

1 EQ 均衡器

1.1 EQ 均衡器简介

EQ 是 Equalizer 的缩写，它的作用就是调整各频段信号的增益值。在音响器材中，均衡器是一种可以分别调节各种频率成分电信号放大量的电子设备，通过对各种不同频率的电信号的调节来调整音色、调整声场及抑制声反馈等作用。

运用数字均衡器组成的均衡器均称为数字均衡器。数字均衡器可以由低通、带通以及高通滤波器组成。

人耳听音的频率范围为 20Hz 到 20KHz，在声音信号频谱分析一般不需要对每个频率成分进行具体分析。为了方便起见，人们把 20Hz 到 20KHz 的声频范围分为几个段落，每个频带成为一个频段。频段的划分采用恒定带宽比，即保持频带的上、下限之比为一常数。实验证明，当声音的声压级不变而频率提高一倍时，听起来音调也提高一倍。若使每一频带的上限频率比下限频率高一倍，即频率之比为 2，这样划分的每一个频段称 1 倍频段，简称倍频段。如果在一个倍频段的上、下限频率之间再插入两个频率，使 4 个频率之间的比值相同（相邻两频率比值 = 1.26 倍）。这样将一个倍频段划分为 3 个频段，称这种频段为 1/3 倍频段。

1.2 EQ 频点参数值详解

频点	特征
31Hz、39Hz	以次声和极低音为主，人耳很难听到，可忽略这两个频点。增益值拉到哪儿影响都不明显。有的教程说 39Hz 这个频点附近区域以沉闷音为特点。
50Hz、62Hz	这两个频点左右如果增益值大于+15dB，对低音的空间感有很弱的影响，主要是轻微的谐波共振感，正增益则产生，负增长率益则缺失。个人听觉的体会是，在大幅度正增益时人声稍微显得有点低沉。
80Hz、100 Hz、125 Hz、200 Hz	这四个频点附近是男低音的主要基音区，与低音的混厚感有很大关系。一般说来，此频点附近丰满，音色则显得较为厚实，反之，音色则会显得苍白无力。但是，如果过强，音色往往会出现低频共振声，听着会有“隆隆”的“同音轰鸣感”，有人将它形容为“极强重感”，尤其是 200Hz 附近，特别明显，甚至有击鼓样的共振音，总之给人的感觉不舒服，如果不是 Rock 风格的人声，建议不要大幅度正增益。

250Hz、350 Hz、420 Hz、500 Hz	这四个频点与中低音的力度很有关系，是中低音声的主音区，因此在调节增益时应特别小心，如果过度正增益，会出现让人心烦的“嗡嗡”声，让人觉得闷，在 250 Hz 很明显，正增益过了就好像感冒鼻音重一般，如果的确鼻音重的人声则可以考虑在此频点附近进行负增益。
640Hz、800Hz、1KHz	这三个频点是中的主音区，与声音的开阔度有一定关系，同时与听觉上的混浊度也相关，正增益是明晰而狭窄，即我们常说的“喉音感”，应该说，适当喉音能增长率加声音的感性，但如果过了，则会让人觉得不自然，就男女声来讲，这几个频点附近的增益应格外小心，在调音学说里将 500-800Hz 频率称为"危险频率"，意思是要谨慎使用；负增益则是混浊开阔，但太过则会显得人声松散无力。800 Hz 和 1KHz 的提升是电话音模拟效果的主要手段。
1.3KHz、1.6KHz、2KHz	这三个频点是中高音的主音区，男声则略表现为偏高音。与声音（在尤其是中音）的硬度相关。负增益则相对舒缓，正增益则相对紧凑。个人体会是 1.3KHz 附近与中音部的声音的明亮感还相关，适当提升能增加声音明亮度，但往往不宜超过 3.5dB（此数值纯粹为个人的体会，仅供参考），不然就很不自然，会有生硬感。
2.5 KHz 、3.2 KHz、4KHz	这三个频点与人声的锐利度相关，也就是表现为声音的穿透力，而且与其它频段相较而言，与混音时的声场远近相关度最大。声学研究表明，人耳腔的谐振频率主要集中在 1KHz 到 4KHz 之间，因而人耳对这个频率相对也就非常敏感的。因此，不论是男声还是女声，在 EQ 上对 2~4kHz 之间这几个频点附近都不宜进行衰减。因为它的缺失和弱化会使声场较远，混音时就缺乏立体感，同时语音也显得较为模糊。当然也不可过强，不然容易产生“咳声”样的附谐音。就我个人的体会而言，4 KHz 是一个特别关键的频点，对声音的锋利度影响最为明显，如果提升太过，会出现明显的齿音，往往是女声出现齿音的基频点。如果是做摇滚乐混音时，倒不妨试试适当提升增加人声的锋利度。
5KHz、6.4 KHz、8KHz	这三个频点是高音的主音区，男声略表现为极高音，频点附近与声音的清脆度相关，主要影响高音的清晰度、明亮度，适当的域值能让音色听起来清脆悦耳，增益不足则人声的音色平淡，增益过多则人声变得尖锐，甚至出现啸叫音及齿音，听觉上刺耳。
10 KHz、13 KHz、16 KHz 、 20 KHz	频率在 8 KHz 以上，人声就很纤细了，不主张进行提升，因为过分提升会让人感到声音发尖、发毛，如果这部分的高音的确缺乏，可考虑用 BBE 的高音激励来实现，那样出来的是谐波，不致破坏整体听觉感受。20KHz 以上就是超声了，人耳是听不到的，因此，一般不作调整。

调整增益参数及频段时可以参考上述 EQ 频点参数详解。

2 二阶 IIR 带通滤波器的设计

2.1 2 阶 IIR 滤波器的传递函数

$$H(Z) = \frac{b_0 + b_1 * z^{-1} + b_2 * z^{-2}}{a_0 + a_1 * z^{-1} + a_2 * z^{-2}} \quad (2.1)$$

已知：增益（dB） 中心频率（ f_0 ） 采样频率（ f_s ） 品质因子（ $Q = f_s / \text{带宽}$ ）得出：

$$A = \sqrt{10^{(db/20)}} \quad (2.2)$$

$$w = 2\pi \frac{f_0}{f_s} \quad (2.3)$$

$$\alpha = \frac{s}{2 * Q} \quad (2.4)$$

二阶 IIR 带通滤波器系数的计算：

$$b_0 = 1 + A * \alpha \quad (2.5)$$

$$b_1 = -2 \cos(w) \quad (2.6)$$

$$b_2 = 1 - A * \alpha \quad (2.7)$$

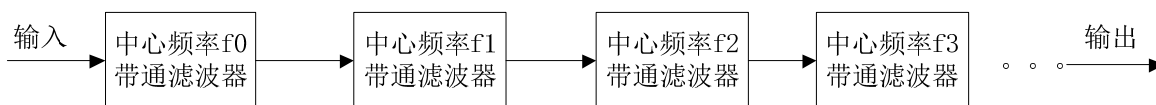
$$a_0 = 1 + \frac{\alpha}{A} \quad (2.8)$$

$$a_1 = -2 \cos(w) \quad (2.9)$$

$$a_2 = 1 - \frac{\alpha}{A} \quad (2.10)$$

2.2 均衡器的实现原理

均衡滤波器目的并非滤掉选定的频段，而是改变选定频段的增益，其工作原理如下所示：



滤波过程如下：

$$y = (b_0 * x + b_1 * x' + b_2 * x'' - a_1 * y' - a_2 * y'') / a_0 \quad (2.11)$$

其中，y'为上一滤波器的输出，y''为上上次滤波器的输出；x'为上次的输入 x''为上上次的输入。

3 RKNanoD SONY Walkman EQ 设计方案

3.1 EQ 关键函数说明

➤ EQ_ClearBuff()

函数原型	void EQ_ClearBuff()
输入参数	无
返回值	无
函数功能说明	清除滤波 buffer

➤ EffectAdjust()

函数原型	void EffectAdjust()
输入参数	无
返回值	无
函数功能说明	EQ 系数设置

➤ EQ_SetPreScale()

函数原型	void EQ_SetPreScale(short ps)
输入参数	0、1 or 2
返回值	无
函数功能说明	设置滤波前预衰减的 dB 数，0:不衰减，1:衰减 9db，2:误差 12db

➤ ROCKEQ_Get_Coeff()

函数原型	void ROCKEQ_Get_Coeff(short *pGain, long Fs)
输入参数	所设置每个 band 的 db 值，采样率
返回值	无
函数功能说明	从表中查找得到各个滤波器的滤波系数

➤ RKEQProcess()

函数原型	void RockEQProcess(short *pData, long PcmLen)
输入参数	输入数据 buffer，buffer 长度
返回值	正确返回 1，错误返回 0

函数功能说明	EQ 音效调整
--------	---------

➤ RockEQFiltering()

函数原型	void RockEQFiltering(short *pwBuffer, long cwBuffer, long LR, long mode)
输入参数	输入数据，输入长度，声道，PCM 存放模式
返回值	无
函数功能说明	滤波函数，共进行 5 个串联滤波

➤ RockBassFiltering()

函数原型	void RockBassFiltering(short *pwBuffer, long cwBuffer, long LR, long mode)
输入参数	输入数据，输入长度，声道，PCM 存放模式
返回值	无
函数功能说明	Bass 滤波函数，共进行 2 个串联滤波

3.2 五段 EQ 设计方案

RKNanoD Sony Walkman EQ 采用五段均衡器，其理论中心频率及其品质因子如下表所示：

表 3.1 RKNanoD SONY Walkman 五段 EQ 及 BASS 模式方案

		EQ 滤波器组				
五段 EQ	中心频率 f_0	100	315	1k	3.15k	10k
	品质因子 Q	1.2	1.2	1.2	1.2	1.2
Bass	中心频率 f_0	30	60	120		
	品质因子 Q	0	0.8	1.2		

假设相邻的量中心频率比值为 $scale$ ，则带宽 B 和品质因子 Q 的计算公式如下：

$$B = \frac{scale - 1}{\sqrt{scale}} \times f_0$$

$$Q = \frac{\sqrt{scale}}{scale - 1}$$

➤ 5 组二阶 IIR 滤波器：

差分方程为：

$$y = \frac{b_0}{a_0}x + \frac{b_1}{a_0}x' + \frac{b_2}{a_0}x'' - \frac{a_1}{a_0}y' - \frac{a_2}{a_0}y'' \quad (3.1)$$

为了减少一次滤波器乘法，令：

$$gain = \frac{b_0}{a_0} \quad (3.2)$$

得出：

$$y = gain * (x + \frac{b_1}{b_0} x' + \frac{b_2}{b_0} x'' - \frac{a_1}{b_0} y' - \frac{a_2}{b_0} y'') \quad (3.3)$$

需要注意的是：由于 y' 与 y'' 分别为上次与上上次的滤波器输出，因此在求取滤波器系数时，不需要除以 $gain$ ，最终得到的差分方程为：

$$y = (x + a1 * y' + a2 * y'' + b1 * x' + b2 * x'') * gain \quad (3.4)$$

其中： $a1 = -\frac{a_1}{b_0}$; $a2 = -\frac{a_2}{b_0}$; $b1 = \frac{b_1}{a_0}$; $b2 = \frac{b_2}{a_0}$; $gain = \frac{b_0}{a_0}$

➤ 实现流程图为：

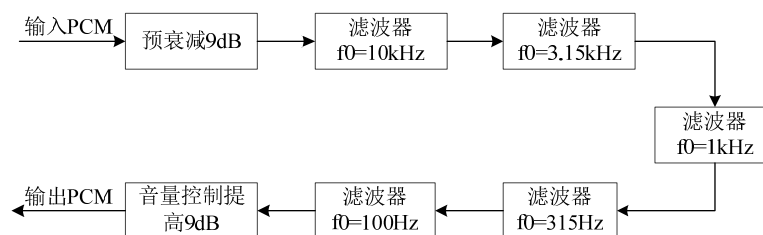


图 3.1 EQ 实现流程图

RockEQFiltering()函数为滤波函数，共进行 5 个串联滤波。如下图所示：

```

_ATTR_RKEQ_TEXT_
void RockEQFiltering(short *pwBuffer, long cwBuffer, long LR, long mode)
{
    short *pwbuffer = pwBuffer;
    long cwbuffer = cwBuffer;
#ifdef SUPPORT_HIGH_PRECISION
    long lAccumulate;
    long lbuff_1, lbuff_2;
    long y, y1, x, x1;
    //long lhigh32, llow32;
    __int64 sum;
#else
    long y;
#endif

    if (cwbuffer < 1) return;

    if (g_FilterState.mode[LR] == EQ_NOR)
    {
        return;
    }

    RockEQReduce9dB(pwBuffer, cwBuffer, LR, mode); // 1预衰减9dB

    //-----
    //Filter 5 [10K]
    //-----
    //SubFiltering(pwBuffer, cwBuffer, LR, g_FilterState.EQ_ps_factor, 8, mode);
    SubFilter_Core(cwbuffer, pwbuffer, &g_FilterState, 8, LR);
    //-----

```

```

//-----
//Filter 5 [10K]
//-----
//SubFiltering(pwBuffer, cwBuffer, LR, g_FilterState.EQ_ps_factor, 8, mode);
SubFilter_Core(cwbuffer,pwbuffer, &g_FilterState , 8 , LR );

//-----
//Filter 4 [3.15K]
//-----
//SubFiltering(pwBuffer, cwBuffer, LR, 0, 6, mode);
SubFilter_Core(cwbuffer,pwbuffer, &g_FilterState , 6 , LR );

//-----
//Filter 3 [1K]
//-----
//SubFiltering(pwBuffer, cwBuffer, LR, 0, 4, mode);
SubFilter_Core(cwbuffer,pwbuffer, &g_FilterState , 4 , LR );

//-----
//Filter 2 [315Hz]
//-----

/* A method may lose some precision, but more quick.
*/
SubFiltering(pwBuffer, cwBuffer, LR, 0, 2, mode);

//-----
//Filter 1 [100Hz]
//-----

```

2.进行5个串联滤波

```

//-----
//Filter 1 [100Hz]
//-----

SubFiltering2(pwBuffer, cwBuffer, LR, mode);

```

3 最后一个滤波器乘以总体增益

需要注意的是：为了节省运算量，我们只在最后一级滤波器的输入中乘以总体增益 factB，即在 SubFiltering2()函数中，进行如下图所示操作：

```

#if SUPPORT_HIGH_PRECISION
_ATTR_RKEQ_TEXT
void SubFiltering2(short *pwBuffer, long cwBuffer, long LR, long mode)
{

```

```

    while (cwbuffer--)
    {
        x = *pwbuffer * factB;
        x1 = x;
        y = x + lbuff_1;
        y1 = y;
        y >>= 13;

        if ( (y>>15) != (y>>31) )
        {
            y >>= 31;
            y ^= 0x7fff;
        }

        *pwbuffer = y; // write the PCM data
        pwbuffer += mode;
    }
}

```

最后一个滤波器输入乘以总体增益

3.3 BASS 模式设计方案

BASS 模式理论中心频率 f_0 及其品质因子 Q 如表 3.1 所示。Sony BASS 模式对低频段 30Hz、60 Hz、120 Hz 进行增益处理。

➤ BASS 二阶 IIR 滤波器

差分方程为：

$$y = -\frac{a_1}{a_0} y' - \frac{a_2}{a_0} y'' + \frac{b_0}{a_0} x + \frac{b_1}{a_0} x' + \frac{b_2}{a_0} x'' \quad (3.5)$$

因此：得到滤波器系数为（从左至右）： $[-\frac{a_1}{a_0}, -\frac{a_2}{a_0}, \frac{b_0}{a_0}, \frac{b_1}{a_0}, \frac{b_2}{a_0}]$

➤ 实现流程图



图 3.2 BASS 模式实现流程图

RockBassFiltering()函数为 BASS 滤波函数，共进行 2 个串联滤波。

BASS 模式的预衰减在 RKEQProcess()函数内实现：

```

RockEQReduce9dB(pData + j[0] * 2, PcmLen - j[0], 0, 2);
RockBassFiltering(pData + j[0] * 2, PcmLen - j[0], 0, 2); // left channel;

RockEQReduce9dB(pData + 1 + j[1] * 2, PcmLen - j[1], 1, 2);
RockBassFiltering(pData + 1 + j[1] * 2, PcmLen - j[1], 1, 2); // right channel
  
```

在 RockBassFiltering()函数内实现两个串联滤波：

```

void RockBassFiltering(short *pwBuffer, long cwBuffer, long LR, long mode)
{
    AudioInOut_Type *pAudio = &AudioIOBuf;
    RKEffect *pEffect = &pAudio->EffectCtl;

    //-----
    //Filter 2 [120]
    //-----
    // SubFiltering_bass(pwBuffer, cwBuffer, LR, 0, 2, mode);
    //SubFilter_Core(cwbuffer,pwbuffer, &g_bass_FilterState , 2 , LR );
    filter_2_int_bass(pwBuffer, cwBuffer, 1, LR);

    //-----
    //Filter 1 [60]
    //-----
    // SubFiltering_bass(pwBuffer, cwBuffer, LR, 0, 0, mode);
    filter_2_int_bass(pwBuffer, cwBuffer, 0, LR);
}
  
```


4 如何修改 EQ 系数表

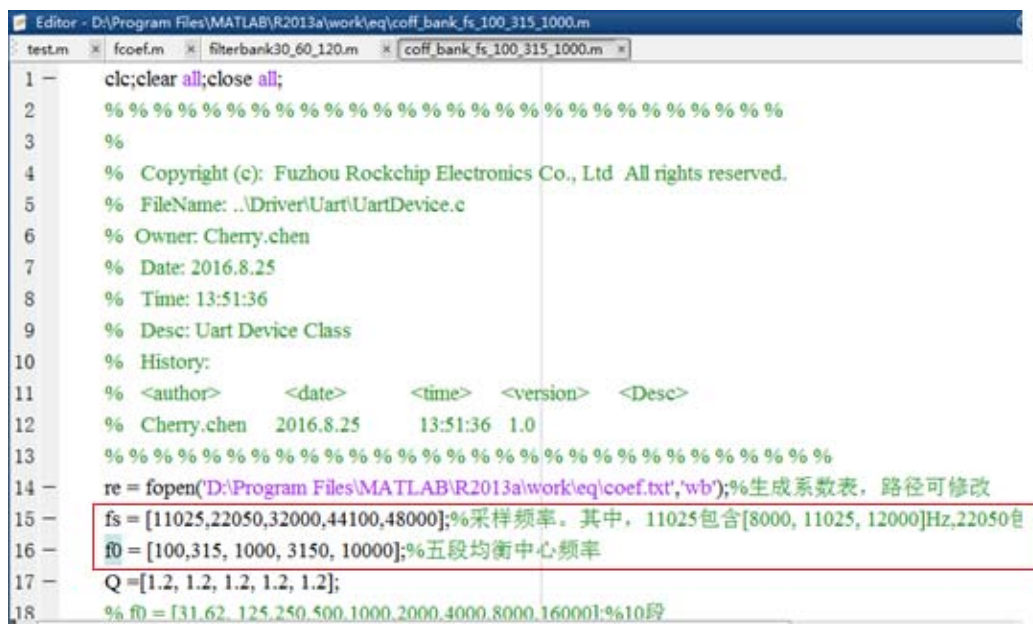
设置 EQ 系数表主要分为三个步骤：（1）EQ 系数表的生成；（2）EQ 系数定点化；（3）添加 EQ 系数表至 RKNanoD 工程；

4.1 EQ 系数表

EQ 在音效处理中起着重要的作用，对于目前市场上大多数 MP3 设备来说，频率范围通常内置为五个频段，每段可进行十级频率响应的调节。下面以五频段，每段可进行 25 级频率响应的调节为例，设置调整 EQ 参数。

➤ 第一步：生成 EQ 系数表

在 matlab 文件 `coef_bank_fs_100_315_1000.m` 中，修改中心频率数组 `f0` (`f0` 数组长度为频段数)，及对应的品质因子 `Q`，生成 EQ 系数表存于 `coef.txt` 文件中（存储路径可修改）。如下图所示：



```
1  clc;clear all;close all;
2  %%%%%%%%%%%%%%%
3  %
4  % Copyright (c): Fuzhou Rockchip Electronics Co., Ltd All rights reserved.
5  % FileName: ..\Driver\Uart\UartDevice.c
6  % Owner: Cherry.chen
7  % Date: 2016.8.25
8  % Time: 13:51:36
9  % Desc: Uart Device Class
10 % History:
11 % <author>      <date>      <time>      <version>      <Desc>
12 % Cherry.chen   2016.8.25    13:51:36    1.0
13 %%%%%%%%%%%%%%%
14 re = fopen('D:\Program Files\MATLAB\R2013a\work\eq\coef.txt','wb');%生成系数表，路径可修改
15 fs = [11025,22050,32000,44100,48000];%采样频率。其中，11025包含[8000, 11025, 12000]Hz,22050电
16 f0 = [100,315, 1000, 3150, 10000];%五段均衡中心频率
17 Q = [1.2, 1.2, 1.2, 1.2, 1.2];
18 % f0 = [31.62, 125.250, 500, 1000, 2000, 4000, 8000, 16000];%10段
```

式（2.5）~（2.10）中带通滤波器系数的计算如下图：

✧ effect.c 中修改宏 EQ_NUM:

```
#define EQ_NUM 5
```

✧ 在 rk_eq.c 中更新 EQ 系数表 Eq_Table[]

```
eq_table_t Eq_Table =
{
    //_ATTR_RKEQ_DATA_
    //short eqArray[]=
    {
        //采样 0
        //采样 0 频段 0
        0x3d01, 0xfffffe2e5, 0xfffffc0db, 0x1f3f, 0x1eea,
        0x3d29, 0xfffffe2be, 0xfffffc0e7, 0x1f34, 0x1f04,
        0x3d4f, 0xfffffe298, 0xfffffc0f3, 0x1f28, 0x1f1d,
        0x3d72, 0xfffffe274, 0xfffffc0ff, 0x1f1b, 0x1f35,
        0x3d94, 0xfffffe252, 0xfffffc10c, 0x1f0e, 0x1f4d,
        0x3db4, 0xfffffe232, 0xfffffc11b, 0x1f00, 0x1f64,
        0x3dd2, 0xfffffe214, 0xfffffc129, 0x1ef1, 0x1f7b,
        0x3def, 0xfffffe1f7, 0xfffffc139, 0x1ee1, 0x1f92,
        0x3e0a, 0xfffffe1dc, 0xfffffc14a, 0x1ed0, 0x1fa8
```

✧ 音效增益设置

在 effect.c 中，修改音效增益，目前我们支持 HEAVY, POP, JAZZ, UNIQUE 四种音效模式。

```
_ATTR_AUDIO_DATA_ short PresetGain[4][EQ_NUM] =
{
    {15, 12, 9, 15, 12}, // HEAVY
    {12, 15, 15, 9, 12}, // POP
    {18, 12, 12, 9, 15}, // JAZZ
    {18, 6, 16, 13, 18} // UNIQUE
```

4.2 BASS 系数

BASS 系数计算与 EQ 参数的计算方法基本一致，需要注意的是：因为精度原因，系数放大 2^{20} 倍。

➤ 第一步：生成 BASS 系数表

由于中心频率为 30Hz 的品质因子为 0，因此只需生成 60Hz 及 120Hz 的 bass 系数即可。在 matlab 文件 eq_bass_60_120.m 中，设置采样频率 fs、中心频率 f0、品质因子 Q 等参数。需要注意的是，bass 系数表的增益为 12:-2:2。生成的 bass 系数表存于 bass_coef.txt（路径可修改）。其基本参数如下所示：

将 ox_eq.txt 中的系数分别添加至 NanoD 工程中, rk_eq.c 中 Eq_Table 中(ox_eq.txt 中, 前一组为 60Hz bass 系数, 后一组为 120Hz 系数)。如下图所示。

```

37:
38: // ATTR_RKEQ_DATA
39: //int bass_60_tbl[]= // 一个采样率下60 Q1.2 12 : -2 : 2 的增益 共5个采样率 0
40: {
41: //采样 0
42: 0x1f9a01,0xffff06144,0x106928,0xffe065ff,0xf3594,
43: 0x1f8e4d,0xffff06cfa,0x105261,0xffe071b3,0xf40a5,
44: 0x1f87f0,0xffff07357,0x104770,0xffe07810,0xf4538,
45: 0x1f7a18,0xffff08132,0x103246,0xffe085e8,0xf4c88,
46: 0x1f6a9e,0xffff090ae,0x101dd7,0xffe09562,0xf517b,
47: 0x1f6235,0xffff09917,0x1013d2,0xffe09dcb,0xf5317,
48: //采样 1
49:
50: 0x1fcde1,0xffff030ef,0x1034e6,0xffe0321f,0xf9a2b,
51: 0x1fc7f3,0xffff036dc,0x102978,0xffe0380d,0xf9fab,
52: 0x1fc4b9,0xffff03a16,0x1023f9,0xffe03b47,0xfa1ef,
53: 0x1fbdb1,0xffff0411f,0x101957,0xffe0424f,0xfa58a,
54: 0x1fb5d2,0xffff048ff,0x100f0e,0xffe04a2e,0xfa7f2,
55: 0x1fb189,0xffff04d48,0x100a01,0xffe04e77,0xfa8b7,
: // ATTR_RKEQ_DATA
: //int bass_120_tbl[]= //120hz 五个采样率 120hz 2.5 -9.5
: { //采样 0
: //采样 0 频段 0
: 0x1f41fe,0xffff0ab42,0x1042a4,0xffe0be02,0xf1219,
: 0x1f3818,0xffff0b52e,0x1034fc,0xffe0c7e8,0xf15d5,
: 0x1f2296,0xffff0cabe,0x101a3f,0xffe0dd6a,0xf1b03,
: 0x1f2296,0xffff0cabe,0x101a3f,0xffe0dd6a,0xf1b03,
: 0x1f2296,0xffff0cabe,0x101a3f,0xffe0dd6a,0xf1b03,
: 0x1f0a96,0xffff0e2cb,0x100000,0xffe0f56a,0xf1d35,
: //采样 1
:
: 0x1fa4ae,0xffff05695,0x1021b1,0xffe05b52,0xf87b9,
: 0x1f9f9b,0xffff05ba8,0x101acd,0xffe06065,0xf898a,

```

修改 bass 音效对应的增益

```

_ATTR_AUDIO_DATA_short EqBassBoostGain[EQ_NUM] = {12, 12, 12, 12, 12}; // BASS BOOST

```

至此, EQ 系数添加完毕。