音频编解码器 AAC 的学习

AAC 编解码器资料 下载网址：http://www.audiocoding.com/

AAC（Advanced Audio Coding），中文称为“高级音频编码”，出现于 1997 年，基于 MPEG-2 的音频编码技术。由 Fraunhofer IIS、杜比实验室、AT&T、Sony（新力）等公司共同开发，目的是取代 MP3 格式。2000 年，MPEG-4 标准出现后，AAC 重新集成了其特性，加入了 SBR 技术和 PS 技术，为了区别于传统的 MPEG-2 AAC 又称为 MPEG-4 AAC。

【扩展名】

AAC 编码的主要扩展名有三种：

(1).AAC - 使用 MPEG-2 Audio Transport Stream(ADTS，参见 MPEG-2)容器，区别于使用 MPEG-4 容器的 MP4/M4A 格式，属于传统的 AAC 编码（FAAC 默认的封装，但 FAAC 亦可输出 MPEG-4 封装的 AAC）

(2).MP4 - 使用了 MPEG-4 Part 14(第 14 部分)的简化版即 3GPP Media Release 6 Basic (3gp6，参见 3GP) 进行封装的 AAC 编码（Nero AAC 编码器仅能输出 MPEG-4 封装的 AAC）；

(3).M4A - 为了区别纯音频 MP4 文件和包含视频的 MP4 文件而由苹果(Apple)公司使用的扩展名，Apple iTunes 对纯音频 MP4 文件采用了".M4A"命名。M4A 的本质和音频 MP4 相同，故音频 MP4 文件亦可直接更改扩展名为 M4A。

【 概览 】

作为一种高压缩比的音频压缩算法，AAC 压缩比通常为 18：1，也有数据说为 20：1，远胜 mp3；在音质方面，由于采用多声道，和使用低复杂性的描述方式，使其比几乎所有的传统编码方式在同规格的情况下更胜一筹。不过直到 2006 年，使用这一格式存储音乐的并不多，可以播放该格式的 mp3 播放器更是少之又少，目前所知仅有苹果 iPod、Sony Walkman（NWZ-A、NWZ-S、NWZ-E、NWZ-X 系列）、任天堂 NDSi 和魅族 M8(微软最新推出的 Windows 7 附带的 Windows media player12 也支持 aac)。此外计算机上很多音乐播放软件都支持 AAC（前提是安装过 AAC 解码器），如苹果 iTunes。但在移动电话领域，AAC 的支持度已很普遍，Nokia、Sony Ericsson、Motorola 等品牌均在其中高端产品中支持 AAC（一开始主要是 **LC-AAC**，随着移动电话性能的发展，**HE-AAC** 的支持也已广泛）。

【特点】

AAC 可以支持多达 48 个音轨，15 个低频（LFE）音轨，5.1 多声道支持，更高的采样率（最高可达 96kHz，音频 CD 为 44.1kHz）和更高的采样精度（支持 8bit、16bit、24bit、32bit，音频 CD 为 16bit）以及有多种语言的兼容能力，更高的解码效率，一般来说，AAC 可以在对比 MP3 文件缩小 30%的前提下提供更好的音质。

【 版本与扩充 】

(1) MPEG-2 AAC Main

(2) **MPEG-2 AAC LC** (Low Complexity) 传统的 LC-AAC 即 low complexity 版本的 AAC。

(3)**MPEG-2 AAC SSR** (Scalable Sampling Rate)

MPEG-2 的两种是是已经过时的了，多用的是 MPEG-4 的 main、LC 和 he 三种模式。

(4)MPEG-4 AAC Main

(5)MPEG-4 AAC LC (Low Complexity) ，为低复杂度版本，适合中等码流 96kbps ～～192kbps，在此码流下，LC-AAC 可以完全打败同码率的用 LAME 最高质量慢速编码模式的 MP3。

(6) MPEG-4 AAC SSR (Scalable Sample Rate)

(7)MPEG-4 AAC LTP (Long Term Predicition)

(8)MPEG-4 AAC LD (Low Delay)

(9)MPEG-4 AAC HE (High Efficiency) ，高效 AAC，HE-AAC 或 AAC-HE，"AAC+") - 结合了**谱带复制** (Spectral Band Replication，SBR) 及 AAC 技术，适用于低比特率（64kbps 以下），简写又称为 **"aacPlus v1"**。

HE 主要是为了低码率，在小于 36kbps 的情况下，能达到比其他编码器都要优秀的音质。HE 有两个版本，一个是 V1,也就是 aac+，包含 LC+SBR。

HE V1 版(aacPlus V1)：即 豪华版 AAC
HE V2 版(aacPlus V2)：即 增强的 豪华版 AAC

(10) HE-AAC v2，又称 **"aacPlus v2"**，enhanced aac plus， – 采用 HE-AAC 中的 SBR 技术，结合**参数化立体声** (Parametric Stereo，PS) 包含 **LC+SBR+PS。**

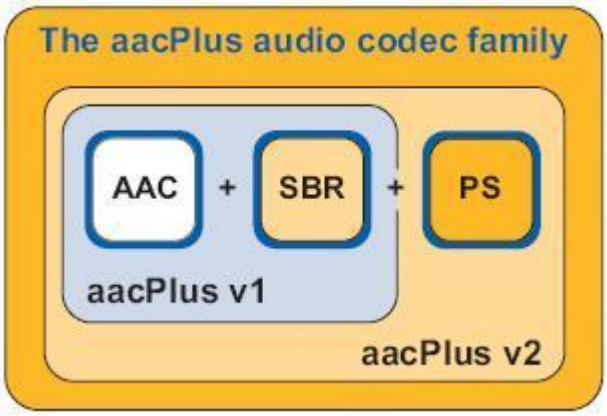
3gpp 组织的 Enhanced aac plus，包含有定点和浮点，需要优化。

4、代码优化

主要从 **基本运算，函数内联，mmx/sse 等方面实现优化**，算法方面改动很少。经过查找，在万方维普等数据库找到了几篇相关的硕士论文，西电有几个导师在带领学生搞这套代码的优化，看了他们在优化方面做的已有的工作。遗憾的是，并没有用上。

5、问题及解决

Faac/faad 的源代码中间可能存在很多问题，如播放声音，女声有尖锐的现象，最后用了 **lp 滤波(low pass??)**来实现，**截断高频**，此控件可供选择使用。



AAC V1 + (PS) ----> AAC V2

aacPlus V1 豪华版 AAC: **AAC 技术+ SBR(谱带复制技术)**

aacPlus V2 也叫增强的豪华版 **AAC**: **AAC 技术 + SBR(谱带复制技术) + PS(参数化立体声)**

【AAC 编码库---- faac 编码库调用接口实习 】:

1、AAC Encode

iso/mpeg 2 AAC 或者 mpeg4 AAC 的编码库提供一个高层面的 接口来编码 mpeg2 a 和 mpeg4 的 iso aac 文件，对于 c/c++程序，通过使用 faac.h 中的函数接口来完成

对于 AAC 的编码，其工程下的 include 包含头文件有 faac.h 及 faaccfg.h 两个头文件。

AAC 编码库的调用顺序：对于编码 AAC 码流，以下调用顺序是必须的。

(1) 每需要一个 AAC encoder 编码器实例，则调用 faacEncOpen() 创建 AAC encoder 编码器。

(2) 设置 AAC encoder 编码器的相关选项：先调用调用 faacEncGetCurrentConfiguration()获取 AAC 编码器的相关选项，调用 faacEncSetConfiguration() 来设置 AAC Encoder 编码的相关选项。

(3) 不断调用 faacEncEncode() 函数对 音频数据进行编码，直到音频数据编码完，faacEncEncode()函数会返回存储于用户所分配的缓冲区的 AAC 编码数据码流。

(4) 用 0 样本输入来初始化 faacEncEncode()过程，然后只需调用 faacEncEncode()，直到所有的音频数据样本都被编码完。

(5) 如果 faacEncEncode() 已经编码完，并返回 0 字节的输出数据，则可以释放 AAC encoder 编码器实例。

AAC 编码库 libfaac 的相关函数接口：

相关函数：

1、 初始化 AAC 编码器及释放 AAC 编码器相关的函数。

(i) faacEncOpen 函数接口:创建，并初始化一个AAC编码器

```
faacEncHandle FAACAPI faacEncOpen
(
   unsigned long sampleRate, /*采样率*/
   unsigned int  numChannels/*通道个数*/,
   unsigned long *inputSamples//由AAC编码器内部返回来的值，返回每次调用 AAC编码器时，要送给编码器编码的样本个数，也即一次需送多少个样本给 AAC编码器
   unsigned long *maxOutputBytes  /*最多输出多少字节*/
);
```

(ii) faacEncClose() 接口，释放 AAC 编解码器

```
void FAACAPI faacEncClose
 (
   faacEncHandle hEncoder
 );
```

(ii) faacEncGetCurrentConfiguration()，AAC 编码器相关参数选项的获取

```
faacEncConfigurationPtr   FAACAPI
faacEncGetCurrentConfiguration
(
  faacEncHandle hEncoder
);
```

(iv) faacEncSetConfiguration() 设置 AAC encoder 编码器的参数相关信息。

```
int FAACAPI faacEncSetConfiguration(faacEncHandle hEncoder,
 faacEncConfigurationPtr config)
```

(v) AAC 编码器核心函数 ：faacEncEncode()

/*

如返回 <0 ，则表明 AAC 编码失败

如返回 0，并不代表失败。

如返回 >0，则返回的是 outputBuffer 中的编码样本字节数

```
*/
int FAACAPI faacEncEncode
(
    faacEncHandle hEncoder,   // AAC编码器的指针
    short *inputBuffer,     //输入语音样本数组
    unsigned int samplesInput, //输入语音样本个数，这个样本数应该同 faacEncOpen()函数在创建 AAC 编码器时所返回来的 inputSamples值一样。
    unsigned char *outputBuffer, /*存储AAC编码之后的数据流，该缓冲区的大小至少必须为 由faacEncOpen()调用后所得到的 maxOutputBytes 值大小。*/
    unsigned int bufferSize
);
```

AAC 编码器配置选项的数据结构：

```
typedef struct faacEncConfiguration
{
    unsigned int mpegVersion;  //mpeg的版本： mpeg2,mpeg4
    unsigned int aacObjectType;//aac对像类型， Main,Low,或 LTP
    unsigned int allowMidside;//设1，表示允许 使用 mid/side编码，0不用 mid/side编码。
    unsigned int useLfe; //设1，使用一个 LFE通道，目前不支持本标志。
    unsigned int useTns; //1表示使用TNS，0 不使用 TNS
    unsigned long bitRate; //每秒钟，每个通道的 bitrate
    unsigned int bandWidth; //最大的 bandwith，单位为 Hz

    /*
      以上为 iso的 faacEncConfiguration数据结构类型， 对于 faac 还有以下的参数：
    */

      /* Quantizer quality */
        unsigned long quantqual;

      /* Bitstream output format (0 = Raw; 1 = ADTS) */
        unsigned int outputFormat;

        /* psychoacoustic model list */
        psymodellist_t *psymodellist;

        /* selected index in psymodellist */
```

```
    unsigned int psymodelidx;

    /*
       PCM Sample Input Format
       0  FAAC_INPUT_NULL      invalid, signifies a
misconfigured config
       1  FAAC_INPUT_16BIT     native endian 16bit
       2  FAAC_INPUT_24BIT     native endian 24bit in 24
bits     (not implemented)
       3  FAAC_INPUT_32BIT     native endian 24bit in 32
bits     (DEFAULT)
       4  FAAC_INPUT_FLOAT     32bit floating point
    */
    unsigned int inputFormat;

    /* block type enforcing
(SHORTCTL_NORMAL/SHORTCTL_NOSHORT/SHORTCTL_NOLONG) */
    int shortctl;

    /*
       Channel Remapping
       Default  0, 1, 2, 3 ... 63  (64 is MAX_CHANNELS in
coder.h)

       WAVE 4.0    2, 0, 1, 3
       WAVE 5.0    2, 0, 1, 3, 4
       WAVE 5.1    2, 0, 1, 4, 5, 3
       AIFF 5.1    2, 0, 3, 1, 4, 5
    */
    int channel_map[64];

} faacEncConfiguration, *faacEncConfigurationPtr;
```

【AAC 解码库faad相关】：

Faad库的 include目录下的接口头文件：

faad.h

neaacdec.h (ne-aac-dec.h) ，ne 表示 nero,aac,decode表示，由 nero 开发的 aac decode 编码器。

AAC解码器下的相关数据结构及函数接口，都在 faad库的 include目录下的 neaacdec.h 中头文件中。

/* AAC解码器类型，即为一个通用的结构指针*/

```
(1)typedef void *NeAACDecHandle;

(2)typedef struct mp4AudioSpecificConfig
{
   /* Audio Specific Info */
   unsigned char objectTypeIndex;
   unsigned char samplingFrequencyIndex;
   unsigned long samplingFrequency;
   unsigned char channelsConfiguration;

   /* GA Specific Info */
   unsigned char frameLengthFlag;
   unsigned char dependsOnCoreCoder;
   unsigned short coreCoderDelay;
   unsigned char extensionFlag;
   unsigned char aacSectionDataResilienceFlag;
   unsigned char aacScalefactorDataResilienceFlag;
   unsigned char aacSpectralDataResilienceFlag;
   unsigned char epConfig;

   char sbr_present_flag;
   char forceUpSampling;
   char downSampledSBR;
} mp4AudioSpecificConfig;

(3)typedef struct NeAACDecConfiguration
{
   unsigned char defObjectType;
   unsigned long defSampleRate;
   unsigned char outputFormat;
   unsigned char downMatrix;
   unsigned char useOldADTSFormat;
   unsigned char dontUpSampleImplicitSBR;
} NeAACDecConfiguration, *NeAACDecConfigurationPtr;

(4)typedef struct NeAACDecFrameInfo
{
   unsigned long bytesconsumed;  //
   unsigned long samples;
   unsigned char channels;
   unsigned char error;
   unsigned long samplerate;   //音频采样率
```

```
        /* SBR: 0: off, 1: on; upsample, 2: on; downsampled,
   3: off; upsampled */
        unsigned char sbr;

        /* MPEG-4 ObjectType */
        unsigned char object_type;

        /* AAC header type; MP4 will be signalled as RAW also
   */
        unsigned char header_type;

        /* multichannel configuration */
        unsigned char num_front_channels;
        unsigned char num_side_channels;
        unsigned char num_back_channels;
        unsigned char num_lfe_channels;
        unsigned char channel_position[64];

        /* PS: 0: off, 1: on */
        unsigned char ps;
      } NeAACDecFrameInfo;
```

解码的相关API函数说明：

(i)NeAACDecOpen 创建 AAC解码器

```
NeAACDecHandle NEAACAPI NeAACDecOpen(void);
```

(ii) NeAACDecClose 释放 AAC解码器

```
void NEAACAPI NeAACDecClose(NeAACDecHandle hDecoder);
```

(iii) NeAACDecGetCurrentConfiguration 获取 AAC解码器相关参数信息

NeAACDecConfigurationPtr NEAACAPI NeAACDecGetCurrentConfiguration(NeAACDecHandle hDecoder);

(iv) NeAACDecSetConfiguration 设置 AAC解码器相关参数信息

unsigned char NEAACAPI NeAACDecSetConfiguration(NeAACDecHandle hDecoder, NeAACDecConfigurationPtr config);

(v)NeAACDecInit 用从 AAC文件所读取的信息初始化 AAC解码库

long NEAACAPI NeAACDecInit(NeAACDecHandle hDecoder, unsigned char *buffer, unsigned long buffer_size, unsigned long *samplerate, unsigned char *channels);

返回值： <0 ，表示失败，>=0 表示读到的字节数。

(VI)NeAACDecInit2　aacdecoder 初始化函数

char　NEAACAPI　NeAACDecInit2(NeAACDecHandle hDecoder, unsigned char *pBuffer, unsigned long SizeOfDecoderSpecificInfo, unsigned long*samplerate, unsigned char *channels);

基于MP4文件中所找到的AudioSpecificConfig 参数来初始化 AAC解码库。

返回值：　<0 ，表示失败，>=0 表示读到的字节数。

(VII) NeAACDecDecode　AAC解码器。

void*　NEAACAPI　NeAACDecDecode(NeAACDecHandle　hDecoder,　//解码器对像指针

NeAACDecFrameInfo　*hInfo,　//语音解码帧信息
unsigned char *buffer,　//解码出来的缓冲区的地址
unsigned long buffer_size); //缓冲区的大小

Decodes　the　AAC　data　passed　in　buffer.