# DSP 音效设置接口文档

## 音频播放通路及内部处理流程

下图为010 项目总体音频通路：

I2Sout

Audio Ref

EQ (HPF)

&

Balance

&

Delay

DSP

Mixer

8257 Music

Aout1L

Rear Aout

Aout1R

Front Aout

Aout2l

Radio

Aout2R

Sub LPF &

Volume &

delay

Aout3L

S

P

\_

A

O

U

T

1

L

D

S

P

\_

A

O

U

T

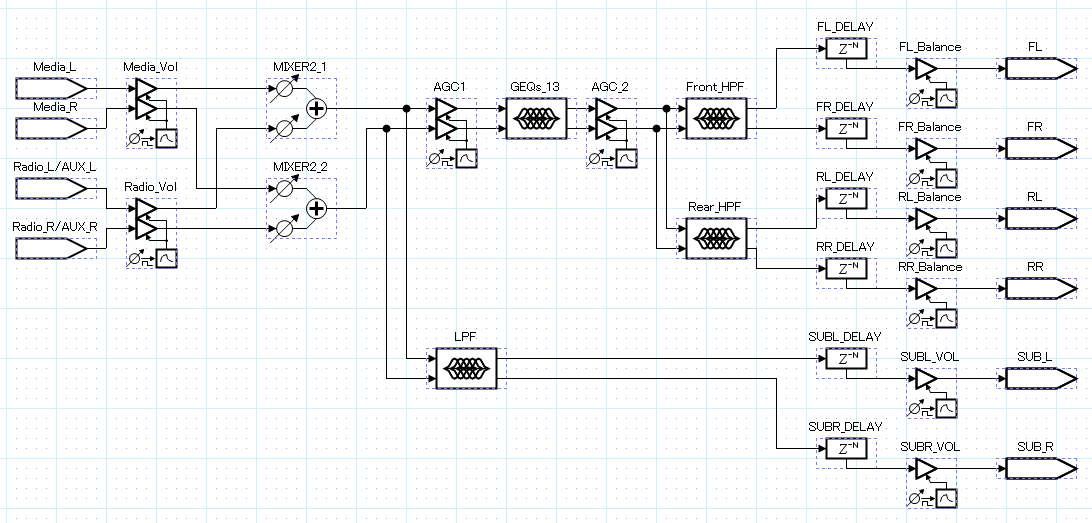
AUX

SubWoof

Aout3R

MiniDSP

下图为mini Dsp 内部设计框图：



简要说明：

1. Music 通路包含android 系统所有声音，此通路必须保持常开；
2. Radio & AUX通路在使用时通过切换开关开启；
3. SubWoof 通道通过SetPEQFilter 接口进行调节；
4. 思必驰参考音通过I2S out回路采集,I2S0声音录制可参考

[\\172.168.1.102\软件工程\ac8257\_010项目\OS项目分享\音频\audio\_ref.rar](W:\\ac8257_010项目\\OS项目分享\\音频\\audio_ref.rar)

文件中MainActivity.java 中 line248 的定义参数进行录制；

|  |
| --- |
| mRecord = new AudioRecord(1998/\*MediaRecorder.AudioSource.MIC\*/,  44100,  AudioFormat.CHANNEL\_IN\_STEREO,  AudioFormat.ENCODING\_PCM\_16BIT,  mixBufSize); |

## 接口介绍

源码路径：hardware/interface/dspeffect/1.0/IDspeffect.hal;

package android.hardware.dspeffect@1.0;

interface IDspeffect {

SetEQ(int32\_t channel, double freq, double Q, double Gain) generates(bool retval);

SetPEQFilter(int32\_t Channel,GEQ Type,double freq,double Q,double Gain) generates(bool retval);

SetChannel(int32\_t Channel) generates(bool retval);

SetMute(int32\_t Mute) generates(bool retval);

SetDelay(int32\_t Channel,int32\_t DelayValue) generates(bool retval);

SetLoudness(int32\_t Bypass,Loudness Type,double freq, double Q,double Gain,double Offset) generates(bool retval);

SetMixer(double Stream1Vol,double Stream2Vol) generates(bool retval);

SetChannelVolume(int32\_t Channel,double VolumeValue) generates(bool retval);

SetStreamVolume(StreamChannel Stream,double StreamVolume) generates(bool retval);

SetBalance(int32\_t Channel,double BalanceValue) generates(bool retval);

SetStreamChannel(int32\_t Channel) generates(bool retval);

SetDspPhaseSwitch(int32\_t Channel,int32\_t Switch) generates(bool retval);

};

### 均衡器设置(GEQ)

**Table1 SetEQ(int32\_t channel,int\_t Number, double freq, double Q, double Gain) ;**

|  |  |
| --- | --- |
| **函数原型** | int SetEQ(int32\_t channel, int\_32\_t Number, double freq, double Q, double Gain) generates(bool retval); |
| **参数** | 1. Channel: enum Channel = {ALL = 0 ,FL,FR,RL,RR,SWL,SWR};//控制通道 2. Number : 0 ~12 //对应频段 3. freq: int EQ\_VAL[] ={ //均衡器13段各个频点值   50, 80, 125,200,315, 500, 800,  1250,2000, 3150,5000, 8000, 12500  };   1. Q: (>= 0)//Q值设置原则上根据频段数配置，用户可按照自身需求进行调节 2. GAIN (-15 ~ 15)//表示均衡器单频段增益值 |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 通过设置每个频段增益值和Q值，补偿高、中、低音频率 |
| **示例代码** | 如设置全局通道，第一频段，频点值为800 ，Q为1.4142，Gain 为5db  ret = SetEQ(ALL, 0，800, 1.4142, 5); |
| **注意** | 由于调节eq 瞬间单频点响应更改，可能会产生爆音，建议调节前后进行mute/unmute处理  如调节单频点，应先mute,待调节完成再进行unmute  如需同时调节多频点，应先mute，待所有频点调节完成再进行unmute |

### LPF & HPF设置

**Table2 SetPEQFilter(int32\_t Channel,GEQ Type,double freq,double Q,double Gain，int32\_t enable);**

|  |  |
| --- | --- |
| **函数原型** | int SetPEQFilter(int32\_t Channel,GEQ Type,double freq,double Q,double Gain，int32\_t enable); |
| **参数** | 1.Channel: enum PEQFilter:int32\_t = {FRONT=1,REAR,SUBWOOF};//控制通道  2.TYPE: //PEQ类型设置  enum GEQ:int32\_t = {  GEQ\_THRH = 1,  GEQ\_LPF, //重低音默认设置此滤波器  GEQ\_HPF, //前后排使用此滤波器  GEQ\_BPF,  GEQ\_1stLPF,  GEQ\_1stHPF,  GEQ\_NOTCH,  GEQ\_PEAKING\_EQ,  GEQ\_LOW\_SHELF,  GEQ\_HIGH\_SHELF,  GEQ\_APF};  3.freq: HPF(> 200) LPF(20 ~ 200)  4.Q:( >0) //斜率调节  5.Gain: default 0  6.enable:  0: not enable //关闭HPF/LPF  1:enable //打开HPF/LPF |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 通过设置滤波器类型、频点、斜率，滤出所需要高频或低频信号 |
| **示例代码** | 如设置前排，高通滤波200 频点，Q为1.4142  ret = SetPEQFilter(FRONT, GEQ\_HPF, 200, 1.4142, 0, 1);  如关闭后排高通滤波  ret = SetPEQFilter(REAR, FRONT, GEQ\_HPF, 200, 1.4142, 0, 0); |
| **注意** | 由于调节HPF/LPF 瞬间单频点响应更改，可能会产生爆音，建议调节前后进行mute/unmute处理 |

### 喇叭配置

**Table3 SetChannel(int32\_t Channel,int32\_t enable);**

|  |  |
| --- | --- |
| **函数原型** | int SetChannel(int32\_t Channel,int32\_t enable); |
| **参数** | 1. Channel:   enum AudioSpeakerLayout :int32\_t{  SPEAKER\_LAYOUT\_FRONT,//前排  SPEAKER\_LAYOUT\_REAR,//后排  SPEAKER\_LAYOUT\_SUBWOOF//重低音  };  2.Enable:  0: not enable //关闭通道  1: enable //打开通道 |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 配置喇叭输出开关 |
| **示例代码** | 如关闭后排输出  ret = SetChannel(SPEAKER\_LAYOUT\_REAR，0); |
| **注意** | 此接口由控制dsp dac 寄存器实现，开关硬件接口本身存在爆音，为避免爆音，调节前后必须进行mute/unmute处理 |

### Mute 控制

**Table4 SetMute(int32\_t Mute) ;**

|  |  |
| --- | --- |
| **函数原型** | int SetMute(int32\_t Mute) ; |
| **参数** | Mute  false: //disable  true: //enable |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | Mute 全局声音 |
| **示例代码** | 如mute 全局声音  ret = SetMute(true); |
| **注意** | 此接口直接控制dsp dac mute寄存器 |

### Delay 控制

**Table5 SetDelay(int32\_t Channel,int32\_t DelayValue);**

|  |  |
| --- | --- |
| **函数原型** | int SetDelay(int32\_t Channel,int32\_t DelayValue); |
| **参数** | 1. Channel: //声道选择   enum Channel = {ALL,FL,FR,RL,RR,SWL,SWR};   1. DelayValue: 0 ~ 200(0 ~ 20ms)//最大可延时20ms(6.8m),步进0.1ms |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 设置各通道延时输出时间 |
| **示例代码** | 如FL 延时1ms  ret = SetDelay(FL, 10); |
| **注意** | 声道延时调节后,开机设置即可,不必频繁设置  如需频繁设置,调节前后必须进行mute/unmute处理 |

### Balance 控制(输出通道音量控制)

**Table6 SetBalance(int32\_t Channel,double BalanceValue);**

|  |  |
| --- | --- |
| **函数原型** | int SetBalance(int32\_t Channel,double BalanceValue); |
| **参数** | 1. Channel: //声道选择   enum Channel = {ALL,FL,FR,RL,RR,SWL,SWR};   1. BalanceValue: //声道增益值   范围：0 ~ -144 |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 设置单个输出声道增益 |
| **示例代码** | 如FL 增益为-10  ret = SetBalance(FL, -10); |
| **注意** | 设置平衡时，四个喇叭都需要设置，单个喇叭不能超过0db  Balance固件调节自身无fade in/fade out，调节前后需进行mute/unmute处理 |

### 输入声道切换

**Table7 SetStreamChannel(int32\_t Channel);**

|  |  |
| --- | --- |
| **函数原型** | int SetStreamChannel(int32\_t Channel); |
| **参数** | Channel: //选择输入通道  enum Stream = {Music=1,radio,Aux}; |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 音频输入通道选择 |
| **示例代码** | 如切换至收音机通道  ret = SetStreamChannel(radio); |
| **注意** | 切换通道由配置dsp adc 寄存器实现，开关硬件接口本身存在爆音，为避免爆音，调节前后必须进行mute/unmute处理(已在hidl 中处理) |

### 输入通道音量控制

**Table8 SetStreamVolume(StreamChannel Stream,double StreamVolume);**

|  |  |
| --- | --- |
| **函数原型** | int SetStreamVolume(StreamChannel Stream,double StreamVolume); |
| **参数** | 1. Channel:   enum StreamChannel:int32\_t {  Navi = 0,  Music,  Radio,  Aux};   1. StreamVolume:   \* radio /aux通道已匹配music音量0 ~ 30  \* 驱动可调节12~ -144db,OFF: 0xff [mute] |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 调节输入通道增益 |
| **示例代码** | 如收音机通道增益设置为8  ret = SetStreamVolume(radio，8); |

### 通道正反向控制

**Table9 SetDspPhaseSwitch(int32\_t Channel,int32\_t Switch);**

|  |  |
| --- | --- |
| **函数原型** | int SetDspPhaseSwitch(int32\_t Channel,int32\_t Switch); |
| **参数** | 1. Channel://声道选择   enum Channel = {ALL,FL,FR,RL,RR,SWL,SWR};   1. Switch：   0-disable,1 - enable |
| **返回值** | =0:Successful implementation; !=0:Error in execution |
| **作用** | 通道正反向调节 |
| **示例代码** | 如SWL反向设置  ret = SetDspPhaseSwitch(SWL, true); |

**注意：**由于每次开机音效app会进行数据下发，所有经由音效app 处理的功能调节时均由应用做mute/unmute操作，底层不予处理