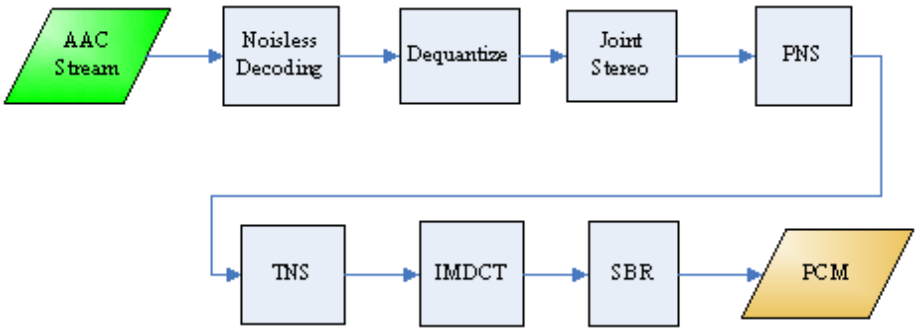
AAC 解码算法原理详解

本文详细介绍了符合 ISO/IEC 13818-7(MPEG2 AAC audio codec)，ISO/IEC 14496-3(MPEG4 Audio Codec AAC Low Complexity)进行压缩的的 AAC 音频的解码算法。

1、程序系统结构

下面是 AAC 解码流程图：



AAC 解码流程图

在主控模块开始运行后，主控模块将 AAC 比特流的一部分放入输入缓冲区，通过查找同步字得到一帧的起始，找到后，根据 ISO/IEC 13818-7 所述的语法开始进行 Noisless Decoding(无噪解码)，无噪解码实际上就是哈夫曼解码，通过反量化(Dequantize)、联合立体声（Joint Stereo），知觉噪声替换（PNS），瞬时噪声整形（TNS），反离散余弦变换（IMDCT），频段复制（SBR）这几个模块之后，得出左右声道的 PCM 码流，再由主控模块将其放入输出缓冲区输出到声音播放设备。

2. 主控模块

主控模块的主要任务是操作输入输出缓冲区，调用其它各模块协同工作。其中，输入输出缓冲区均由 DSP 控制模块提供接口。输出缓冲区中将存放的数据为解码出来的 PCM 数据，代表了声音的振幅。它由一块固定长度的缓冲区构成，通过调用 DSP 控制模块的接口函数，得到头指针，在完成输出缓冲区的填充后，调用中断处理输出至 I2S 接口所连接的音频 ADC 芯片（立体声音频 DAC 和 DirectDrive

耳机放大器）输出模拟声音。

3. 同步及元素解码

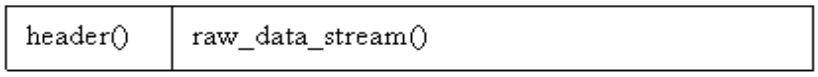
同步及元素解码模块主要用于找出格式信息，并进行头信息解码，以及对元素信息进行解码。这些解码的结果用于后续的无噪解码和尺度因子解码模块。

AAC 的音频文件格式有以下两种：

ADIF：Audio Data Interchange Format 音频数据交换格式。这种格式的特征是可以确定的找到这个音频数据的开始，不需进行在音频数据流中间开始的解码，即它的解码必须在明确定义的开始处进行。故这种格式常用在磁盘文件中。

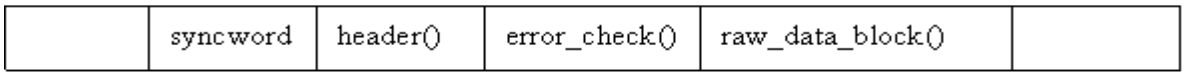
ADTS：Audio Data Transport Stream 音频数据传输流。这种格式的特征是它是一个有同步字的比特流，解码可以在这个流中任何位置开始。它的特征类似于 mp3 数据流格式。

AAC 的 ADIF 格式见下图：



3.1 ADIF 的组织结构

AAC 的 ADTS 的一般格式见下图：



3.2 ADTS 的组织结构

图中表示出了 ADTS 一帧的简明结构，其两边的空白矩形表示一帧前后的数据。

ADIF 和 ADTS 的 header 是不同的。它们分别如下所示：

Syntax	No. of bits	Mnemonic
adif_header()		
{		
adif_id	**32**	**bslbf**
copyright_id_present	**1**	**bslbf**
if(copyright_id_present)		
copyright_id	**72**	**bslbf**
original_copy	**1**	**bslbf**
home	**1**	**bslbf**
bitstream_type	**1**	**bslbf**
bitrate	**23**	**uimsbf**
num_program_config_elements	**4**	**bslbf**
for (i = 0; i < num_program_config_elements + 1; i++) {		
if(bitstream_type == '0')		
adif_buffer_fullness	**20**	**uimsbf**
program_config_element()		
}		
}		

3.3 ADIF 的头信息

Syntax	No. of bits	Mnemonic
adts_fixed_header()		
{		
Syncword	**12**	**bslbf**
ID	**1**	**bslbf**
Layer	**2**	**uimsbf**
protection_absent	**1**	**bslbf**
Profile	**2**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
private_bit	**1**	**bslbf**
channel_configuration	**3**	**uimsbf**
original/copy	**1**	**bslbf**
Home	**1**	**bslbf**
Emphasis	**2**	**bslbf**
}		

3.4 ADTS 的固定头信息

Syntax	No. of bits	Mnemonic
adts_variable_header()		
{		
copyright_identification_bit	**1**	**bslbf**
copyright_identification_start	**1**	**bslbf**
aac_frame_length	**13**	**bslbf**
adts_buffer_fullness	**11**	**bslbf**
no_raw_data_blocks_in_frame	**2**	**uimsbf**
}		

ADTS 的可变头信息

3.5 帧同步

帧同步目的在于找出帧头在比特流中的位置，13818-7 规定，aac ADTS 格式的帧头为 12 比特的“1111 1111 1111”.

3.6 头信息解码

ADTS 的头信息为两部分组成，其一为固定头信息，紧接着是可变头信息。固定头信息中的数据每一帧都相同，而可变头信息则在帧与帧之间可变。

3.7 元素信息解码

在 AAC 中，原始数据块的组成可能有六种不同的元素。它们分别是

SCE: Single Channel Element 单通道元素。单通道元素基本上只由一个 ICS 组成。一个原始数据块最可能由 16 个 SCE 组成。

CPE: Channel Pair Element 双通道元素，由两个可能共享边信息的 ICS 和一些联合立体声编码信息组成。一个原始数据块最多可能由 16 个 SCE 组成。

CCE: Coupling Channel Element 藕合通道元素。代表一个块的多通道联合立体声信息或者多语种程序的对话信息。

LFE: Low Frequency Element 低频元素。包含了一个加强低采样频率的通道。

DSE: Data Stream Element 数据流元素，包含了一些并不属于音频的附加信息。

PCE: Program Config Element 程序配置元素。包含了声道的配置信息。它可能出现在 ADIF 头部信息中。

FIL: Fill Element 填充元素。包含了一些扩展信息。如 SBR，动态范围控制信息等。

3.8 处理流程

(1). 判断文件格式，确定为 ADIF 或 ADTS

(2). 若为 ADIF，解 ADIF 头信息，跳至第 6 步。

(3). 若为 ADTS，寻找同步头。

(4). 解 ADTS 帧头信息。

(5). 若有错误检测，进行错误检测。

(6). 解块信息。

(7). 解元素信息。

4.无噪声解码

无噪编码就是哈夫曼编码，它的作用在于进一步减少尺度因子和量化后频谱的冗余，即将尺度因子和量化后的频谱信息进行哈夫曼编码。

全局增益编码成一个 8 位的无符号整数，第一个尺度因子与全局增益值进行差分编码后再使用尺度因子编码表进行哈夫曼编码。后续的各尺度因子都与前一个尺度因子进行差分编码。

量化频谱的无噪编码有两个频谱系数的划分。其一为 4 元组和 2 元组的划分，另一个为节划分。对前一个划分来说，确定了一次哈夫曼表查找出的数值是 4 个还是 2 个。对后一个划分来说，确定了应该用哪一个哈夫曼表，一节中含有若干的尺度因子带并且每节只用一个哈夫曼表。

4.1 分段

无噪声编码将输入的 1024 个量化频谱系数分为几个段（section），段内的各点均使用同一个哈夫曼表，考虑到编码效率，每一段的边界最好同尺度因子带的边界重合。所以每一段必段传送信息应该有：段长度，所在的尺度因子带，使用的哈夫曼表。

4.2 分组和交替

分组是指忽略频谱系数所在窗，将连续的，具有相同尺度因子带的频谱系数分为一组放在一起，共享一个尺度因子从而得到更好的编码效率。这样做必然会引起交替，即本来是以

c[组][窗][尺度因子带][系数索引]

为顺序的系数排列，变为将尺度因子带同的系数放在一起：

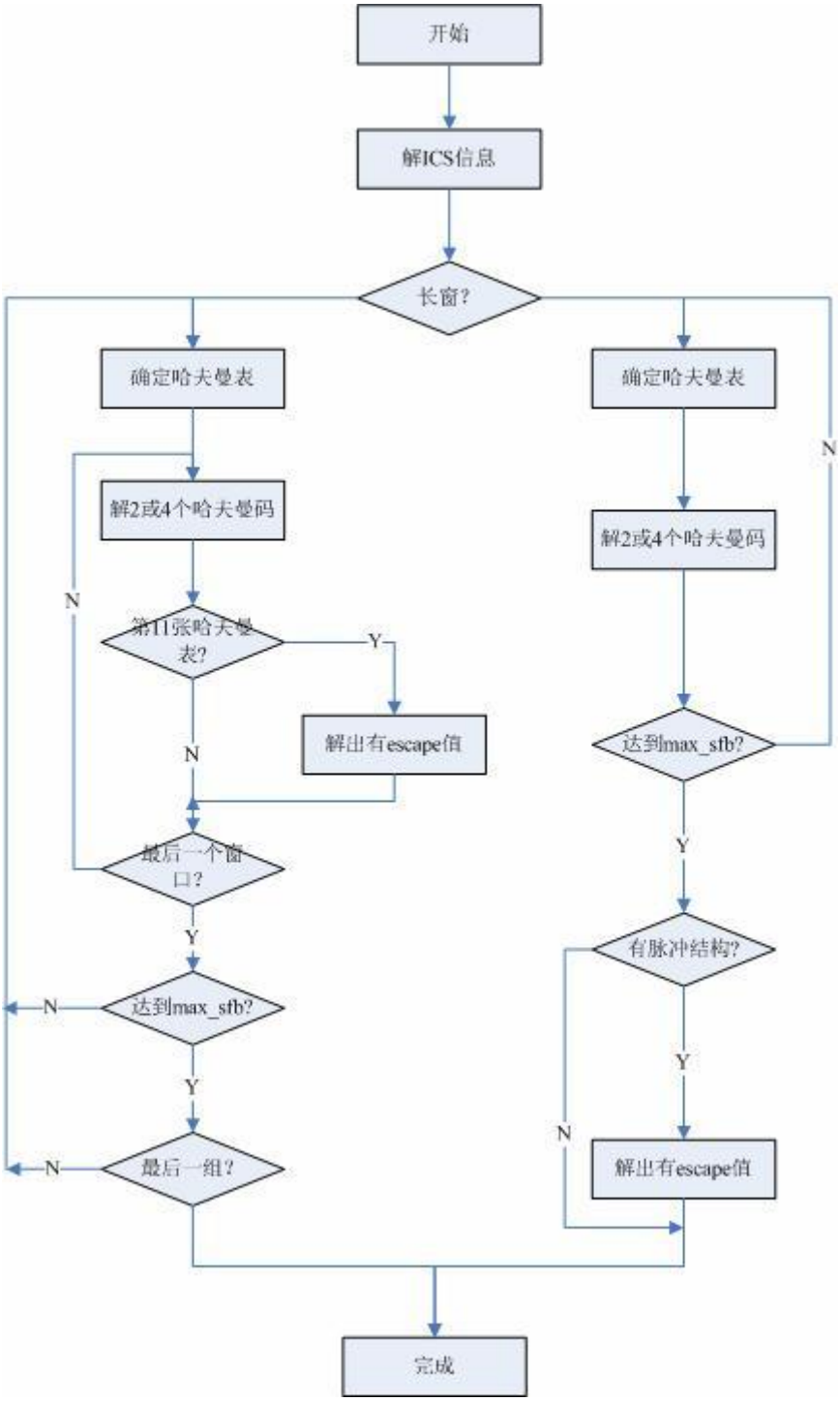
c[组][尺度因子带][窗][系数索引]

这样就引起了相同窗的系数的交替。

4.3 大量化值的处理

大量化值在 AAC 中有两种处理方法：在哈夫曼编码表中使用 escape 标志或使用脉冲 escape 方法。前者跟 mp3 编码方法相似，在许多大量化值出现时采用专门的哈夫曼表，这个表暗示了它的使用将会在哈夫曼编码后面跟跟一对 escape 值及对值的符号。在用脉冲 escape 方法时，大数值被减去一个差值变为小数值，然后使用哈夫曼表编码，后面会跟一个脉冲结构来帮助差值的还原。

无噪解码的流程图如下：



无噪声解码流程图

5.尺度因子解码及逆量化

在 aac 编码中，逆量化频谱系数是由一个非均匀量化器来实现的，在解码中需进行其逆运算。即保持符号并进行 4/3 次幂运算。

在频域调整量化噪声的基本方法就是用尺度因子来进行噪声整形。尺度因子就是

一个用来改变在一个尺度因子带的所有的频谱系数的振幅增益值。使用尺度因子这种机制是为了使用非均匀量化器在频域中改变量化噪声的比特分配。

5.1 尺度因子带（scalefactor-band）

频率线根据人耳的听觉特性被分成多个组，每个组对应若干个尺度因子，这些组就叫做尺度因子带。为了减少信息含有短窗的边信息，连续的短窗可能会被分为一组，即将若干个短窗当成一个窗口一起传送，然后尺度因子将会作用到所有分组后的窗口去。

5.2 反量化公式：

x_invquant = sign(x_quant) * | x_quant| ^(4/3)

其中

x_invquant 表示反量化的结果

sign (x)表示取 x 的符号

^ 表示幂运算

5.3 应用尺度因子公式：

x_rescal = x_invquant * gain

gain = 2 ^ (0.25 * (sf - SF_OFFSET))

其中

x_rescal 为应用了尺度因子公式之后的值

gain 为一个增益

sf 为尺度因子值

SF_OFFSET 为一个常数，设为 100

6、联合立体声解码

联合立体声有两种，M/S stereo(中间旁道立体声)和 intensity stereo(强度立体声)

6.1 M/S stereo

在 M_S 立体声模式中，传送的是规格化的中间/旁边声道的信息，计算公式如下：

$$\begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} m \\ s \end{bmatrix}$$

其中，

l，r 表示转换后的左右声道值

m 表示中间声道值

s 表示旁边声道值

6.2 Intensity stereo

在强度立体声模式中，左声道传的是幅值，右声道的 scalefactor 传的是立体声的位置 is_pos。如果仅在一个指定了 common_window 为 1 的 CPE 中的右通道中指定哈夫曼表为 INTENSITY_HCB 或 INTENSITY_HCB2，则解码时使用强度立体声模式。其计算公式如下：

is_pos += dpcm_is_pos

scale = invert_intensity * 0.5 ^(0.25 * ispos)

r_spec = scale * l_spec

从完全备份中还原

从完全备份中还原数据库非常简单，在 9.3.2 节中会详细地

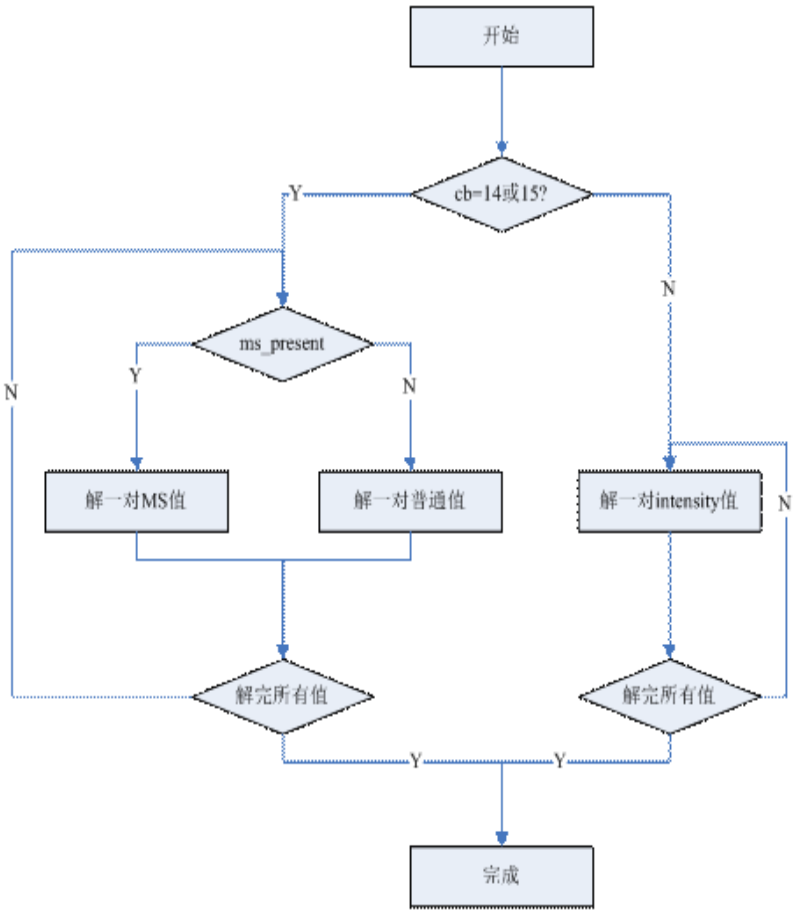
is_pos 是右声道传送的 scalefactor

dpcm_is_pos 是上一个 is_pos,初值为 0

scale 为强度因子

invert_intensity 为是否反转哈夫曼表（表 14 和表 15）这个变量由 ms_used 指定，关系为：invert_intensity = 1 - 2 * ms_used，另外，当 ms_mask_present 为 0 时，invert_intensity 恒为 1。

6.3 处理流程



联合立体声解码流程图

7、PNS

PNS(Perceptual Noise Substitution) 知觉噪声替换模块是一种以参数编码的方式模拟噪声的模块。在判别出音频值中的噪声后，将些噪声不进行量化编码，而是采用一些参数告诉解码器端这是某种噪声，然后解码器端将会对这些噪声用一些随机的编码来制造出这一类型的噪声。

在具体操作上，PNS 模块对每个尺度因子带侦测频率 4kHz 以下的信号成分。如果这个信号既不是音调，在时间上也无强烈的能量变动，就被认为是噪声信号。其信号的音调及能量变化都在心理声学模型中算出。

在解码中，如果发现使用了哈夫曼表 13(NOISE_HCB)，则表明使用了 PNS。

由于 M/S 立体声解码与 PNS 解码互斥，故可以用参数 ms_used 来表明是否两个声道都用同样的 PNS。如果 ms_used 参数为 1，则两个声道会用同样的随机向量来生成噪声信号。

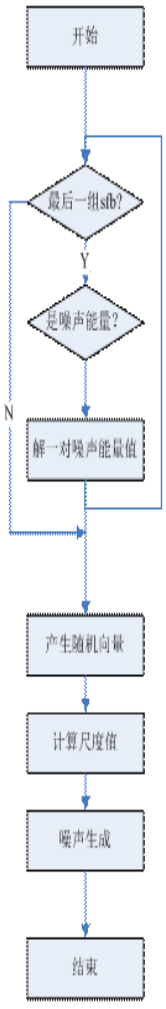
PNS 的能量信号用 noise_nrg 来表示，如果使用了 PNS，则能量信号将会代替各

自的尺度因子来传送。

噪声能量编码同尺度因子一样，采用差分编码的方式。第一个值同样为全局增益值。它同强度立体声位置值及尺度因子交替地放在一起，但对差分解码来说又彼此忽略。即下一个噪声能量值以上一个噪声能量值而不是强度立体声位置或尺度因子为标准差分解码。

随机能量将会在一个尺度因子带内产生 noise_nrg 所计算出的平均能量分布。

7.1 处理流程



PNS 解码流程图

8、TNS

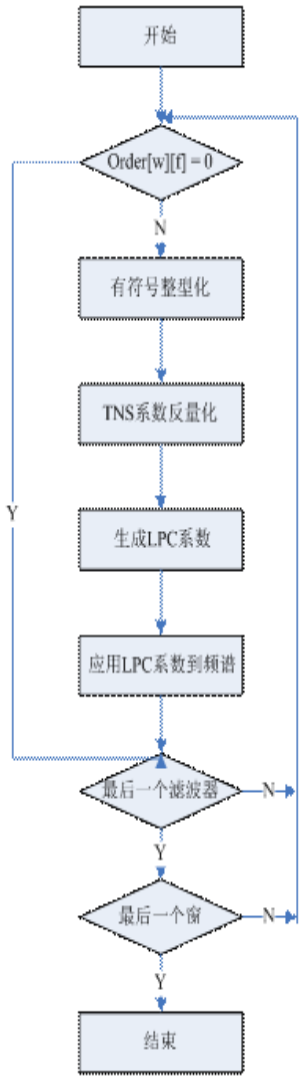
TNS 瞬态噪声整形用于控制一个转换窗口内的瞬时噪声形态。它是用一个对单个通道的滤波过程来实现的。

传统的变换编码方案常常遇到信号在时域变化非常剧烈的问题，特别是语音信号，这个问题是因为量化后的噪声分布虽然在频率域上得到控制，但在时域上却以一个常数分布在一个转换块内。如果这种块中信号变化得很剧烈却又不转向一个短块去，那这个常数分布的噪声将会被听到。

TNS 的原理利用了时域和频域的二元性和 LPC(线性预测编码)的时频对称性，即在其中的任意一个域上做编码与在另一域上做预测编码等效，也就是说，在一个域内做预测编码可以在另一域内增加其解析度。量化噪声产生是在频域产生的，降低了时域的解析度，故在这里是在频域上做预测编码。

在 AACplus 中，由于基于 AAC profile LC，故 TNS 的滤波器阶数被限制在 12 阶以内。

8.1 处理流程



TNS 解码流程图

9. IMDCT

将音频数据从频域转换到时域的过程主要是由将频域数据填入一组 IMDCT 滤波器来实现的。在进行 IMDCT 变换后，输出数值经过加窗，叠加，最后得到时域数值。

9.1 IMDCT 公式

$$x_{i,n} = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} spec[i][k]\cos\left(\frac{2\pi}{N}(n+n_0)(k+\frac{1}{2})\right)$$

for 0<= n <N

其中

n 为采样点索引值

i 为窗索引值

k 为频谱系数索引值

N 为窗函数的长度，全为短窗 N = 256,其余情况为 2048

n0=(N/2 +1)/2

9.2 块型

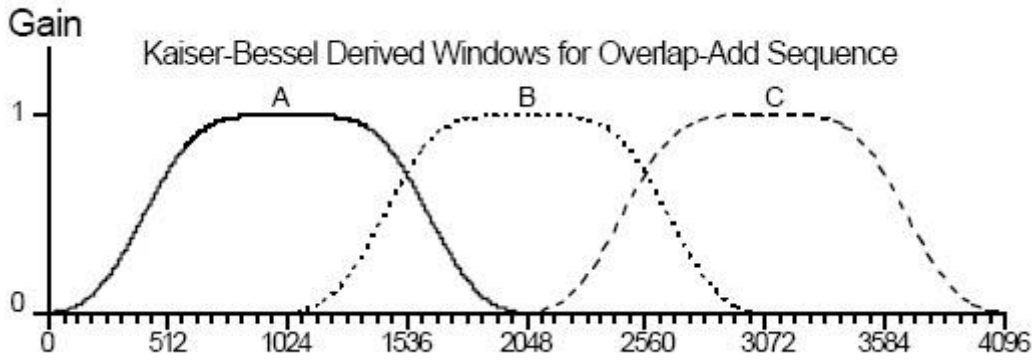
由于长块的频域分辨率较高而短块的时域分辨率较高，故长块较适合相对平稳的时域信号，而短块较适合变化相对较快的时域信号。

长块长度为 2048 个点，短块长度为 256 个点。

9.3 加窗

AAC 用到两种窗函数，分别为 Kaiser-Bessel 类(KBD)窗和正弦窗。

KBD 窗如下所示：



其定义为：

$$W'(n,\alpha)=\frac{I_0\left[\pi\alpha\sqrt{1.0-\left(\frac{n-N/4}{N/4}\right)^2}\right]}{I_0[\pi\alpha]}$$

for 0 <= n <= N/2

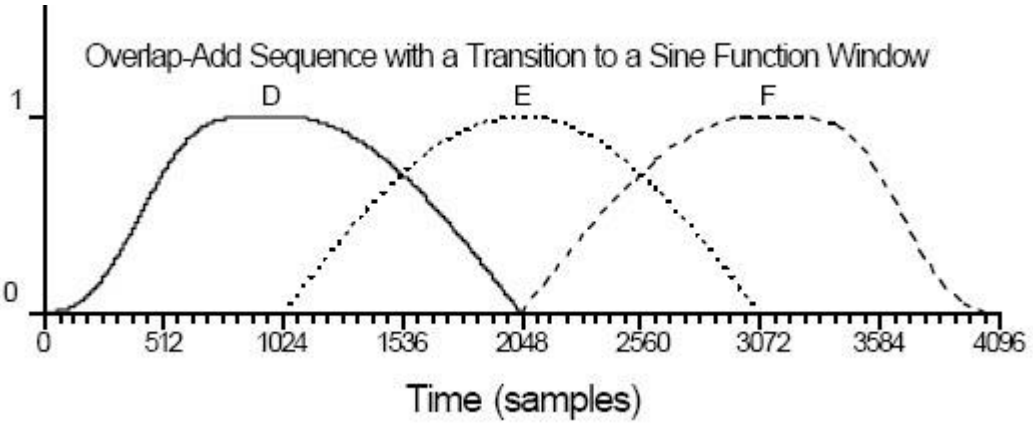
其中

$$I_0[x]=\sum_{k=0}^{\infty}\left[\frac{(\frac{x}{2})^k}{k!}\right]^2$$

$$\alpha=\begin{cases}4 \text{ for N} = 2048\ (1920)\\ 6 \text{ for N} = 256\ (240)\end{cases}$$

使用 KBD 窗时，window_shape 为 1

正弦窗如下所示：



其定义为

$$W_{SIN\_LEFT,N}(n) = \sin(\frac{\pi}{N}(n+\frac{1}{2})) \quad \text{for} \quad 0 \le n < \frac{N}{2}$$

$$W_{SIN\_RIGHT,N}(n) = \sin(\frac{\pi}{N}(n+\frac{1}{2})) \quad \text{for} \quad \frac{N}{2} \le n < N$$

使用正弦窗时，window_shape 为 0

另定义：

$$W_{LEFT,N}(n) = \begin{cases} W_{KBD\_LEFT,N}(n), \text{ if } window\_shape\_previous\_block == 1 \\ W_{SIN\_LEFT,N}(n), \text{ if } window\_shape\_previous\_block == 0 \end{cases}$$

对应于四种不同的窗序列分别进行不同的加窗变换：

1.）仅有长块：

window_shape 为 1：

$$W(n) = \begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0 \le n < N\_l/2 \\ W_{KBD\_RIGHT,N\_l}(n), & \text{for } N\_1/2 \le n < N\_l \end{cases}$$

window_shape 为 0：

$$w(n)=\begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0\le n < N\_l/2 \\ W_{SIN\_RIGHT,N\_l}(n), & \text{for } N\_l/2\le n < N\_l \end{cases}$$

加窗后，时域信号可用 w(n)表示为：

$$z_{i,n}=w(n)\cdot x_{i,n};$$

2.） 长开始块：

window_shape 为 1：

$$w(n)=\begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0\le n < N\_l/2 \\ 1.0, & \text{for } N\_l/2\le n < \frac{3N\_l-N\_s}{4} \\ W_{KBD\_RIGHT,N\_s}(n+\frac{N\_s}{2}-\frac{3N\_l-N\_s}{4}), & \text{for } \frac{3N\_l-N\_s}{4}\le n < \frac{3N\_l+N\_s}{4} \\ 0.0, & \text{for } \frac{3N\_l+N\_s}{4}\le n < N\_l \end{cases}$$

window_shape 为 0：

$$w(n)=\begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0\le n < N\_l/2 \\ 1.0, & \text{for } N\_l/2\le n < \frac{3N\_l-N\_s}{4} \\ W_{SIN\_RIGHT,N\_s}(n+\frac{N\_s}{2}-\frac{3N\_l-N\_s}{4}), & \text{for } \frac{3N\_l-N\_s}{4}\le n < \frac{3N\_l+N\_s}{4} \\ 0.0, & \text{for } \frac{3N\_l+N\_s}{4}\le n < N\_l \end{cases}$$

加窗后，时域信号可用 w(n)表示为：

$$z_{i,n}=w(n)\cdot x_{i,n};$$

3.） 只有短块：

window_shape 为 1：

$$W_0(n)=\begin{cases}W_{LEFT,N\_s}(n), & \text{for } 0\le n<N\_s/2\\ W_{KBD\_RIGHT,N\_s}(n), & \text{for } N\_s/2\le n<N\_s\end{cases}$$

$$W_{1-(M-1)}(n)=\begin{cases}W_{LEFT,N\_s}(n), & \text{for } 0\le n<N\_s/2\\ W_{KBD\_RIGHT,N\_s}(n), & \text{for } N\_s/2\le n<N\_s\end{cases}$$

window_shape 为 0：

$$W_0(n)=\begin{cases}W_{LEFT,N\_s}(n), & \text{for } 0\le n<N\_s/2\\ W_{SIN\_RIGHT,N\_s}(n), & \text{for } N\_s/2\le n<N\_s\end{cases}$$

$$W_{1-(M-1)}(n)=\begin{cases}W_{SIN\_LEFT,N\_s}(n), & \text{for } 0\le n<N\_s/2\\ W_{SIN\_RIGHT,N\_s}(n), & \text{for } N\_s/2\le n<N\_s\end{cases}$$

加窗后，时域信号可用 w(n)表示为：

$$z_{i,n}=\begin{cases}0, & \text{for } 0\le n<\frac{N\_l-N\_s}{4}\\ x_{0,n-\frac{N\_l-N\_s}{4}}\cdot W_0(n-\frac{N\_l-N\_s}{4}), & \text{for } \frac{N\_l-N\_s}{4}\le n<\frac{N\_l+N\_s}{4}\\ x_{j-1,n-\frac{N\_l+(2j-3)N\_s}{4}}\cdot W_{j-1}(n-\frac{N\_l+(2j-3)N\_s}{4})+x_{j,n-\frac{N\_l+(2j-1)N\_s}{4}}\cdot W_j(n-\frac{N\_l+(2j-1)N\_s}{4}), & \\ \qquad \text{for } 1\le j<M,\ \frac{N\_l+(2j-1)N\_s}{4}\le n<\frac{N\_l+(2j+1)N\_s}{4} & \\ x_{M-1,n-\frac{N\_l+(2M-3)N\_s}{4}}\cdot W_{M-1}(n-\frac{N\_l+(2M-3)N\_s}{4}), & \\ \qquad \text{for } \frac{N\_l+(2M-1)N\_s}{4}\le n<\frac{N\_l+(2M+1)N\_s}{4} & \\ 0, & \text{for } \frac{N\_l+(2M+1)N\_s}{4}\le n<N\_l\end{cases}$$

2.）长结束块：

window_shape 为 1：

$$w(n)=\begin{cases}0.0, & \text{for } 0\le n<\frac{N\_l-N\_s}{4}\\ W_{LEFT,N\_s}(n-\frac{N\_l-N\_s}{4}), & \text{for } \frac{N\_l-N\_s}{4}\le n<\frac{N\_l+N\_s}{4}\\ 1.0, & \text{for } \frac{N\_l+N\_s}{4}\le n<N\_l/2\\ W_{KBD\_RIGHT,N\_l}(n), & \text{for } N\_l/2\le n<N\_l\end{cases}$$

window_shape 为 0：

$$w(n)=\begin{cases}0.0, & \text{for } 0\le n<\frac{N\_l-N\_s}{4}\\ W_{LEFT,N\_s}(n-\frac{N\_l-N\_s}{4}), & \text{for } \frac{N\_l-N\_s}{4}\le n<\frac{N\_l+N\_s}{4}\\ 1.0, & \text{for } \frac{N\_l+N\_s}{4}\le n<N\_l/2\\ W_{SIN\_RIGHT,N\_l}(n), & \text{for } N\_l/2\le n<N\_l\end{cases}$$

加窗后，时域信号可用 w(n)表示为：

$$out_{i,n}=z_{i,n}+z_{i-1,n+\frac{N}{2}};\qquad \text{for } 0\le n<\frac{N}{2},\quad N=2048\ (1920)$$

9.4 覆盖叠加

在加窗完成后得到的时域信号值 z 经过前后窗相互叠加计算，得出最后的 PCM 值：

$$out_{i,n}=z_{i,n}+z_{i-1,n+\frac{N}{2}};\qquad \text{for } 0\le n<\frac{N}{2},\quad N=2048\ (1920)$$

术语说明

AAC: Advanced Audio Coding 高级音频编码

AAC LC: AAC with Low Complexity AAC 的低复杂度配置

AAC plus: 也叫 HE-AAC, AAC+,MPEG4 AAC LC 加入 SBR 模块后形成的一个 aac 版本

MPEG：Motion Picture Expert Group

IMDCT：反离散余弦变换

ADIF：Audio Data Interchange Format 音频数据交换格式

ADTS：Audio Data Transport Stream 音频数据传输流

SCE: Single Channel Element 单通道元素

CPE: Channel Pair Element 双通道元素

CCE: Coupling Channel Element 藕合通道元素

DSE: Data Stream Element 数据流元素

PCE: Program Config Element 程序配置元素

FIL: Fill Element 填充元素

ICS: Individual Channel Stream 独立通道流

PNS: Perceptual Noise Substitution 知觉噪声替换

SBR: Spectral Band Replication 频段复制

TNS: Temporal Noise Shaping 瞬时噪声整形

ch：channel 通道