_{RS} RS(n) + G_{FR} Front\,Reverb(n) + G_{SR} Surround\,Reverb(n)$$

Observe that $G_{xy} = XY\_GAIN \cdot ScalingValue\_xy$ and that Out(n) can be either left or right output signal, see below for different scale factors in each channel.

The table below provides the values of the *ScalingValue_xy* for each Rendering Mode and both output channels, denoted Out_Mixer_Left and Out_Mixer_Right. For the reverberation addition there is no pre-scaling value.

ScalingValue_xy	LF	RF	CF	LFE	LS	RS
Out_Mixer_Left	0,2929	0	0,2071	0,2071	0,2929	0
Out_Mixer_Right	0	0,2929	0,2071	0,2071	0	0,2929

The table shows the scaling factor for Rendering Mode = MLRMD. The values of the scaling parameters are taken from ITU-R BS 775 recommendation including normalization with 4.

ScalingValue_xy	LF	RF	CF	LFE	LS	RS
Out_Mixer_Left	0,2626	0	0,1856	0,1856	-0,2144	-0,1516
Out_Mixer_Right	0	0,2626	0,1856	0,1856	0,1516	0,2144

The table shows the scaling factor for Rendering Mode = MSMD. The values of the scaling parameters are taken from Dolby ProLogicII including normalization with 5.

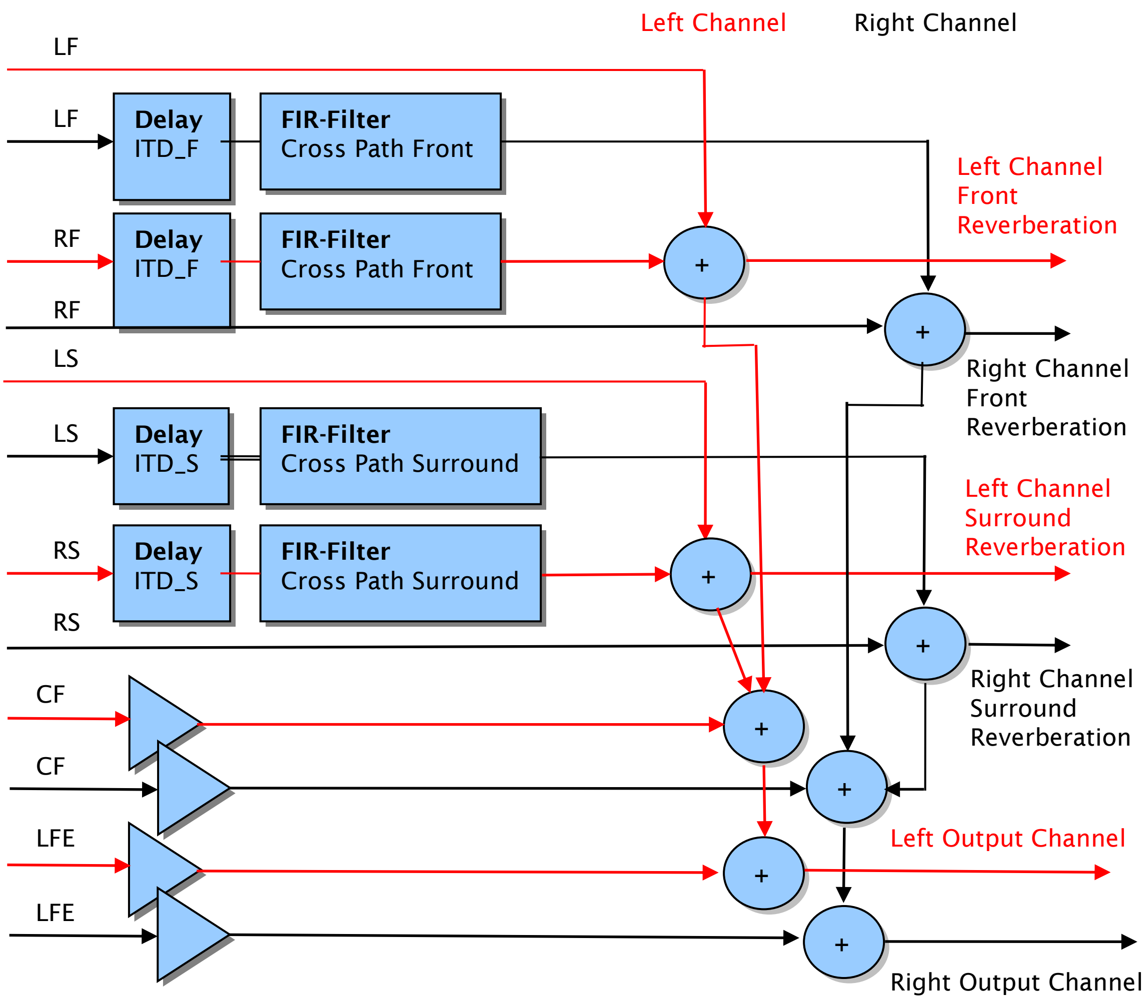
Per-default the mixing gains are 0 mB = 1.0 in linear scale. When changed by a user the mixer will update the corresponding gain $G_{xy} = XY\_GAIN \cdot ScalingValue\_xy$ accordingly (scale the gain once, do not apply two gains to every sample).

The scaling factors in the Rendering Mode = MSVD are embedded in the filter coefficients, see section on Virtualization Processing. In this rendering mode the mixer will only mix the stereo room simulation signals and the virtualized stereo signal which correspond to

$$Out\_Mixer\_Left(n) = Virtualized\_Left(n) + G_{FR} Front\mathrm{Re}verb\_Left(n) + G_{SR} Surround\mathrm{Re}verb(n)$$

$$Out\_Mixer\_Right(n) = Virtualized\_Right(n) + G_{FR} Front\mathrm{Re}verb\_Right(n) + G_{SR} Surround\mathrm{Re}verb\_Right(n)$$

3.2 Details – Virtualization Processing



An overview of the algorithm is shown in the figure below. The red color is to distinguish whether a signal is part of left or right output channel.

Observe the multiple output pairs;
Left and right output channel connected to the multichannel mixer,
Left and right room simulation front to transport to the Room Simulation Component,
Left and right room simulation surround to transport to the Room Simulation Component.

The delays are static as well as the filters and are given below. The gains in the CF and LFE channel paths are all equal to 0.2071. The scaling coefficients for the filter coefficients are equal to 0.2929.

The scaling are chosen to prevent overflow (4 major contributing signals) and to maintain a ratio of approximately 1.5 for low frequencies between the energy in the Front and Surround compared to Center and LFE i.e. a difference of 3 dB in gain factors.

For 48 kHz sample rate:

```
#define ITD_F           20 //Nbr Samples
#define ITD_S           30 //Nbr Samples
#define FilterLength    24

static double FrontCrossPathFilter[FilterLength]=

{1.3350140491, -1.2958213193, 1.6064858581, -1.6951546872,
1.9279427866, -2.3618071067, 1.7158364773, -1.5094340371,
1.4213929267, -0.9242773986, 0.6967906917, -0.0874822265,
-0.2727191849, 0.0177337655, -0.3420586483, 0.4090871737,
-0.2525245360, 0.3038753510, -0.0577015072, -0.0933968168,
-0.1440930239, -0.0071032693, 0.1363469103, -0.0894641511
};

static double SurroundCrossPathFilter[FilterLength] =
{
0.5779864100, -0.2721829886, 0.4419516062, -0.4567398157,
0.5155719975, -0.4839318874, 0.3281144982, -0.2888947143,
0.2361084931, -0.1069127924, 0.1029413038, 0.1021681987,
-0.0845649646, 0.0277488969, -0.1027936522, 0.0383021177,
0.0096673142, 0.0362158017, 0.0969802960, -0.0559862231,
0.0371187357, -0.0416770545, 0.0511471175, -0.0358427400
};
```

Filter coefficients are given without normalization using the scaling coefficients stated above.

3.2.1 For information – Filter Calculations

The filters given above are obtained as follows. From the Ericsson Research database of HR-filters we extract the HR filter corresponding to 15 degrees azimuth angle and 0 degree elevation (one filter for left ear and one filter for right ear). Using these two filters we can position the LF signal and the RF (switch place on left and right filter) signal since the filters are symmetric across azimuth 0 degree.

Doing so will create several resonance frequencies resulting in peaks and dips in the frequency response function of the respective filter.

To remove some coloration of the frequency content i.e. decrease the magnitude and number of peak and dips in the frequency response we consider removing the filter of the direct path to the closest sink i.e. ipsilateral path. And instead of filtering with the cross path filter (contralateral path) we calculate the difference between the filter and use that to filter the cross path signal.

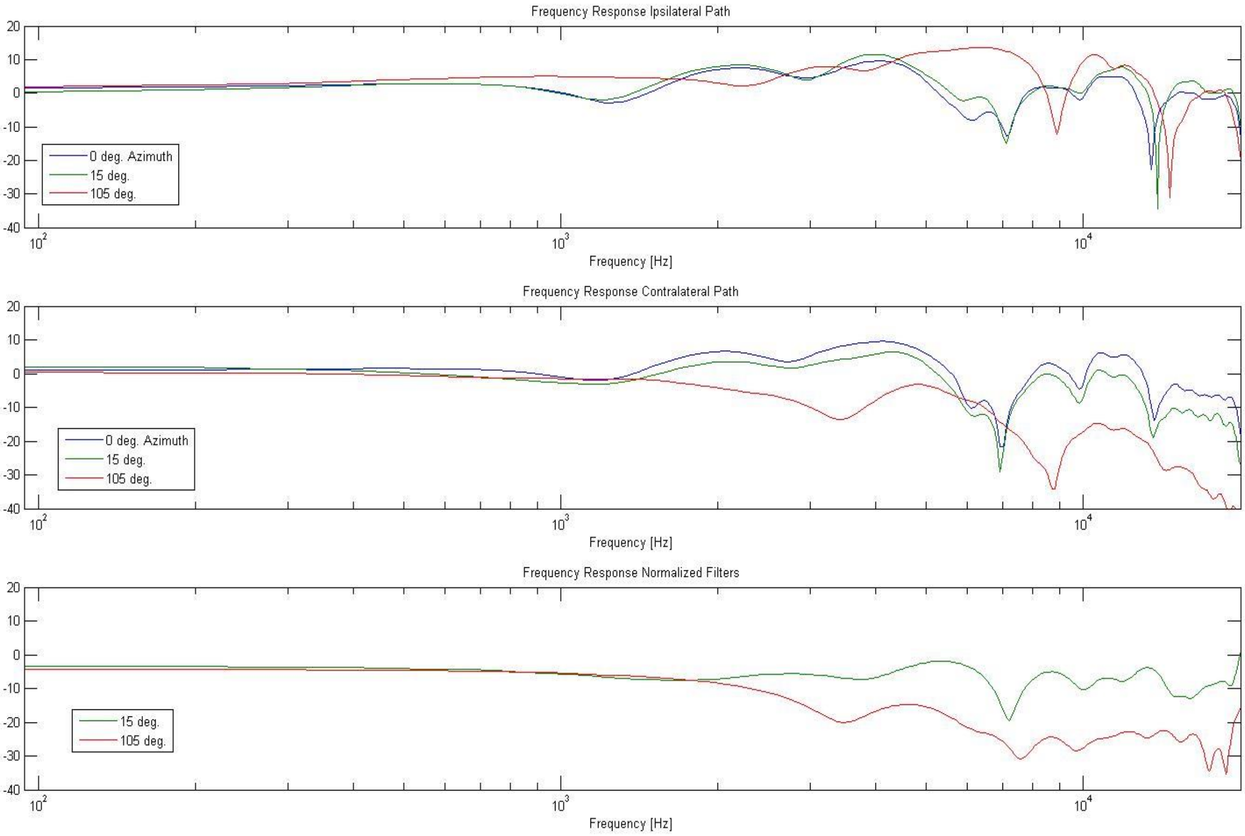
So considering for example the signal in the left and right ear when virtualizing the LF signal,

$$LeftEar(f) = H_d(f)LF(f)$$
$$RightEar(f)) = H_c(f)LF(f)\text{,}$$

that is then approximated by

$$LeftEar(f) = LF(f)$$
$$RightEar(f)) = \frac{H_c(f)}{H_d(f)}LF(f)\text{,}$$

where $H(f) = \frac{H_c(f)}{H_d(f)}$ are the FrontCrossPathFilter. The analogue calculations for the surround channels are made to obtain the SurroundCrossPathFilter and here the azimuth angle is 105 degrees.



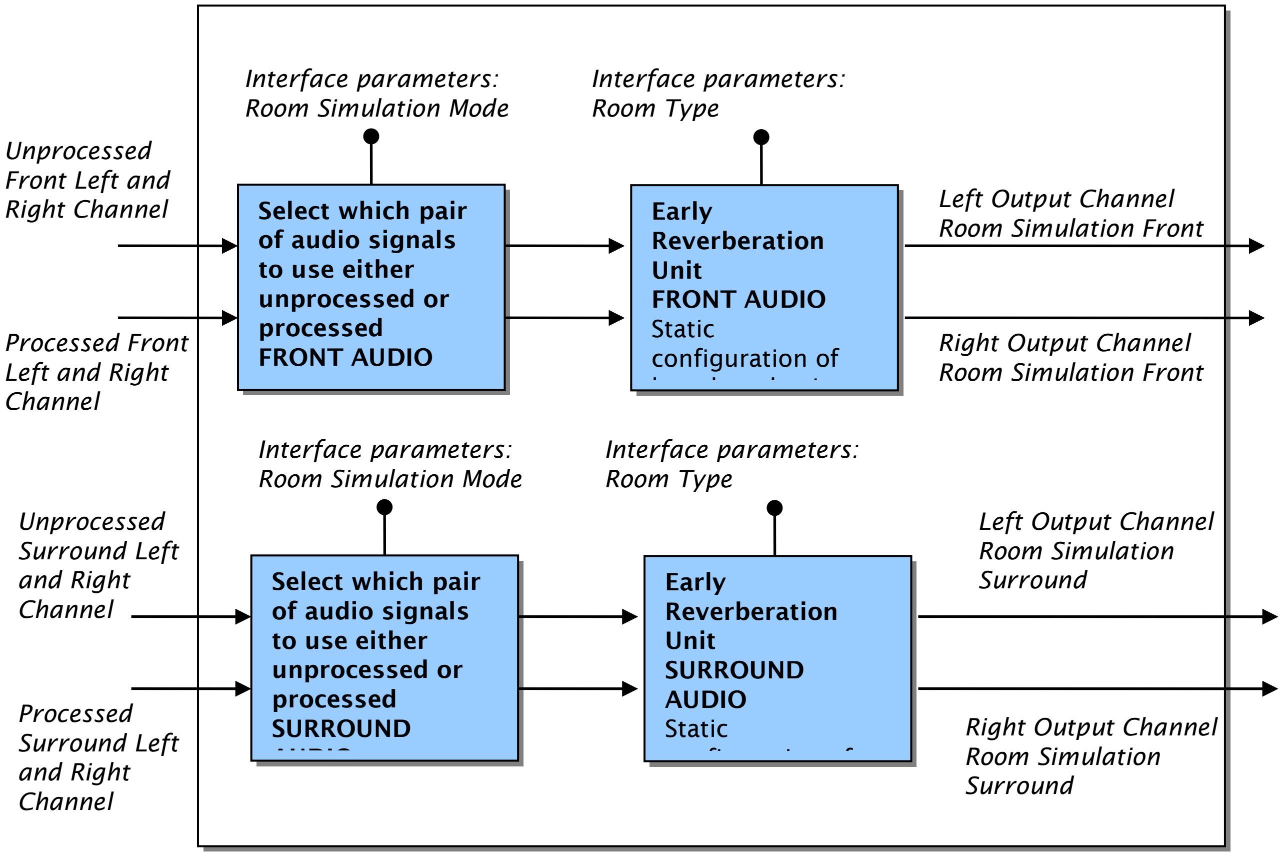
Due to the normalization taken to decrease the frequency impact and also gaining some computational load we will of course loose some virtualization effect. So we can not take the ITD times corresponding to the HR filters.

The chosen ITD times are experimentally obtained during listening tests and are much larger than the true ITD times.

3.3 Details – Room Simulation Processing

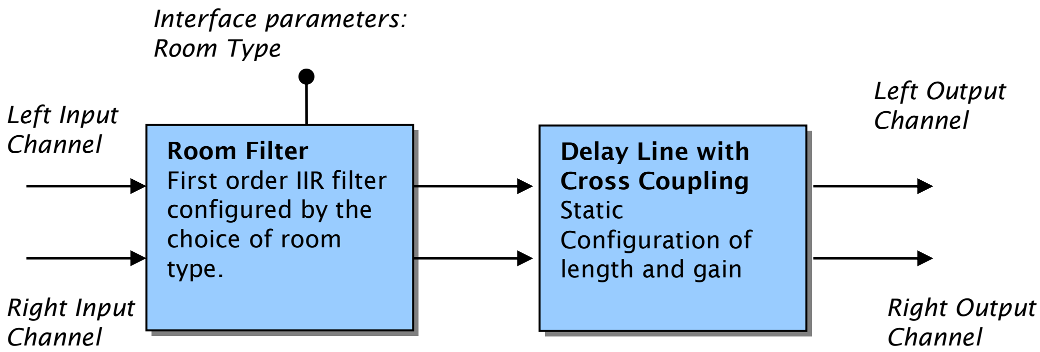
The room simulation is comprised of two basic early reverberation units. One for the pair of front speakers (LF and RF) and one for the pair of surround speakers (LS and RS).

ROOM SIMULATION PROCESSING COMPONENT



Above is an overview picture of the room simulation processing component.

The early reverberation unit is described below.



The Room Filter is configured from the choice of Room Type. The delays and gains of the delay line are static and not configurable.

3.3.1 Room Filter

The room filter is a first order IIR filter $H(z) = \frac{b_0}{1 - a_1 z^{-1}}$ where

$b_0 = k(1-a)$.

From the room type the following parameters are read:

lRoom – Intensity level at low frequencies
lRoomHF – attenuation at high frequencies (relative to lRoom)
lReflections – Intensity of early reflections
flHFReference – Reference high frequency

Values for each room type are listed below in section Room Simulation Parameters
The normalization constant $k = 10^{(\mathrm{lRoom}+1000+\mathrm{lReflections})/20}$ is calculated to normalize the total amplification in the room filter.

Further, $\omega = 2\pi \frac{flHF\,\mathrm{Re}\,ference}{fs}$, where *fs* is the sample rate in Hz,

$g = 10^{lRoomHF/1000}$ and

$$a = \frac{1 - g\cos(\omega) - \sqrt{2g(1-\cos(\omega)) - g^2(1-\cos(\omega)^2)}}{1-g} \quad .$$

3.3.2 Delay Line with Cross Coupling

There are two delay lines in one reverberation unit, one for left channel and one for right channel. The delay lines are static and equal to longest delay plus the frame size i.e.,

```
#define MAX _DELAY_LINE 1201
 double LeftDelayLine[MAX _DELAY_LINE + FrameSize];
double RightDelayLine[MAX _DELAY_LINE + FrameSize];
```

In each delay line there are 8 tapping points for input to output (direct path). These are denoted *SFX_ER_h_ll_delays* and *SFX_ER_h_rr_delays* and 5 tapping points for second input to output (cross path), denoted *SFX_ER_h_rl_delays and SFX_ER_h_lr_delays.*

The gain of each tap in the delay lines are given by the coefficients below denoted *SFX_ER_h_ll, SFX_ER_h_rr, SFX_ER_h_rl* and *SFX_ER_h_lr.*

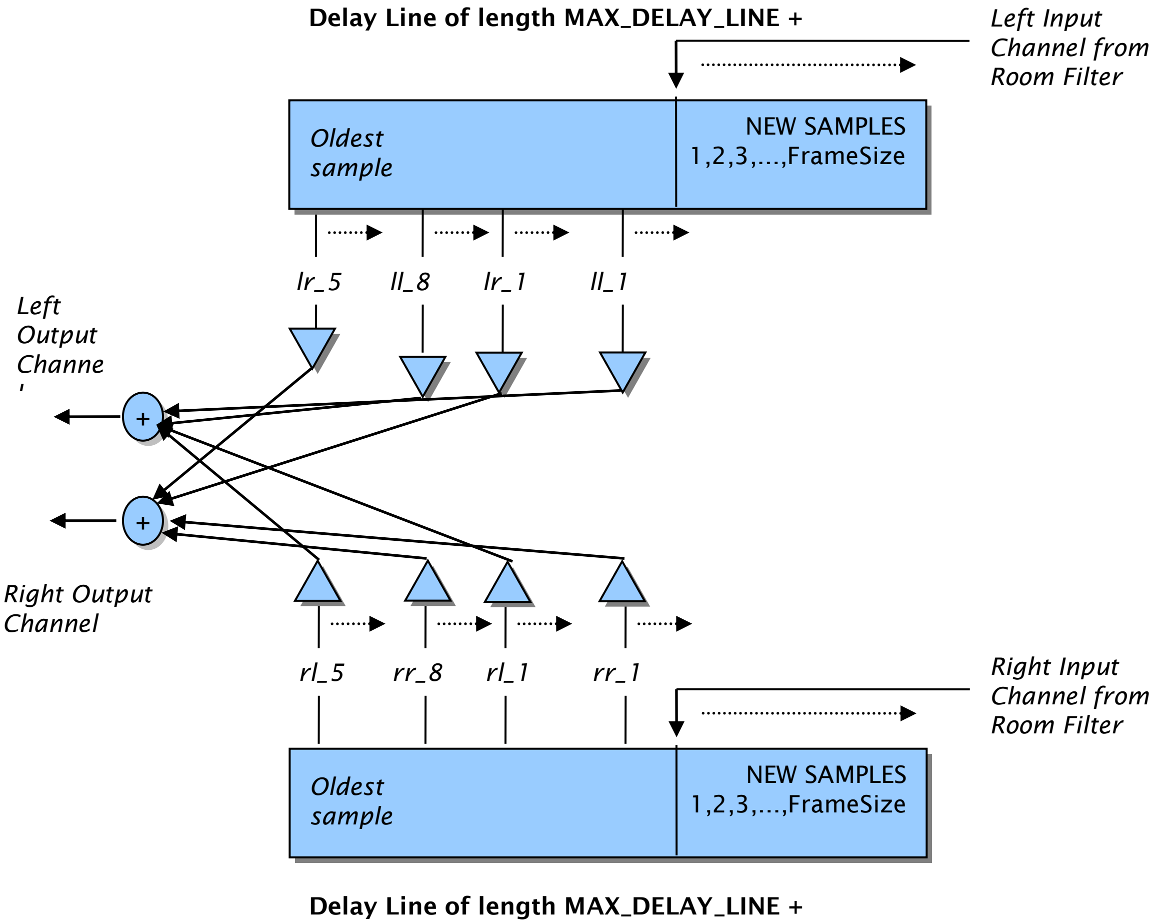
```
// The delay line taps for the adjustment filters.
// All delays are given in samples @ 48 kHz.

static const double SFX_ER_h_ll[8] = {0.2680, -0.2256, 0.2507, 0.1345, 0.3016, -0.0907,
0.3021, 0.2262};
static const short  SFX_ER_h_ll_delays[8] = {75, 160, 220, 390, 450, 620, 700, 1050};
static const double SFX_ER_h_rr[8] = {0.2801, -0.2244, 0.1587, 0.2299, 0.1414, -
0.1069, 0.0696, 0.0894};
static const short  SFX_ER_h_rr_delays[8] = {75, 160, 225, 384, 458, 618, 720, 1061};
static const double SFX_ER_h_rl[5] = {0.2000, 0.2993, -0.0764, -0.0860, -0.1552};
static const short  SFX_ER_h_rl_delays[5] = {110, 299, 560, 902, 1198};
static const double SFX_ER_h_lr[8] = {0.2000, 0.2820, -0.1807, -0.0978, -0.1181};
static const short  SFX_ER_h_lr_delays[8] = {110, 300, 550, 900, 1201};
```

Updating the Delay Lines.

When the output signals of left and right channel has been created the delay lines must be updated i.e. shifted to the left so that the oldest samples are removed and space is made for the next new frame of audio.

1. This requires a memcopy from the right part of the delay line to the left part of the delay line of DELAY_LINE_LENGTH – FrameSize number of samples, see figure below. (This is how its done in the reference code)

2. Instead of fixed positions of delay line taps we could keep the distance fixed and let the relative position rotate, as a circular buffer. However, doing so we need to keep track of delay line start and end and at the same time considering that using NEON the data should be stored in a linear storage. Hence, it might be that utilizing NEON fully is more beneficial than avoiding memcopy operations since there are a lot of (at least 2*13*FrameSize) multiply and accumulate operations to do before two memcopy operations of MAX_DELAY_LINE number of samples.



3.3.3 Room Simulation Parameters

```
typedef struct{
  long lRoom;                  // [-10000, 0]      default: -10000 mB
  long lRoomHF;                // [-10000, 0]      default: 0 mB
  double flRoomRolloffFactor;  // [0.0, 10.0]      default: 0.0
  double flDecayTime;          // [0.1, 20.0]      default: 1.0 s
  double flDecayHFRatio;       // [0.1, 2.0]       default: 0.5
  long lReflections;           // [-10000, 1000]   default: -10000 mB
  double flReflectionsDelay;   // [0.0, 0.3]       default: 0.02 s
  long lReverb;                // [-10000, 2000]   default: -10000 mB
  double flReverbDelay;        // [0.0, 0.1]       default: 0.04 s
  double flDiffusion  ;        // [0.0, 100.0]     default: 100.0 %
  double flDensity;            // [0.0, 100.0]     default: 100.0 %
  double flHFReference;        // [20.0, 20000.0]  default: 5000.0 Hz}
SFX_Reverb_RoomParameters_t;

static const SFX_Reverb_RoomParameters_t SFX_Reverb_RoomParameters[] = {
  // I3DL2_ROOM_DEFAULT
  {-10000,    0, 0.0,  1.00, 0.50, -10000, 0.020, -10000, 0.040, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_ROOM
  { -1000, -454, 0.0,  0.40, 0.83,      0, 0.002,     53, 0.003, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_LIVINGROOM
  { -1000,-3000, 0.0,  0.50, 0.10,      0, 0.003,  -1104, 0.004, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_AUDITORIUM
  { -1000, -476, 0.0,  4.32, 0.59,      0, 0.020,   -389, 0.030, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_CONCERTHALL
  { -1000, -500, 0.0,  3.92, 0.70,      0, 0.020,   -102, 0.029, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_ARENA
  { -1000, -698, 0.0,  7.24, 0.33,      0, 0.020,   -300, 0.030, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_SMALLROOM
  { -1000, -600, 0.0,  1.10, 0.83,      0, 0.005,    500, 0.010, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_MEDIUMROOM
  { -1000, -600, 0.0,  1.30, 0.83,      0, 0.010,    300, 0.020, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_LARGEROOM
  { -1000, -600, 0.0,  1.50, 0.83,      0, 0.020,    200, 0.040, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_MEDIUMHALL
  { -1000, -600, 0.0,  1.80, 0.70,      0, 0.015,    200, 0.030, 100.0, 100.0, 5000.0},

  // I3DL2_ROOM_LARGEHALL
  { -1000, -600, 0.0,  1.80, 0.70,      0, 0.030,    100, 0.060, 100.0, 100.0, 5000.0},

};
```

4.0 VIRTUAL SURROUND - API

4.1 STRUCTURES

4.1.1 t_host_effect_config

```
t_host_effect_config
--------------------
unsigned int block_size
t_host_effect_format infmt
t_host_effect_format outfmt


t_host_effect_format
--------------------
unsigned short freq (7)
unsigned short nof_channels (6 on input, 2 on output)
unsigned short nof_bits_per_sample (16)
unsigned short headroom
t_bool interleaved (0 or 1)


sample_freq
-----------
0=UNKNOWN
1=192kHz
2=176.4kHz
3=128kHz
4=96kHz
5=88.2kHz
6=64kHz
7=48kHz
8=44.1kHz
9=32kHz
10=24kHz
11=22.05kHz
12=16kHz
13=12kHz
14=11.025kHz
15=8kHz
16=7.2kHz
17=LAST
```

```
4.1.2
  void t_vs_handle
  ----------------
```

4.1.3 t_vs_allocation_params

```
-----------------------
unsigned short nof_channels (6)
unsigned short sample_rate (48000)
int frame_size (any number of samples)
t_bool interleave (0=linear, 1=interleaved)
t_audio_channel_type channel_mapping[8] (1, 2, 3, 6, 4, 5, 0, 0)


t_audio_channel_type
--------------------
AUDIO_CHANNEL_NONE=0
AUDIO_CHANNEL_LF=1
AUDIO_CHANNEL_RF=2
AUDIO_CHANNEL_CF=3
AUDIO_CHANNEL_LS=4
AUDIO_CHANNEL_RS=5
AUDIO_CHANNEL_LFE=6
AUDIO_CHANNEL_CS=7
AUDIO_CHANNEL_LR=8
AUDIO_CHANNEL_RR=9
AUDIO_CHANNEL_MAX=0x7fffffff
```

4.1.4 t_virtual_surround_config

```
--------------------------
t_bool enable(0 or 1)
t_bool room_simulation_front(0 or 1)
t_bool room_simulation_surround(0 or 1)
t_virtual_surround_mode(1,2, or 3)
t_room_simulation_mode(1 or 2)
t_room_simulation_room_type(0, 1, .. 10)
t_mix_gain(-7800 ... +1200)


t_virtual_surround_mode
-----------------------
STANDARD_DOWN_MIX=1
SURROUND_DOWN_MIX=2
SURROUND_VIRTUALIZATION=3

t_room_simulation_mode
----------------------
MIXED=1
VIRTUALIZED=2

t_room_simulation_room_type
---------------------------
DEFAULT=0
ROOM=1
LIVING_ROOM=2
AUDITORIUM=3
CONCERT_HALL=4
ARENA=5
SMALL_ROOM=6
MEDIUM_ROOM=7
LARGE_ROOM=8
MEDIUM_HALL=9
LARGE_HALL=10

t_mix_gain gains
----------------
short left_front_gain (-7800 to 1200 milliBell)
short right_front_gain ”
short center_front_gain ”
short left_surround_gain ”
short right_surround_gain ”
short low_frequency_effects_gain ”
short front_room_simulation_gain ”
short surround_room_simulation_gain ”
```

4.2 FUNCTIONS

4.2.1 int open
(t_host_effect_config *config)

4.2.4 int setConfig
(t_virtual_surround_config config)

4.2.3 void reset
(t_effect_reset_reason reason)

4.2.4 void process
(void)

4.2.5 void close
(void)

4.3 API PROCEDURE

4.3.1 Create and init global variables

4.3.1.1 direct API interface variables

```
t_bool mEnable=FALSE
unsigned short mInChannels,mOutChannels,mBlockSize,mBufferSize=0
t_host_effect_config *config=0
t_virtual_surround_config configparams=0
short inputBuffer[ ] outputBuffer[ ]
```

4.3.1.2 internal variables needed and programmed by effect functions

```
t_vs_handle *pHandle=NULL
t_vs_allocation_params mParams=0
t_vs_param mConfig=0
```

4.3.2 Configuration set up

```
set config->block_size
set config->infmt and outfmt values
```

4.3.3 open(config)

4.3.4 Parameter set up (ref 7.0 for tuning values)

```
set configparams->
enable,room_simulation_front,room_simulation_surround,virtual_surround_mode,
room_type,room_simulation_mode

set configparams->gains->left_front, right_front, center_front, left_surround,
right_surround, low_frequency_effects, front_room_simulation_gain,
surround_simulation_gain
```

4.3.5 setconfig(configparams)

4.3.6 reset()

4.3.7 fill input buffer samples, execute process(), empty output buffer samples until finished

4.3.8 close()

5.0 MIPS

Virtual Surround standalone test results using ca9sim (Cortex A9 simulator)

test	MIPS	CONTENTS
vs_virtualization	13.7	speech
vs_virtualization_room_no_gain	42.5	speech
vs_virtualization_room_max_gain	42.5	speech
vs_virtualization_room_max_front_gain	28.2	speech
vs_virtualization_room_mixed_no_gain	42.5	speech
vs_virtualization_room_mixed_max_surround_gain	28.2	speech
vs_standard_downmix	3.1	speech
vs_standard_downmix_room_no_gain	31.9	speech
vs_standard_downmix_room_max_gain	31.9	speech
vs_standard_downmix_room_max_surround_gain	17.5	speech
vs_surround_downmix	3.2	speech
vs_surround_downmix_room_no_gain	31.9	speech
vs_surrround_downmix_room_max_gain	31.9	speech
vs_surrround_downmix_room_max_front_gain	17.5	speech
vs_standard_downmix	3.1	music Buena
vs_virtualization_room_no_gain_concert_hall	42.5	music Buena
vs_standard_downmix	3.1	music Miles
vs_virtualization_room_no_gain_concert_hall	42.5	music Miles
vs_standard_downmix	3.1	music Hancock
vs_virtualization	13.7	music Hancock
vs_virtualization_room_no_gain_concert_hall	42.5	music Hancock
vs_standard_downmix	3.1	music Connick
vs_virtualization	13.7	music Connick
vs_virtualization_room_no_gain_concert_hall	42.5	music Connick
vs_virtualization_room_no_gain_concert_hall	42.5	music Connick1
vs_standard_downmix	3.1	music Clapton
vs_virtualization	13.7	music Clapton
vs_standard_downmix	3.1	DVD Pourpres
vs_virtualization	13.7	DVD Pourpres
vs_standard_downmix	3.1	DVD StarWars3
vs_virtualization_room_no_gain_living_room	42.5	DVD StarWars3
vs_standard_downmix	3.1	DVD Transformers
vs_virtualization	13.7	DVD Transformers
vs_virtualization_room_no_gain_living_room	42.5	DVD Transformers

6.0 Virtual Surround Listening Test Survey

The object of the listening test survey was to obtain preferences of a limited number of subjects (engineers, musicians, and non-technical personnel) on the following 3 virtual surround effect configurations:

1. **standard down mixing**
2. **virtualization**
3. **virtualization with room simulation.**

On:

1. **music**
2. **DVD films**
3. **Spatially moving audio message.**

6.1 VIRTUAL SURROUND LISTENING TESTS FOR MUSIC (18 June 2012)

6.1.1 SURROUND_VIRTUALIZATION vs. STANDARD_DOWN_MIX (Clapton)

Prefer extra effect	xxxx
Prefer simple down mix	xxxxx

6.1.2 SURROUND_VIRTUALIZATION_WITH_ROOM SIMULATION vs. STANDARD_DOWN_MIX (BuenaVista, Connick, Miles)

Prefer extra effect	xxx,	xxxxxxx,	xxxxx
Prefer simple down mix	xxxxxx,	xxxxxx,	xxxx

6.1.3 SURROUND_VIRTUALIZATION_WITH_ROOM_SIMULATION vs. SURROUND_VIRTUALIZATION (Hancock)

Prefer extra room simulation effect	xxxxxxx
Prefer only virtualization	xxx

6.1.4 Total

Prefer extra effects	26
Don't prefer extra effects	23

6.2 VIRTUAL SURROUND LISTENING TESTS FOR DVD FILM (21 June 2012)

SURROUND_VIRTUALIZATION_WITH_ROOM SIMULATION vs. STANDARD_DOWN_MIX (Transformers)

Prefer extra effect	xxxxxx
Prefer only simple down mix	xxxxxx

Total

Prefer extra effects	6
Don't prefer extra effects	6

6.3 VIRTUAL SURROUND LISTENING TESTS FOR A SPATIALLY MOVING AUDIO MESSAGE (25 June 2012)

6.3.1 SURROUND_VIRTUALIZATION vs. STANDARD_DOWN_MIX

Prefer extra effect	xxxxxxxxxxxx
Prefer only simple down mix	x

6.3.2 SURROUND_VIRTUALIZATION_WITH_ROOM SIMULATION vs. SURROUND_VIRTUALIZATION

Prefer virtualization room simulation effect	xx
Prefer virtualization only	xxxxxxxxxx

6.4 Conclusion

Music: Roughly half the listeners prefer only down mixing vs. virtualization, but if virtualization is preferred there seems to be a slight additional preference to room simulation. Whether the subjects were musicians, engineers, or non-technical personnel, there seem to be roughly the same results.

DVD Movie: Half the listeners prefer only down mixing vs. virtualization with room simulation.

Spatially Moving Audio Message: Almost all prefer virtualization but not the additional room simulation.

7.0 TUNING

Typical values for 3 tuning configurations.

7.1 <u>STANDARD DOWN MIX</u> config for all types of audio signals and low MIPS count (3.1 MIPS)

```
Configparams->enable=1
Configparams->room_simulation_front=0
Configparams->room_simulation_surround=0
Configparams->virtual_surround_mode=1
Configparams->room_simulation_mode=1
Configparams->room_type=0
Configparams->gains.left_front_gain=1200
Configparams->gains.right_front_gain=1200
Configparams->gains.center_front_gain=1200
Configparams->gains.left_surround_gain=1200
Configparams->gains.right_surround_gain=1200
Configparams->gains.low_frequency_effects_gain=0
Configparams->gains.front_room_simulation_gain=0
Configparams->gains.surround_room_simulation_gain=0
```

7.2 <u>SURROUND VIRTUALIZATION</u> config for all types of audio signals and compromise between added virtualization effect and relatively low MIPS count (13.7 MIPS)

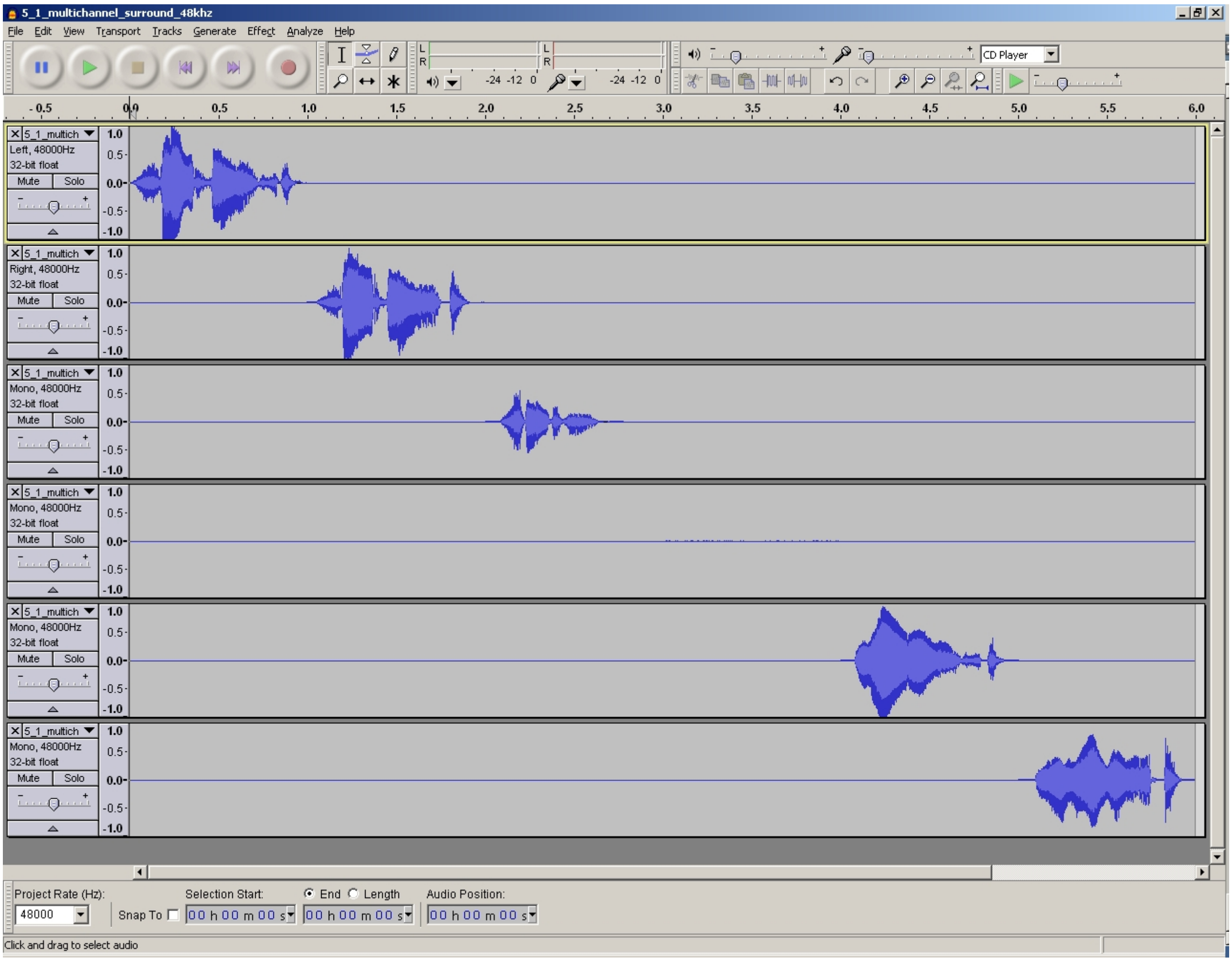
```
Configparams->enable=1
Configparams->room_simulation_front=0
Configparams->room_simulation_surround=0
Configparams->virtual_surround_mode=3
Configparams->room_simulation_mode=1
Configparams->room_type=0
Configparams->gains.left_front_gain=0
Configparams->gains.right_front_gain=0
Configparams->gains.center_front_gain=0
Configparams->gains.left_surround_gain=0
Configparams->gains.right_surround_gain=0
Configparams->gains.low_frequency_effects_gain=0
Configparams->gains.front_room_simulation_gain=0
Configparams->gains.surround_room_simulation_gain=0
```

7.3 SURROUND VIRTUALIZATION WITH ROOM SIMULATION

config especially for music but with a relatively high MIPS count (42.5 MIPS)

```
Configparams->enable=1
Configparams->room_simulation_front=1
Configparams->room_simulation_surround=1
Configparams->virtual_surround_mode=3
Configparams->room_simulation_mode=1
Configparams->room_type=4(CONCERT_HALL)
Configparams->gains.left_front_gain=0
Configparams->gains.right_front_gain=0
Configparams->gains.center_front_gain=0
Configparams->gains.left_surround_gain=0
Configparams->gains.right_surround_gain=0
Configparams->gains.low_frequency_effects_gain=0
Configparams->gains.front_room_simulation_gain=0
Configparams->gains.surround_room_simulation_gain=0
```

7.4 Multi-channel 5.1 input signal (Spatially moving audio message).



7.5 Display of the corresponding 2 Binaural output signals tuned as above for the Standard Down Mix configuration and then for the Surround Virtualization configuration (respectively).