CD 11172-3

运动图像及其音频的编码

——具有 1.5Mbit/s 数据传输率的数字存储媒体

第 3 部分：音频

目 录

声明

由于译者水平有限，加上时间匆忙，本文档中难免有错误和不足之处，欢迎感兴趣的朋友们批评和指正。请把您的意见和建议发送到kongsuozt@126.com，我也会及时更新大家的意见和建议，争取把这篇文档做得更好。

本文档仅供从事相关行业的人员作学习和交流之用，不得用于出版、发行或者其他商业目的。

前言

这份国际标准草案是由众所周知的 MPEG（Moving Pictures Expert Group，运动图像专家组），也就是 SC29/WG11 准备的。MPEG 成立于 1988 年，用于发布存储于数字存储媒体上的运动图像和相关音频的编码标准。

这份标准由四个部分组成。第一部分是系统部分，阐述了系统编码层的详细信息。它结合了音频和视频数据信息定义了一个复合结构，以及实时重放同步（数据）序列所需要的再现时序信息的方法。第二部分是视频部分，阐述了视频数据的编码表示方法和重建解码图像时的解码处理过程。第三部分是音频部分，阐述了音频数据的编码表示方法和解码音频信号的要求。这份标准的第一部分中提供所有附录并且不包含标准化的需求。

这份标准的第二部分中，附录 2-A、附录 2-B、附录 2-C 包含标准化的需要条件，并且是这个标准的主要部分。提供附录 2-D 和附录 2-E 并且不包含标准化的需求。

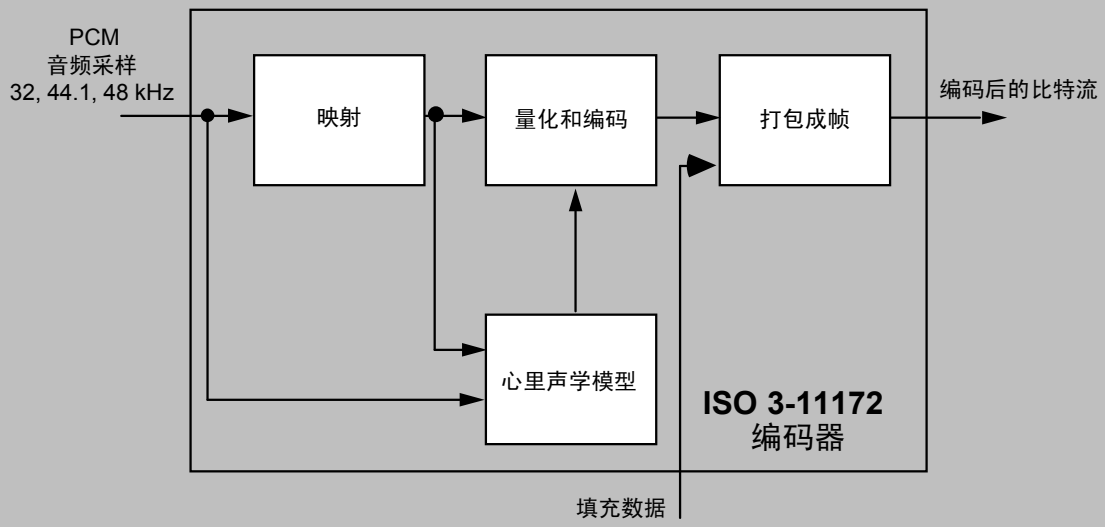
这个标准的第三部分中，附录 3-A 和附录 3-B 包含标准化的需要条件，并且是这个标准的主要部分。提供所有其他的附录并且不包含标准化的需求。

简介

为了便于理解存储压缩比特流和解码相关的详细规范，下面简要描述一下编码、存储、解码的顺序。

- **编码**

编码器处理数字音频信号并且输出便于存储的压缩比特流。编码的算法并没有标准，可以采用很多种方法来编码，比如估计听觉遮蔽阈值、量化、缩放。但是编码器的输出必须符合2.4节中指定的标准以便音频数据适于实际应用。编码器的基本框图见序图 1。



序图 1　编码器基本框图

输入的音频采样送入编码器。映射处理完成建立滤波器和代表输入音频流的子带采样。映射后的采样可以称为子带采样（例如层 I 或 II，见下文）或者变换子带采样（例如层 III，见下文）。心理声学模型产生量化和编码的一组控制数据。这些数据根据实际的编码器工具而有所不同。一种可能是用一组估算的遮蔽阈值来控制量化器。量化器和编码模块根据映射过的输入信号产生一组编码符号。此外，这个模块可以跟随编码系统而变化。帧打包模块将其他模块输出的数据组成实际的比特流，并且添加需要的其他信息（如差错校验）。

可能有四种不同的模式：单声道、双声道（两个独立音频信号编码到一个比特流中）、立体声（立体声对的左右声道信号编码到一个比特流中）和联合立体声（用不相干的立体感觉将立体声对的左右声道信号编码到一个比特流中并利用冗余）。

- **层数**

在具体应用中，可以使用编码系统中不同的层数以带来编码器复杂性和性能的增加。一个 ISO/MPEG 音频层数为 N 的解码器能够解码以层数为 N 层和 N 层以下编码的所有比特流。

Layer I：

这一层包含数字音频输入到 32 个子带的基本映射，以固定分割的方式将数据组成块，用心理声学模型决定适应的比特分配、使用块压缩和标准化来量化。Layer I 理论上最小的编码/解码延时大约是 19ms。

Layer II：

这一层提供比特分配、比例因子和采样的附加编码。使用不同的格式。Layer II 理论上最小的编码/解码延时大约是 35ms。

Layer III：

这一层在混合滤波器组的基础上引入了频率精度的提高。它添加了不同的量化器，自适应分段和对量化值的熵编码。Layer III 理论上最小的编码/解码延时大约是 19ms。

联合立体声可以作为附加特征增加到任何层中。

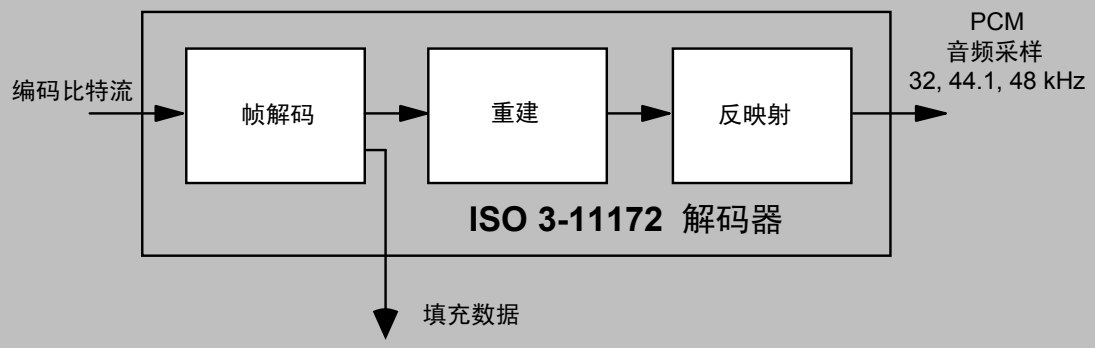
- **存储**

多种编码后的图像、声音、同步数据、系统数据、辅助数据流可以一起存放在的存储媒介上。当编辑指针强制和地址指针相一致时，编辑声音数据将变得简单。

对存储器的访问可以包含通过通信系统的远程访问。（数据的）访问应该由相应的控制单元来完成，而不是解码器自身。这个控制单元能够接收用户命令，读取和解释数据库结构信息，从媒介中读取存储信息，分解出非音频信息，并且把存储的音频数据流以需要的比特率传输到解码器中。

- **解码**

解码器以2.4.1节中定义的语法接收压缩过的音频比特流，根据2.4.2节中的规定解码数据元素，并且按照2.4.3节中的规定利用这些信息产生数字音频输出。解码流程简图见序图 2



序图 2　解码器基本框图

比特流数据送入解码器。比特流解压器和解码模块根据编码器中是否增加了差错校验而进程错误检查（参见2.4.2.4）。比特流数据被解压以恢复出许多信息片段。重建模块重建映射采样装置的量化形式。反映射就是将这些映射过的采样转换回成统一的PCM形式。

1. 基本的标准化要素

1.1 目标

这份国际标准指定了针对存储媒体的高质量音频编码表示法和解码出高质量音频信号的方法。编码器的输入和解码器的输出是可以和现有的 PCM 标准（比如压缩磁盘（CD）和数字音频磁带）相媲美的。

这份国际标准打算应用于数字存储媒体中，以提供总计高达 1.5MBit/s 连续传输速率的音频信号和视频信号，比如 CD，DAT，磁盘。存储媒体既可以直接和解码器相连，也可以通过其他方法，例如通信线路和这份标准第一部分（ISO11172）定义的复合比特流与解码器相连。这份国际标准设计的采样频率为 32KHz、44.1KHz 和 48KHz。

1.2 参考文献

下列国际标准包含从本文引用的规定，构成国际标准的条款。在出版时，这些版本标识是有效的。所有标准都服从修订，并且在这份国际标准的基础上，推荐研究以下最近标准版本的应用可能性，以统一各方意见。ISO 和 IEC 的成员维护当前有效国际标准的记录。

Recommendations and reports of the CCIR, 1990
XVIIth Plenary Assembly, Dusseldorf, 1990
Volume XI - Part 1
Broadcasting Service (Television)
Rec. 601-1 "Encoding parameters of digital television for studios".

CCIR Volume X and XI Part 3
Recommendation 648: Recording of audio signals.

CCIR Volume X and XI Part 3
Report 955-2: Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of Advanced Digital System II.

IEEE Draft Standard "Specification for the implementation of 8x 8 inverse discrete cosine transform".P1180/D2, July 18,1990

IEC publication 908:198, "CD Digital Audio System".

2. 技术标准化要素

2.1 定义

出于国际标准的目的，使用下列定义。对于局部细节，用附加说明来提示。

AC 系数[视频]：在一维或二维情况下频率不为 0 的任何 DCT 系数。

访问单元[系统]：在音频压缩的情况下存取单元指的是一个音频访问单元，在视频压缩情况下存取单元是指一副图像的编码表示。

自适应分段[音频]：在可变时间段中音频信号的数字表象的细分。

自适应位分配[音频]：码元到子带上的一次性分配和根据心理声学模型的频率变化格式。

自适应噪声分配[音频]：编码噪声到频率带上的一次性分配和根据心理声学模型的频率变化。

镜像[音频]：由子带的奈奎斯特采样造成的镜像(频率)成分。

分解滤波器组[音频]：编码器中将输入的宽带 PCM 音频信号转换成一组二次采样过的子带采样的滤波器组。

音频访问单元[音频]：音频存取单元定义为编码比特中流最小的部分，并且能够自解码。这里解码是指“完全重建声音”。

音频缓冲区[音频]：系统中存放着解码器的对象——压缩音频数据的缓冲区。

音频序列[音频]：一组不间断的正确的音频帧，并且以下参数保持不变：

--ID；

--Layer；r

--采样频率；

--对 Layer I 和 Layer II：比特率索引。

后向运动向量[视频]：显示顺序中在参考图片之后的时间里用作运动补偿的运动向量。

巴克[音频]：临界带等级的单位。巴克比例是频率比例在音频范围内非常接近人耳在（整个）频带上的频率选择性的非线性映射。

双向预测编码图像；B-图[视频]：根据前向和/或后向参考图片使用运动补偿预测编码的图像。

比特率：压缩比特流从存储介质传输到解码器输入的速率。

块压缩扩展[音频]：在特定时间段内音频信号的归一化数字表象。

块[视频]：一组 8 行×8 列的正交像素块。

边界[音频]：强度立体声编码中用到的最低子带。

字节：8 个比特的序列。

字节对齐：如果一个比特相对于编码后的比特流中第一个比特的位置是 8 的倍数，那么该比特就是字节对齐的。

声道：用来存储或者传输 ISO 11172 流的数字媒介。

临界带[音频]：频域反映人耳频率选择性的中心里声学的度量。该选择性以巴克表示。

色度(成份)[视频]：用 CCIR 601 中定义的方式表示相对于参考颜色而言两种颜色差异中的一个分量的矩阵、块或者像素采样。用来表示颜色差异信号的符号是 Cr 和 Cb。

编码音频比特流[音频]：在这个国际标准中指定的音频信号的编码表示方法。

编码视频比特流[视频]：在这个国际标准中指定的一连串的一副或多幅图像的编码表示方法。

编码顺序[视频]：图像存储和解码的顺序。这个顺序不必和显示顺序一致。

编码表示法：用以表示编码来源的数据元素。

编码参数[视频]：用户可定义的描述编码视频流特征的参数设置。编码参数刻画了比特流的特性。解码器由能够解码的比特流来描述。

成份[视频]：构成一幅图像的三个矩阵（亮度和两个色度）中的一个矩阵、块或像素数据采样。

压缩：减少一类数据中的比特位数的方法。

定比特率编码视频[视频]：常数比特率的压缩视频流。

定比特率：压缩比特流中从头到尾比特率都是常数的操作。

限定参数[视频]：在视频规范中，编码参数集的值在第二部分的 2.4.3.2 节中定义。

限定系统参数流（CSPS）[系统]：应用了第一部分 2.4.6 节的一个 ISO11172 复合流。

CRC：循环冗余校验码。

临界带等级：频域中用来代表人耳的频率选择性的心理声学的度量。

临界带[音频]：表示一个巴克宽度的频域部分。

数据元素：表示编码前或者解码后的一类数据。

DC 系数[视频]：在二维中频率为零的 DCT 系数。

DC 编码图像；D-图[视频]：仅用自身信息编码的图像。对于编码表示中的 DCT 系数，仅存在 DC 系数。

DCT 系数：指定余弦成分函数的幅度。

解码流：压缩比特流的解码重建。

解码器输入缓冲[视频]：视频缓冲校验器中指定的 FIFO 缓冲。

解码器输入速率[视频]：视频缓冲校验器中和编码视频流编码时指定的数据速率。

解码器：解码流程的具体实现。

解码流程：本国际标准中定义的读入编码比特流和输出解码图像或音频采样的流程。

解码时间戳；DTS[系统]：数据包头部可能出现的一个表示在系统解码器对象中一个访问单元被解码的时间的区域。

去加重[音频]：在存储或者传输之后应用到音频信号上的滤波以抑制由于加重导致的线性失真。

解量化[音频]：对量化编码了的子带采样进行解码以恢复原始的量化值。

解量化[视频]：在比特流中 DCT 系数的表示被解码后并在 DCT 系数被反 DCT 变换之前重新缩放量化 DCT 系数的过程。

数字存储媒体；DSM：一个数字存储或传输设备或者系统。

离散余弦变换；DCT[视频]：正向离散余弦变换或者反向离散余弦变换。DCT 是可逆的，离散正交变换。反向 DCT 在第二部分的附录 A-2 中定义。

显示顺序[视频]：解码图像应该被显示的顺序。通常这和它们在编码器输入端呈现的顺序一致。

双声道模式[音频]：两个有独立节目内容（例如双语）的音频声道被编码到一个比特流的模式。编码过程和立体声模式一样。

编辑：操作一个或多个压缩比特流以产生一个新的比特流的过程。被编辑的比特流必须符合本国际标准中的要求。

基本流[系统]：编码视频、编码音频或者其他编码比特流的通用术语。

加重[音频]：在音频信号存储或传输之前应用的滤波以提高高频信噪比。

编码器：编码过程的具体实现。

编码流程：本国际标准中没有指定的一个过程，即从输入图像或者音频采样读取比特流并输出一个本国际标准中定义的有效的编码比特流。

熵编码：信号的数字表象的变长度去噪编码以减少冗余。

快进[视频]：比实时显示更快的顺序显示图像的一个序列或者一个序列的一部分的过程。

快速傅立叶变换：实现离散傅立叶变换的快速算法（正交变换）。

滤波器组[音频]：覆盖整个音频范围的一组带通滤波器。

固定分段[音频]：将数字形式的音频信号分割成固定时间段的方法。

禁止：本文中将条目“禁止”定义为编码比特流不应该被使用。这是用来避免对启动代码的仿真。

被动更新[视频]：宏块被实时内部编码以确保编码器和解码器中的反向 DCT 处理的失配错误不能过分增大的过程。

前向运动向量[视频]：显示顺序中在参考图像之前的时间里用作运动补偿的运动向量。

帧[音频]：来自音频采集单元中的符合音频 PCM 采样的部分音频信号。

自由格式[音频]：每层中除了定义的比特率之外小于最大可变比特率的任意比特率。

未来参考图像[视频]：未来参考图像是在显示顺序中出现在当前图像之后的参考图像。

颗粒（Layer II）[音频]：32 个子带中的三个连续子带采样并且在量化前（这些采样）被看作一起的。它们代表 96 个 PCM 采样。

颗粒（Layer III）[音频]：包含各自边带信息的 576 个采样值。

图像群组[视频]：有助于随机存取的连续的一幅或多幅图像。图像群组是本国际标准中第二部分定义的编码语义中的一层。

汉宁窗[音频]：在傅立叶变换前一个采样接着一个采样的加在一个音频采样块的时域函数。

哈夫曼编码：一个指定的熵编码方法。

混合滤波器组[音频]：一系列子带滤波器组和 MDCT 的组合。

IMDCT[音频]：反向修正离散余弦变换（Inverse Modified Discrete Cosine Transform）。

联合立体声[音频]：立体声音频编程中在仅保留高频部分右声道和左声道能量包络的基础上利用立体声的细节和冗余的方法。

隔行交错扫描[视频]：传统电视图像在时间上交替图像的行以呈现不同（图像）实例的属性。

内部编码[视频]：一个块或者图像仅使用自身信息的压缩编码方式。

内部编码图像；I-图[视频]：仅使用自身信息编码的图像。

ISO 11172（复合）流[系统]：由零个或更多以本国际标准中第一部分定义的方式组合的元素流构成的比特流。

联合立体声编码[音频]：任何利用立体声的细节和冗余的方法。

联合立体声模式[音频]：用联合立体声编码的音频编码算法模式。

层[音频]：在本国际标准中定义的音频系统的编码层次中的一个层。

层[视频和系统]：在本国际标准中第一和二部分定义的视频和系统规范的数据层次中的一个层。

亮度（成份）[视频]：用来表示一个信号的单色表象和在 CCIR 601 中定义的基色的对比的矩阵、块或者像素采样。用来表示亮度的符号是 Y。

宏块[视频]：由四个 8×8 亮度数据块和两个对应的 8×8 色度数据块组成图像中亮度成份的一个 16×16 的段。宏模块有时用作参考像素数据，而有时是像素和本国际标准中第二部分定义的宏块层语义定义的数据元素的编码表示。宏块的使用在文中很清楚。

映射[音频]：通过子带滤波器和/或 MDCT 将音频信号从时域转变到频域的过程。

屏蔽阈值[音频]：时域和频域表示人类听觉系统不能感知到声音信号下界的函数。

屏蔽[音频]：人类听觉系统的特性，一个音频信号被另外的信号遮蔽而不能感知。

MDCT[音频]：修正离散余弦变换，Modified Discrete Cosine Transform。

运动补偿[视频]：用运动向量以提高像素值的预测效率。预测用运动向量来提供到过去和/或未来参考帧的偏移量，包含之前解码的用来形成预测和差分误差信号的像素。

运动估算[视频]：编码过程中估算运动向量的过程。

运动向量[视频]：一个二维向量用作运动补偿以提供从当起图像的坐标位置到参考图像的坐标位置的偏移量。

中边立体声[音频]：立体声编程中用累加和差分信号替代左右声道进行编码的一种利用立体声细节或者冗余的方法。

非内部编码[视频]：使用自身和其他时刻的块和图像的信息对块或者图像的编码。

非音成分[音频]：音频信号中类似于噪声的成分。

奈奎斯特采样：以信号最大带宽的两倍或者两倍以上进行采样。

打包[系统]：一个包由包头以及之后的一个或多个数据包组成。它是本国际标准中第一部分描述的系统编码语义中的一个层。

包数据[系统]：数据包中来自数据元素流里的连续的数据字节。

包头[系统]：用来承载包数据中包含的数据元素流信息的数据结构。

数据包[系统]：一个数据包由包头以及之后许多连续数据元素流中的字节组成。它是本国际标准中第一部分描述的系统编码语义中的一个层。

填充[音频]：为在时间上和 PCM 采样一致而调整音频帧平均长度的一种方法，通过有条件的在音频帧中添加一个插入域来实现。

过去参考图像[视频]：过去参考图像是在显示顺序中出现在比当前图像早的参考图像。

像素纵横比[视频]：显示名义上的垂直像素高度和水平宽度的比例。

像素[视频]：一个 8 位的亮度或者色度数据采样。

图像周期[视频]：图像速率的倒数。

图像速率[视频]：名义上图像应该从解码处理输出的速率。

图像[视频]：源或重建的图像数据。一幅图像由三个矩阵组成，分别代表亮度和两个色度信号，每个矩阵的元素是 8 位数字。图像层是本国际标准中第二部分定义的编码语义中的一个层。注意：术语“图像”在本国际标准中总是偏向于术语区域或帧。

多相滤波器组[音频]：有特殊相位关系的一组等带宽滤波器，能提高滤波器组的效率。

预测[视频]：用预测器提供一个当起正被解码的像素或数据的估算值。

预测编码图片；P-图[视频]：从过去参考图像用运动补偿预测编码的图像。

预测器[视频]：先前解码的像素或数据元素的线性组合。

显示时间戳；PTS[系统]：数据包头部可能存在的一个域，表示在系统目标解码器中一个显示单元的显示时间。

心理声学模型[音频]：人类声学系统屏蔽特性的数学模型。

量化矩阵[视频]：解量化器用到的一组 64 个 8 位比例因子。

量化 DCT 系数[视频]：解量化前的 DCT 系数。量化 DCT 系数的可变长度编码表示被作为压缩视频流的一部分而存储。

量化器比例因子[视频]：比特流中的一组数据元素，用在解码时缩放解量化过程。

随机存取：在一个随机位置处开始读取和解码编码过的比特流的过程。

参考图像[视频]：参考图像是显示顺序中最靠近当前图像的 I-或 P-图。

重排序缓冲区[视频]：系统目标解码器中用来存储重建的 I-图或 P-图的缓冲区。

保留：本文中的术语“保留”被定义为编码比特流用来表示该值可能在将来 ISO 的定义扩展时使用。

逆序播放[视频]：以相反的显示顺序显示图像序列的过程。

比例因子带[音频]：Layer III 中一组受同一比例因子缩放的频谱线。

比例因子索引[音频]：比例因子的数字编码。

比例因子[音频]：一组用来对量化前的数据进行缩放的因子。

顺序头[视频]：编码比特流中包含许多数据元素的编码表示的数据块。它是本国际标准中第二部分定义的编码语义中的一个层。

附属信息：比特流中控制解码器必须的信息。

省略宏块[视频]：没有存储数据的宏块。

片段[视频]：一连串的宏块。它是本国际标准中第二部分定义的编码语义中的一个层。

插入域[音频]：插入域是比特流中的基本部分，在 Layer1 中插入域表示四个字节，Lay2、3 中插入域表示一个字节。

源数据流：在压缩编码前单一非复合的采样流

扩展函数[音频]：用来描述遮蔽的频率扩展的函数。

起始码元[系统和视频]：编码比特流中植入的唯一的 32 位码元。在编码语义上它们有很多用处包括标识一些层数等信息。

STD 输入缓冲[系统]：系统目标解码器中输入端的先入先出缓冲，用来存储解码前基本流中的压缩数据。

立体声模式[音频]：模式，来自立体声对（左和右）的两个音频声道被编码到一个比特流。编码过程和双声道模式一样。

填充位；填充字节[视频]：可能插入到压缩比特流中而在解码过程中被丢弃的编码字。它们的目的是增加流的比特率。

子带[音频]：对音频频带进行的子划分。

子带滤波器组[音频]：一组覆盖整个音频范围的带通滤波器。在这个国际标准中的第三部分子带滤波器就是多相滤波器。

子带采样[音频]：音频编码器中的子带滤波器组产生的经过滤波和子采样的输入音频流的表示。滤波过的采样被称为子带采样。从 384 个时间上连续的输入音频采样可在 32 个子带内产生 12 个时间上连续的子带采样。

同步字[音频]：植入比特流中的 12 位码元，用来标识一帧的开始。

综合滤波器组[音频]：解码器中从子带采样中重建音频信号的滤波器组。

系统头[系统]：系统头是本国际标准中第一部分定义的一个数据结构，它携带着 ISO 11172 复合流的系统特征的信息总括。

系统目标解码器；STD[系统]：一个假想的解码过程的参考模型，用来描述 ISO 11172 复合流的语义。

时间戳[系统]：一个表示事件时间的术语。

音调成分[音频]：音频信号中类似于正弦曲线的成分。

变比特率：在对压缩的比特流进行解码时比特率随时间变化的操作。

变长编码；VLC：一个可逆的过程，编码时对出现频率高的事件分配较短编码字而对出现频率低的事件分配较长编码字。

视频缓冲校验器；VBV[视频]：一个假想的概念上连接到编码器输出的解码器。它的目的是对编码器或编辑处理可能产生的数据速率可变性的一个限制。

视频序列[视频]：一连串的一幅或多组图像。

Z 字形扫描顺序[视频]：DCT 系数一个特定的几乎从最低频率到最高频率的顺序排序。

2.2 符号和缩写

本国际标准中用来描述的数学运算和 C 编程语言中使用的非常相似。然而，对整数的截取和舍入是特别定义的。位操作定义为用两个补码表示的整数运算。编号和循环计数通常是从 0 开始。

2.2.1 数学运算符

＋	加法。
－	减法（双目运算符）或者负数（单目运算符）。
＋＋	自增。
－－	自减。
*	乘法。
^	乘方。
/	整数除法，丢弃余数部分。例如 7/4 和-7/-4 的结果都是 1，-7/4 和 7/-4 的结果都是-1。

//	整数除法，结果舍入到最接近的整数。小数部分四舍五入，除非有其他特殊声明。例如 3//2 的结果是 2，3//-2 的结果是-2。
DIV	整数除法，并且将结果截断到-∞（小数部分无限循环）。
%	取余运算符。只对正数有效。
Sign( )	Sign(x) = 1 x > 0 0 x = 0 -1 x < 0
NINT( )	就近取整运算符。返回和实数最接近的整数值，小数部分四舍五入。
sin	正弦函数。
cos	余弦函数。
exp	指数。
√	平方根。
$\log_{10}$	以 10 为底的对数
$\log_e$	以 e 为底的对数

2.2.2 逻辑符号

||	逻辑或
&&	逻辑与
!	逻辑非

2.2.3 关系运算符

>	大于
>=	大于等于
<	小于
<=	小于等于
==	等于
max[,…,]	参数列表中的最大值
min[,…,]	参数列表中的最小值

2.2.4 位运算符

&	位与
|	位或
>>	符号位扩展的右移
<<	左移，低位补零

2.2.5 赋值运算符

=	赋值运算符

2.2.6 缩写

定义下面的缩写来描述编码比特流中使用的不同的数据类型。

bslbf	比特串，左边的位优先（Bit string, left bit first,）。这里的“左”表示比特串在本国际标准中的书写顺序。比特串以一连串的“0”和“1”的形式书写，例如：“1000 0001”。比特串中间的空格是用来方便阅读的，没有特殊意义。
ch	声道（channel）。
nch	声道数目；单声道模式等于 1，其他模式等于 2。
gr	音频 Layer II 中由 3×32 个子带采样，音频 Layer III 中由 18×32 个子带采样构成的颗粒（granule）。
main_data	比特流中的主数据（main_data）部分包含比例因子、哈夫曼编码数据和辅助信息。
Part2_length	这个值包含着主数据中比例因子使用的比特数。
rpchof	余数多项式系数，高阶优先。
sb	子带（subband）。
scfsi	比例因子选择器信息。
uimsbf	无符号整数，最高位优先。
vlclbf	变长编码，左边位优先。这里的“左”指的是 VLC 码元书写的顺序。
window	在 block_type == 2，0 ≤ window ≤ 2 条件下实际时间空档的数量。

多字节字的字节顺序是最高位优先（大端）。

2.2.7 常量

pi	3.14159265359...
e	2.71828182846...

2.3 比特流语法的描述方法

2.4.1节中描述了解码器接收到的比特流。比特流中每个数据都用粗体表示。用它的名称、长度的比特数、类型的缩写和传输顺序来描述。

比特流中解码数据元素的动作取决于数据元素本身的值和之前一个解码的数据元素。解码过程中用到的数据元素的解码和状态变量的定义在2.4.2中描述。下面的结构用来表达数据元素出现的条件和常规类型：

注意这个语法使用了“C”代码风格，变量或表达式评估为非零值等价于条件为真。

```
while (condition) {                    // 如果条件为真，那么下面的一组数据出现
    data_element                       // 在比特流中，数据重复直到条件不为真时
    ...
}
do {
    data_element                       // 数据至少出现一次
    ...
} while (condition)                    // 数据元素重复直到条件不为真时
if (condition) {                       // 如果条件为真，那么第一组数据出现
    data_element                       // 紧接着数据流
    ...
```

```
}
else {                                          // 如果条件不为真，第二组数据出现
    data_element                                // 紧接着数据流
    ...
}
for (i = 0; i < n; i++) {                       // 条件结构，数据重复出现 n 次
    data_element                                // 组中的数据依赖于循环控制变量 i 的值
    ...                                         // 第一次循环时 i 设为 0，每次循环 i 自增
}
```

注意，这些组中的数据元素可能会遇到嵌套条件结构的情况。为了让程序看起来紧凑，在循环内部只有一个数据时省略“{}”。

```
data_element []                                 // data_element[]是一个数组。数据元素
                                                // 的个数由上下文指示。
data_element [n]                                // data_element[n]是数组中的第
                                                // n+1 个元素
data_element [m][n]                             // data_element[m][n]是二维数组中第
                                                // m+1，n+1 个数据元素
data_element [l][m][n]                          // data_element [l][m][n]是三位数组
                                                // 中第 l+1，m+1，n+1 个数据元素
data_element [m..n]                             // 是数据元素中第 m 到 n 中间的比特位
```

当用程序条目的形式表示语法时，不能再认为2.4.3节的实现满足解码过程。特别的，这定义了正确的而没有错误输入的比特流。事实上，解码器必须包括寻找到起始代码的方法，才能正确解码。

字节对齐函数的定义

字节对齐函数 bytealigned( )在当前位置是字节边界时返回 1，也就是说比特流中的下一位是一个字节的第一位。否则返回 0。

后继比特函数的定义

后继比特函数 nextbits( )允许比特流中将要解码的后继比特和一个比特串进行比较。

后继起始代码函数的定义

后继起始编码函数移除任何的填充为 0 比特和为 0 字节并查找下一个起始代码。

语法	比特数	标识符
next_start_code() {		
while (!bytealigned())		
zero_bit	1	'0'
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
zero_byte	8	'00000000'
}		

这个函数当前位置是否是字节对齐的。如果不是，则会出现为 0 的填充位。在这之后任意个数的为 0 字节后可能出现在起始代码前。因此起始代码总是字节对齐的并（在起始代码之前）可能被加上任意个数的填充位。

2.4 要求

2.4.1 音频比特流编码语法的规范

2.4.1.1 音频序列

语法	比特数	标识符

audio sequence()
{
while (nextbits()==syncword)
{
frame()
}
}		

2.4.1.2 音频帧

语法	比特数	标识符

frame()
{
header()
error_check()
audio_data()
ancillary_data()
}		

2.4.1.3 帧头

语法	比特数	标识符
header() {		
syncword	12	bslbf
ID	1	bslbf
layer	2	bslbf
protection_bit	1	bslbf
bitrate_index	4	bslbf
sampling_frequency	2	bslbf
padding_bit	1	bslbf
private_bit	1	bslbf
mode	2	bslbf
mode_extension	2	bslbf
copyright	1	bslbf
original/copy	1	bslbf
emphasis	2	bslbf
}		

2.4.1.4 差错校验

语法	比特数	标识符

```
error_check()
{
    if (protection_bit==0)
        crc_check                                   16      rpchof
}
```

2.4.1.5 音频数据，Layer I

语法	比特数	标识符
audio_data() {		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
allocation[ch][sb]	4	uimsbf
for (sb=bound; sb<32; sb++) {		
allocation[0][sb]	4	uimsbf
allocation[1][sb]=allocation[0][sb]		
}		
for (sb=0; sb<32; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
scalefactor[ch][sb]	6	uimsbf
for (s=0; s<12; s++)		
{		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
sample[ch][sb][s]	2..15	uimsbf
for (sb=bound; sb<32; sb++)		
if (allocation[0][sb]!=0)		
sample[0][sb][s]	2..15	uimsbf
}		
}		

2.4.1.6 音频数据，Layer II

语法	比特数	标识符
audio_data()		
{		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
allocation[ch][sb]	2..4	uimsbf
for (sb=bound; sb<sblimit; sb++) {		
allocation[0][sb]	2..4	uimsbf
allocation[1][sb]=allocation|0]|sb]		
}		
for (sb=0; sb<sblimit; sb++)		

for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0)		
scfsi[ch][sb]	2	bslbf
for (sb=0; sb<sblimit; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0) {		
if (scfsi[ch][sb]==0)		
{ **scalefactor[ch][sb][0]**	6	uimsbf
scalefactor[ch][sb][1]	6	uimsbf
scalefactor[ch][sb][2] }	6	uimsbf
if (scfsi[ch][sb]==1) || (scfsi[ch][sb]==3)		
{ **scalefactor[ch][sb][0]**	6	uimsbf
scalefactor[ch][sb][2] }	6	uimsbf
if (scfsi[ch][sb]==2)		
scalefactor[ch][sb][0]	6	uimsbf
}		
for (gr=0; gr<12; gr++) {		
for (sb=0; sb<bound; sb++)		
for (ch=0; ch<nch; ch++)		
if (allocation[ch][sb]!=0) {		
if (grouping[ch][sb])		
samplecode[ch][sb][gr]	5..10	uimsbf
else for (s=0; s<3; s++)		
sample[ch][sb][3*gr+s] }	3..16	uimsbf
for (sb=bound; sb<sblimit; sb++)		
if (allocation[0][sb]!=0) {		
if (grouping[0][sb])		
samplecode[0][sb][gr]	5..10	uimsbf
else for (s=0; s<3; s++)		
sample[0][sb][3*gr+s] }	3..16	uimsbf
}		
}		

2.4.1.7 音频数据，Layer III

语法	比特数	标识符
audio_data()		
{		
main_data_begin	9	uimsbf
if (mode==single_channel) **private_bits**	5	bslbf
else **private_bits**	3	bslbf
for (ch=0; ch<nch; ch++)		
for (scfsi_band=0; scfsi_band<4; scfsi_band++)		
scfsi[ch][scfsi_band]	1	bslbf
for (gr=0; gr<2; gr++)		

for (ch=0; ch<nch; ch++) {		
part2_3_length[gr][ch]	12	uimsbf
big_values[gr][ch]	9	uimsbf
global_gain[gr][ch]	8	uimsbf
scalefac_compress[gr][ch]	4	bslbf
window_switching_flag[gr][ch]	1	bslbf
if (window_switching_flag[gr][ch]) {		
block_type[gr][ch]	2	bslbf
mixed_block_flag[gr][ch]	1	uimsbf
for (region=0; region<2; region++)		
table_select[gr][ch][region]	5	bslbf
for (window=0; window<3; window++)		
subblock_gain[gr][ch][window]	3	uimsbf
}		
else {		
for (region=0; region<3; region++)		
table_select[gr][ch][region]	5	bslbf
region0_count[gr][ch]	4	bslbf
region1_count[gr][ch]	3	bslbf
}		
preflag[gr][ch]	1	bslbf
scalefac_scale[gr][ch]	1	bslbf
count1table_select[gr][ch]	1	bslbf
}		
main_data		
}		

主数据比特流定义如下。在 audio_data()语法中 **main_data** 域包含主数据流中的字节。但是，因为 Layer III 中使用的哈夫曼编码的（长度）变化性质，一个帧的主数据通常并不跟在当前帧的帧头和附属信息之后。一个帧的 main_data 在比特流中的起始位置位于本帧帧头之前，用 main_data_begin 的值以偏移量的形式给出（在附录 3-A 中图 3-A.7.1 参看定义）。

语法	比特数	标识符
main_data()		
{		
for (gr=0; gr<2; gr++) {		
for (ch=0; ch<nch; ch++) {		
if ((window_switching_flag[gr][ch]==1) &&		
(block_type[gr][ch]==2)) {		
if (mixed_block_flag[gr][ch]) {		
for (sfb=0; sfb<8; sfb++)		
scalefac_l[gr][ch][sfb]	0..4	uimsbf
for (sfb=3; sfb<12; sfb++)		
for (window=0; window<3; window++)		
scalefac_s[gr][ch][sfb][window]	0..4	uimsbf
}		

else {		
for (sfb=0; sfb<12; sfb++)		
for (window=0; window<3; window++)		
scalefac_s[gr][ch][sfb][window]	0..4	uimsbf
}		
else {		
if ((scfsi[ch][0]==0) || (gr == 0))		
for (sfb=0;sfb<6;sfb++)	0..4	uimsbf
scalefac_l[gr][ch][sfb]		
if ((scfsi[ch][1]==0) || (gr == 0))		
for (sfb=6;sfb<11;sfb++)	0..4	uimsbf
scalefac_l[gr][ch][sfb]		
if ((scfsi[ch][2]==0) || (gr == 0))		
for (sfb=11;sfb<16;sfb++)	0..3	uimsbf
scalefac_l[gr][ch][sfb]		
if ((scfsi[ch][3]==0) || (gr == 0))		
for (sfb=16;sfb<21;sfb++)	0..3	uimsbf
scalefac_l[gr][ch][sfb]		
}		
Huffmancodebits()		
}		
}		
for (b=0; b<no_of_ancillary_bits; b++)		
ancillary_bit	1	bslbf
}		

哈夫曼编码的语法如下：

语法	比特数	标识符
Huffmancodebits() {		
for (l=0; l<big_values*2; l+=2) {		
hcod[|x|][|y|]	0..19	bslbf
if (|x|==15 && linbits>0) **linbitsx**	1..13	uimsbf
if (x != 0) **signx**	1	bslbf
if (|y|==15 && linbits>0) **linbitsy**	1..13	uimsbf
if (y != 0) **signy**	1	bslbf
is[l] = x		
is[l+1] = y		
}		
for (; l<big_values*2+count1*4; l+=4) {		
hcod[|v|][|w|][|x|][|y|]	1..6	bslbf
if (v!=0) **signv**	1	bslbf
if (w!=0) **signw**	1	bslbf
if (x!=0) **signx**	1	bslbf
if (y!=0) **signy**	1	bslbf

```
            is[l] = v
            is[l+1] = w
            is[l+2] = x
            is[l+3] = y
        }
        for (; l<576; l++)
            is[l] = 0
}
```

2.4.1.8 填充数据

语法	比特数	标识符
ancillary_data() {		
if ((layer == 1) || (layer == 2))		
for (b=0; b<no_of_ancillary_bits; b++)		
ancillary_bit	1	bslbf
}		

2.4.2 音频比特流的语法的语义学

2.4.2.1 音频序列概述

frame Layer I 和 Layer II：比特流中的部分可以自解码。在 Layer I 中包含 384 个采样信息而 Layer II 中包含 1152 个采样。帧由同步字开始，在下一个同步字之前结束。一个帧包含整数个填充位（对 Layer I 为 4 个字节，对 Layer II 为 1 个字节）。Layer III：比特流中的一部分可通过预先获得的主要信息来解码。在 Layer III 中包含了 1152 个采样信息。尽管连续同步字的起始处之间的距离为整数个插入域（在 Layer III 中为一个字节），但是属于一帧的音频信息并不完全包含在两个连续的同步字中间。

2.4.2.2 音频帧

header 比特流中包含同步信息和状态信息的部分。

error_check 比特流中包含差错检测信息的部分。

audio_data 比特流中包含音频采样数据信息的部分。

ancillary_data 比特流中可能用作辅助数据的部分。

2.4.2.3 帧头

一帧的前 32 位（4 字节）是帧头信息，这在所有层中都一样。

syncword 比特串“1111 1111 1111”[1]。

ID 一个比特来表示算法 ID。1 表示 MPEG，0 保留。

Layer 两个个比特来表示使用的层，具体见表 2.1：

表 2.1 层数索引

数值	层数
00	保留
01	Layer III
10	Layer II
11	Layer I

[1]: 这种说法只是在本标准中适用，在 MPEG2 中修改了这部分定义，采用 11 个连续的“1”来表示帧同步字，而 ID 部分则用两个比特来表示。

如果改变层数，那么需要音频解码器复位。

protection_bit 一个比特位用来指示是音频比特流中否添加冗余校验以有助于检测差错和潜在错误。为“1”表示没有冗余校验，为“0”表示有冗余校验。

bitrate_index 指示比特率。全为 0 表示“自由格式”情况，可以使用不是表 2.2中的固定比特率。固定表示一帧包含 N 或者 N+1 个字节，取决于填充位的值。比特率索引是表 2.2的索引值，不同层值不一样。

bitrate_index（比特率索引）表示与模式（立体声、联合立体声、双声道、单声道）无关的总比特率。

表 2.2 比特率索引[2]

索引值	MPEG1			MPEG2	
	层 1	层 2	层 3	层 1	层 2、3
0000	free				
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	forbidden				

[2]: 表中 MPEG2、MPEG2.5 部分的索引值为译者加上去的，在本标准中并未体现出来。

为了提供最小可能延时和复杂度，在 Layer I 和层 Layer II 中并不要求解码器支持连续变化的比特率。Layer III 通过交替比特率索引支持变比特率。比特率索引的交替既可以被用来优化 DSM 上的存储需求，也可以通过在比特率查询表中交替接近的（比特率）值以达到调整平均数据速率的目的。然而，在自由模式下，必须要求是固定比特率。在自由模式下解

码器也不要求支持高于 448Kbit/s、384Kbit/s、320Kbit/s 的比特率，分别对应 Layer I、II 和 III 的最高比特率。

对于Layer II，并不是所有总比特率和模式的组合都是允许的。见表 2.3。

表 2.3 Layer II 中比特率和允许的模式组合

比特率（Kbit/s）	允许模式
free 格式	所有模式
32	单声道
48	单声道
56	单声道
64	所有模式
80	单声道
96	所有模式
112	所有模式
128	所有模式
160	所有模式
192	所有模式
224	立体声、强度立体声、双声道
256	立体声、强度立体声、双声道
320	立体声、强度立体声、双声道
384	立体声、强度立体声、双声道

sampling_frequency 指示采样频率，具体见表 2.4：

表 2.4 采样率索引

Bits	MPEG 1	MPEG 2	MPEG 2.5
00	44100	22050	11025
01	48000	24000	12000
10	32000	16000	8000
11	保留		

改变采样率时，需要解码器复位。

padding_bit 如果该位为“1”，那么此帧包含一个用来将平均比特率校正到和采样频率一致的插入域，否则该位必须为“0”。填充位在采样频率为 44.1KHz 时是必须的。比特率为自由格式时也可能要求填充位。

填充位应该被应用在这样的比特流中：经过一定数量的音频帧之后，编码帧的累加长度和下面的计算值的误差不超过（+0，-1）个插入域（字节数）：

$$\text{累加的帧长度} = \sum_{\text{第一帧}}^{\text{当前帧}} (\text{帧大小} \times \text{比特率}) / \text{采样频率}$$

这里帧大小 = 384，对于 Layer I，

1152 对于 Layer II 和 Layer III。

下面的方法可以被用来决定是否使用填充位：

对于第一个音频帧：

```
rest = 0;
padding = no;
```

对于后面的每个音频帧：

```
if (Layer == 1)
    dif = (12 * bitrate) % sampling_frequency;
else
    dif = (144 * bitrate) % sampling_frequency;
rest = rest - dif;
if (rest < 0) {
    padding = yes;
    rest = rest + sampling_frequency;
}
```

private_bit 私有使用的位。ISO 在以后将不使用该位。

mode 指示模式，具体见表 2.5。在Layer I和Layer II中联合立体声模式即为强度立体声，在 Layer III 中则为强度立体声或者中边立体声。

表 2.5 模式索引

值	模式
00	立体声
01	联合立体声（强度立体声/中边立体声）
10	双声道
11	单声道

在 Layer I 和 II 中，除了联合立体声之外的所有模式，边界（bound）的值都等于 sblimit。在联合立体声模式中边界由模式扩展（mode_extension）决定。

mode_extension 这些位用在联合立体声模式中。在 Layer I 和 II 中指示那些子带是强度立体声（模式）。所有其他子带则是立体声编码。具体见表 2.6。

表 2.6 模式扩展

模式扩展（Layer I 和 II）	含义
00	子带 4 – 31 为强度立体声，边界 == 4
01	子带 8 – 31 为强度立体声，边界 == 8
10	子带 12 – 31 为强度立体声，边界 == 12
11	子带 16 – 31 为强度立体声，边界 == 16

在层III中模式扩展表示使用的是哪种立体声编码方式，具体参见表 2.7。强度立体声和中边立体声模式应用的频率范围暗含在算法中。详见2.4.3.4。

表 2.7 模式扩展索引

模式扩展（Layer III）	强度立体声	中边立体声
00	off	off
01	on	off
10	off	on
11	on	on

注意如果模式位中指定的是立体声，或与之相等的模式位指定联合立体声且模式扩展中指定的强度立体声和中边立体声都为“off”，那么使用的是立体声。

copyright “0”表示 MPEG/音频比特流没有版权，“1”则表示受版权保护。

original/copy 该位为“0”表示是拷贝，为“1”表示是原创。

emphasis 表示应该使用的去加重类型，具体见表 2.8：

表 2.8 强调类型

值	加重类型
00	无强调
01	50/15 毫秒
10	保留
11	CCITT J.17

2.4.2.4 错误校验

crc_check 使用 16 位奇偶校验字作编码比特流中可选的差错校验。

2.4.2.5 音频数据，Layer I

allocation[ch][sb] 表示编码声道 ch 中的子带 sb 的采样所使用的 bit 数。对于在强度立体声模式中的子带而言比特流中每个子带仅包含一个分配的数据元素。见表 2.9。

表 2.9 allocation[ch][sb]分配

allocation[ch][sb]	每采样比特数
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
10	11
11	12
12	13
13	14

续上表

allocation[ch][sb]	每采样比特数
14	15
15	无效

注意：对于编码“0000”表示没有采样将被传输。

scalefactor[ch][sb] 表示声道 ch 中子带 sb 的因子，在解量化声道 ch 中子带 sb 的采样时应该乘上的数值。这六个比特组成一个无符号整数，是附录 3-B 中表 3-B.1“Layer I，II 比例因子”的索引。

sample[ch][sb][s] 声道 ch 中子带 sb 内第 s 个采样的编码表示。对于强度立体声模式中的子带而言采样的编码表示在两个声道都有效。

2.4.2.6 音频数据，Layer II

allocation[ch][sb] 包含了供声道 ch 中子带 sb 的采样使用的量化器信息，（具体为）三个连续采样的信息是否被分组到一个编码，以及编码这些采样所使用的比特数。该域的含义和长度取决于子带的序号、比特率和采样频率。从一个无符号整数中提取出该域的比特，被用作与之相关的附录 3-B，表 3-B.2“Layer II 比特分配表”的索引，该表给出了量化使用的数量级别。对强度立体声模式中的子带而言比特流中每个子带仅包含一个分配数据元素。

scfsi[ch][sb] 比例因子选择信息。给出声道 ch 中子带 sb 内转换的比例因子的数量以及本帧中信号的那些部分是有效的。帧被分成三个相等部分，每个子带包含 12 个子带采样。见表 2.10。

表 2.10 scfsi[sb]描述

scfsi[sb]	描述
00	传输三个比例因子，分别对应第 0、1、2 部分
01	传输两个比例因子，第一个对第 0、1 部分有效，第二个对第 2 部分有效
10	传输一个比例因子，对所有三个部分都有效
11	传输两个比例因子，第一个对第 0 部分有效，第二个对第 1 和 2 部分有效

scalefactor[ch][sb][p] 指示帧中解量化的声道 ch 中子带 sb 内第 p 部分采样应该乘以的因子。六个比特组成一个无符号整数，是附录 3-B，表 3-B.1“Layer I、II 比例因子”的索引。

grouping[ch][sb] 是一个函数，决定分组（操作）是否影响对声道 ch 中子带 sb 内采样的编码。分组的含义是，当 Grouping[ch][sb]为真时，当前声道 ch 中子带 sb 内的三个连续采样和当前颗粒 gr 使用一个常规编码字编码传输而不使用三个独立的编码字，如果比特分配表正在使用（见 3-B.2），那么在 sb（行）和 allocation[sb]（列）中找到的值是 3、5 或者 9。否则该值为假。对于强度立体声模式中的子带分组对两个声道都有效。

samplecode[ch][sb][gr] 在声道 ch 中子带 sb 内颗粒 gr 里连续三个采样的编码表示。对于强度立体声模式而言采样码的编码表示在两个声道都有效。

sample[ch][sb][s] 声道 ch 中子带 sb 内第 s 个采样的编码表示。对于强度立体声模式而言采样的编码表示在两个声道都有效。

2.4.2.7 音频数据，Layer III

下面一些场合的定义包含下标“[ch]”是因为每个声道需要独立的值。Layer III 的语义学使用一个名为“nch”的参数来指定“[ch]”的范围。参数“nch”在单音模式时值为 1，所以其他模式值都为 2。

main_data_begin main_data_begin 的值被用来确定一个帧的主要数据的第一个比特的位置。main_data_begin 的值以一个相对于音频同步字第一个字节的负偏移量（以字节为单位）的形式指定了（主数据的起始）位置。属于帧头和附属信息的字节数并没有计算在内。例如，如果 main_data_begin == 0，那么主数据起始于附属信息之后。附录 3 中图 3-A.7.1 给出了示例。

private_bits 私有使用的位。ISO 在以后将不使用这些位。

scfsi[ch][scfsi_band] 在 Layer III 中比例因子选择信息工作机理类似于 Layer I 和 II。主要的不同是用变量 scfsi_band 使 scfsi 应用到比例因子的分组，以替代单个的比例因子。scfsi控制着比例因子到颗粒中的使用。具体见表 2.11。

表 2.11 scfsi[scfsi_band]描述

scfsi[scfsi_band]	含义
0	比例因子每个颗粒独自传输
1	传输给颗粒 0 的比例因子在颗粒 1 中也有效

如果短窗被选中，例如其中一个颗粒的 block_type == 2，那么这一帧中的 scfsi 总是 0。

scfsi_band 在比例因子（scfsi_bands）的分组中控制着比例因子选择信息的使用。具体见表 2.12。

表 2.12 scfsi_band 描述

scfsi_band	比例因子带（见附录 3-B，表 3-B.8）
0	0，1，2，3，4，5，
1	6，7，8，9，10，
2	11…15
3	16…20

part2_3_length[gr][ch] 该值包含了主数据中比例因子和哈夫曼编码使用的比特数。因为附属信息的长度总是相同的，这个值可以用来计算每个颗粒主要信息的起始处和填充信息的位置（如果用了的话）。

big_values[gr][ch] 每个颗粒的频谱值是用不同的哈夫曼表来编码的。从 0 到奈奎斯特采样频率的整个频带范围被分为许多区域，并用不同的表(哈夫曼表)来编码。频带分区是根据量化的最大值来进行的。这是建立在认为高频区域的频谱幅值较低或者不需要进行编码的假设上。从高频区域开始，一对量化的值如果等于 0 则被计数。这个数值被命名为“rzero”。同样的，四个量化的值如果其绝对值不超过 1（例如，仅 3 种可能的量化级别的情况）则被计数，这个数值就被命名为“count1”。因此，

计数值（的结果）保留为偶数。最后，频谱中一直下降到频率为 0 的区域中，（量化值）对的个数被命名为“big_values”。这个范围内绝对值的最大值被限制在 8191 以内。下面的图解释了频谱分区：

```
xxxxxxxxxxxx-------------------------000000000000000000000000000
|           |                         |                         |
1           bigvalues×2               bigvalues×2+count1×4      iblen
“000”       值就是全部为 0。
“---”       值是-1、0 或+1。他们的数目应该乘以 4。
“xxx”       值没有范围。
iblen       值是 576.
```

global_gain[gr][ch] 量化器的量化步长信息在附属信息的变量 global_gain 中传输。它是对数量化的。对于**global_gain**的应用，参考2.4.3.4中的公式“解量化和全缩放的公式”。

scalefac_compress[gr][ch] 根据表 2.13来选择传输比例因子所使用的比特位的个数。

```
if block_type is 0, 1, or 3:
    slen1:  比例因子带 0 - 10 中比例因子长度
    slen2:  比例因子带 11 - 20 中比例因子长度
if block_type is 2 and mixed_block_flag is 0:
    slen1:  比例因子带 0 - 5 中比例因子长度
    slen2:  比例因子带 6 - 11 中比例因子的长度
if block_type is 2 and mixed_block_flag is 1:
    slen1:  比例因子带 0 - 7（long window scalefactor band）和 3 - 5（short
            window scalefactor band）中比例因子长度。注意：比例因子带 0 - 7
            是来自“long window scalefactor band”表中，比例因子带 3 - 11
            是来自“short window scalefactor band”表中。这种分区组合是连续
            并且跨越整频带范围的。
    slen2:  比例因子带 6 - 11 中比例因子长度
```

表 2.13 scalefac_compress[gr]描述

scalefac_compress[gr]	slen1	slen2
0	0	0
1	0	1
2	0	2
3	0	3
4	3	0
5	1	1
6	1	2
7	1	3
8	2	1
9	2	2
10	2	3
11	3	1

续上表

scalefac_compress[gr]	slen1	slen2
12	3	2
13	3	3
14	4	2
15	4	3

window_switching_flag[gr][ch] 用来表示块使用了不是标准窗（type 0）的标志。

如果设置了 window_switching_flag，那么默认也设置了一些其他变量：

region0_count = 7 （在**block_type** == 1 或者 **block_type** == 3 或者 **block_type** ==2 且设置 mixed_block_flag 的情况下）

region0_count = 8 （在**block_type** == 2 且未设置 mixed_block_flag 的情况下）

region1_count = 63 在 big_value 区域中所有剩余的值都包含在 region 1 中。

如果没有设置 window_switching_flag，那么 **block_type** 的值为 0。

block_type[gr][ch] 指出实际颗粒的加窗函数类型（参见 Layer III 中滤波器组的描述）。如表 2.14所示。

表 2.14 block_type[gr]描述

block_type[gr]	值
0	保留
1	start block
2	3 short windows
3	end block

block_type（块类型）和**mixed_block_flag**（混合块标记）给出了块中数值的组织信息和与转换有关的长度和数量信息（见附录 3-A，图 3-A.4 中的方案，附录 3-C中的分析描述）。如果window_switching_flag ==1，那么mixed_block_flag指示了低频多相滤波器子带是否使用标准窗类型编码。多相滤波器的描述见2.4.3。

在 long blocks（长块）情况下（block_type 不为 2 或者 block_type 为 2 时较低的子带中）IMDCT 每输入 18 个数据产生 36 个输出。输出数据根据 block_type 而加窗，并且前半部分和前一个块的后半部分交叠。作为结果的矢量是一个子带中多相滤波器的合成部分的输入。

在 short blocks（短块）情况下（block_type 为 2 时较高的子带）处理产生的 12 个输出数据时各自用到了 3 次变换。这三个矢量被加窗并且相互交叠。作为结果的矢量中两端都加上 6 个 0，使得矢量长度为 36，该矢量以和长变换的输出一样的方式被处理。

mixed_block_flag[gr][ch] 表示较低频率部分是用一个与高频区域不用的窗类型转换。如果 mixed_block_flag 为 0，那么所有块都是以 block_type[gr][ch]中的（块类型）转换。如果 mixed_block_flag 为 1，那么频率线对应的两个最低频率的多相子带以标准窗（block_type == 0）转换，而剩下的 30 个子带以 block_type[gr][ch] 转换。

table_select[gr][ch][region] 根据最大的量化值和信号的局部统计学特性而选择不同的哈夫曼编码表。附录 3-B 中表 3-B.7 中总共有 32 个可能的哈夫曼表。

subblock_gain[gr][ch][window] 根据全局增益来表示一个子块的增益偏移（量化因子是

4）。只在块类型是 2 时（short windows）时使用。在解码器中，子块的值必须除以 $4^{subblock\_gain[window]}$。

region0_count[gr][ch] 为了更进一步提高哈夫曼编码的性能而对频谱进行了更细的划分。这是 big_values 所描述区域中的细分。进行这样细分的目的是为了获得更好的鲁棒性和编码效率。总共使用 3 个区域，分别命名为：regin0，1 和 2。每个区域用不同的哈夫曼表来编码，这取决于最大量化值的大小和信号的局部统计特性。

regin0_count 和 region1_count 的值被用来表示这些（频谱）区域的边界。区域的边界与频谱在比例因子带上的分区对齐。

region0_count 域包含一个小于比例因子带在 region 0 中的数量（的子带）。在短块情况下，每个比例因子带对应每个短窗而被计数 3 次，因此如果 region0_count 的值为 8 表示 region1 的起始处在编号为 3 的比例因子带。

如果 block_type == 2 并且 mixed_block_flag == 0，在这种情况下颗粒的比例因子带总数为 12×3 = 36。如果 block_type == 2 并且 mixed_block_flag == 1，比例因子带的数量是 8 + 9 ×3 = 35。如果 block_type != 2，比例因子带的数量是 21。

region1_count[gr][ch] region1_count 计数了一个小于比例因子带在 region 1 中的数量（的子带）。同样如果 block_type == 2，那么比例因子带表示不同的时间域是分别计数的。

preflag[gr][ch] 这是量化值在附加高频时应用的捷径。如果设置了 preflag，附录 3-B 中的表 3-B.6 就会加到比例因子中。这等价于解量化的比例因子值乘以了表中的值。在 block_type 为 2（短块）的块类型中，preflag 标志不使用。

scalefac_scale[gr][ch] 比例因子根据 scalefac_scale 中的值以 2 或者 √2 为步长进行对数量化的。表 2.15表示了在解量化等式中每个（量化）步长使用的比例因子乘数。

表 2.15 scalefac_scale[gr]描述

scalefac_scale[gr]	scalefac 乘数
0	0.5
1	1

count1table_select[gr][ch] 对于数量级不超过 1 而采用四个值为一组量化的区域，这个标志决定从两个可能的哈夫曼编码表中选择一个（表）。见表 2.16。

表 2.16 count1table_select[gr]描述

count1table_select[gr]	选择的表
0	附录 3-B.7 的表 A
1	附录 3-B.7 的表 B

scalefac_l[gr][ch][sfb]，scalefac_s[gr][ch][sfb][window] 比例因子用来修饰量化噪声。如果用适合形式去修饰噪声，那么噪声将被完全遮蔽。和 Layer I、Layer II 不同，Layer III 的比例因子对局部量化信号的最大值没有任何作用。在 Layer III 中，解码器通过比例因子来获得块（block）的值的除数因子。在这种情况下的 Layer III

中，块延展而跨越几个频率线。这些块被称作比例因子带，并且尽可能像临界带一样分组。

scalefac_compress 表表示比例因子 0－10 的范围是 0－15（最大比特数是 4 位），而比例因子 11－21 的范围是 0－7（最大比特数是 3 位）。

如果使能了强度立体声（扩展模式中），不同（右）声道中“zero_part”的比例因子被用作强度立体声的位置（参看2.4.3.4，MS立体声模式）。

将频谱划分为比例因子带对每个块长度和采样频率而言都是固定的，并且存放在编码器和解码器的表中（参看附录 3，表 3-B.8）。表中在最高频率线以上的频率线的比例因子为 0，这表示实际上的乘数因子是 1.0。

增益因子是对数量化的。量化步长取决于 scalefac_scale。

huffmancodebits( ) 哈夫曼编码数据。

huffmancodebits( )的语法表示了量化值是如何编码的。在 big_values 区，绝对值小于 15 的一对量化值直接使用哈夫曼代码进行编码。编码码元是从表 3-B.7 的哈夫曼表 0－31 中选择的。并且总以一对（x，y）的值进行编码。如果量化值的大小大于等于 15，这些值在哈夫曼代码之后一个单独的域中进行编码。如果一个数对中有一个或两个值不为零，那么在相应的编码字中会加上符号位。

big_values 区域的哈夫曼表由以下三个参数组成：

`hcod[|x|][|y|]`	是值 x，y 的哈夫曼编码表的入口。
`hlen[|x|][|y|]`	是值 x，y 的哈夫曼长度表的入口。
`linbits`	是 linbitsx 或 linbity 编码时的长度。

huffmancodebits 的语法包含了下列域和参数：

signu	是 u 的符号位（0 为正，1 为负）。
signv	是 v 的符号位（0 为正，1 为负）。
signx	是 x 的符号位（0 为正，1 为负）。
signy	是 y 的符号位（0 为正，1 为负）。
linbitsx	如果 x 的大小大于等于 15 则被用来编码 x 的值。这个域只在当 hcod 中的|x|等于 15 时编码。如果 linbits 为 0，因此实际上当|x| == 15 时并没有编码，所以 linbitsx 的值被定义为 0。
linbitsy	和 linbitsx 的含义一样，但是针对 y。
is[l]	是编号为 l 的频率线的量化值。

linbitsx 或 linbitsy 域只在需要编码的值大于或等于 15 时使用。这个域被认为是无符号整数并且被加到 15 以获取编码值。如果选择的（编码）表是最大量化值小于 15 的块时，linbitsx 和 linbitsy 域不能使用。注意值 15 也可以和一个 linbits 为 0 的哈夫曼表一起编码。在这种情况下，linbitsx 或者 linbitsy 域并没有实际编码，因为 linbits 为 0。

在 count1 区，四个幅度值小于或者等于 1 的值一起编码。幅值同样也用表 3-B.7 中的哈夫曼代码表 A 或 B 来编码。对于每个非零值，在哈夫曼码元后面也会加上一个符号位。

count1 区域的哈夫曼表由以下参数组成：

`hcod[|v|][|w|][|x|][|y|]`	是值 v、w、x、y 的哈夫曼编码表入口
`hlen[|v|][|w|][|x|][|y|]`	是值 v、w、x、y 的哈夫曼长度表入口

哈夫曼编码表 A 并不是真正的四维编码，因为它是由很简易的码元构成：0 被编码为 1，而 1 被编码为 0。

在 count1 区以上的量化值全部为 0，因此它们没有被编码。

为清楚起见，本文中的参数“count1”用来表示哈夫曼码元在 count1 区域的数目。然而，和 big_values 区域不一样，在编码语法中 count1 区域值的数目并没有直接用一个区域来编码。count1 区域的末尾只有在当前颗粒（由 part2_3_length 指定）的所以比特都被用尽后才能确定，而 count1 的值只有在解码完 count1 区域之后才能间接得到。

哈夫曼数据的顺序取决于当前颗粒的 block_type（块类型）。如果 block_types 是 0、1 或 3，则编码后的哈夫曼数据是根据频率递增的方式排序的。

如果 block_type == 2（短块），则编码后的哈夫曼数据以当前颗粒的比例因子值的顺序排序。编码后的哈夫曼数据被映射到连续的比例因子带，起始于比例因子带 0。在各个比例因子带内，这些数据又被映射到连续的时间窗函数，起于窗 0，终于窗 2。在各个窗内，量化数据又是按频率递增的方式排列的。

2.4.2.8 填充数据

Ancillary_bit 用户定义。

填充数据的数目（no_of_ancillary_bits）等于在一个音频帧内可用的比特数减去帧头、差错校验和音频数据实际使用的比特数。在 Layer I 和 II 中填充数据的数目等于音频数据的结尾与下一个帧头之间的距离。在 Layer III 中填充数据的数目等于 Huffman_code_bits 的结尾和比特流中下一个 main_data_begin 指针指向的位置之间的距离。

2.4.3 音频解码过程

2.4.3.1 总括

第一步就是将解码器和输入比特流同步。在启动之后可以通过在比特流中寻找 12 位的同步字来实现。在一些应用中，ID 号，层数和保护状态均已被解码器知晓，所以帧头的前 16 个比特应该被当做一个 16bit 的同步字，这样使得同步更可靠。连续同步字的位置可以通过同步字后的第 7 比特提供的信息来计算：比特流是被划分为（多个）插入域。两个连续同步字的距离是一个常量，并且等于“N”个插入域（字节）。“N”的值取决于层数。对 Layer I 有：

$$N = 12 \times \frac{bitrate}{sampling\_frequency}$$

对 Layer II 和 Layer III 有：

$$N = 144 \times \frac{bitrate}{sampling\_frequency}$$

如果计算出来的结果不是整数，结果将下取整（忽略小数），这时需要填充（padding）。这种情况下一帧的插入域数目将会在 N 和 N+1 之间变动。当插入域的数目是 N 时，填充位置零，其他情况为 1。连续同步字的位置理论使得同步被极大地简化。

如果比特率索引中的值是“0000”，则并没有指出精确的比特率。N 可以通过连续的同步字的距离和填充位的值来确定。

如果帧头中的保护位为 0，表示在帧头后面增加了一个 CRC 校验字。差错检测方法使用的是“CRC-16”，其生成多项式如下：

$$G(X) = X^{16} + X^{15} + X^2 + 1$$

附录 3-B，表 3-B.5 中给出了 CRC 校验中包含的比特。

附录 3-A，图 3-A.9“CRC 校验图”描述了校验方法。移位寄存器的初始状态是“1111 1111 1111 1111”。之后包含 CRC 校验在内的所有比特都输入到附录 3-A，图 3-A.9“CRC 校验图”中的电路。输出 b15–b0 组成一个字，并被用来和比特流中的 CRC 校验相比较。如果两个 CRC 校验字不一样，表示比特流中受保护的区域发生了一个传输错误。为了避免烦人的失真，建议增加一些伪装技术，比如将音频帧静音或重发上一帧。

2.4.3.2 Layer I解码流程

在所有层公共的解码部分（见2.4.3.1）之后，读取所有子带的比特分配信息和所有分配的比特数非零的子带中的比例因子。附录 3-A，图 3-A.1“Layer I和II解码流程图”给出了解码流程。

- **解量化子带采样**

从比特分配中读取的每个子带内的采样的比特数nb已知。各种模式下采样的排列顺序在2.4.1.5节中给出。在从比特流中集齐了一个采样的比特之后，第一个比特必须被反转。作为结果的数值可以被认为是一个二进制补码表示的分数，而MSB表示值-1。解量化的值可以通过使用下面的线性公式得到：

$$s'' = \frac{2^{nb}}{2^{nb} - 1} \times (s''' + 2^{-nb+1})$$

其中，s''' 是分数；

s'' 是解量化的值；

nb 是子带中分配给采样的比特数。

在强度立体声模式下子带中的采样必须被拷贝到两个声道。解量化的值必须重新缩放。乘数因子可以在附录 3-B，表 3-B.1“Layer I、II 比例因子”中找到。重新缩放的值 s' 计算如下：

$$s' = factor \times s''$$

- **综合子带滤波器**

如果一个子带没有比特分配给它，则该子带内的采样被设置为 0。每次一个声道所有 32 个子带的子带采样都被计算，它们可以被应用到综合子带滤波器，并且 32 个连续的音频采样可以被计算得出。附录 3-A，图 3-A.2“综合子带滤波器流程图”中的步骤表示了（声音的）重建操作。矩阵运算时的系数 N_{ik} 由下面的公式得出：

$$N_{ik} = \cos[(16 + i)(2k + 1)\frac{\pi}{64}], \text{for i} = 0 \text{ to } 63, \text{k} = 0 \text{ to } 31$$

加窗操作的系数 D_i 可以在附录 3-B，表 3-B.3“综合加窗的系数 D_i”中找到。该系数经过数字优化得出。一个帧包含 12×32 = 384 个子带采样，而经过滤波后的结果，是 384 个音频采样。

2.4.3.3 Layer II解码流程

Layer II是比Layer I更有效但是更复杂的编码方案。附录 3-A，图 3-A.1“Layer I和II解码流程图”适用于Layer I和II。第一步就是执行所有三个层共同的解码操作（见2.4.3.1节）。

- **比特分配的解码**

对于比特率和采样频率的不同组合，存在不同的比特分配表（附录 3-B，表 3-B.2“Layer II 比特分配表”）。注意表头中给出的比特率是每个声道的。如果模式不是单声道，比特率应该除以 2 以获得每个声道的比特率。比特分配表的解码分三步完成。第一步包含从比特流中读取一个子带“nbal”（2、3 或 4）个比特的信息。“nbal”的值在相关的附录 3-B，表 3-B.2“Layer II 比特分配表”中的第二列给出。这些比特应该被解释为一个无符号整数。第二步用这个数和子带数作为索引指向上表中的一个值。这个值代表用来量化子带中采样的级别“nlevels”的数量。作为第三步，使用附录 3-B，表 3-B.4“Layer II 量化种类”可以获得用来编码已量化采样的比特数、解量化系数和三个连续子带采样的编码是否被组合为一个编码的信息。可以从比特分配表中看出一些最高的子带根本没有分配比特。没有比特分配的最低子带的数目则被分配到标示符“sblimit”。

- **比例因子选择信息的解码**

一帧中一个子带内的 36 个采样被分为三个相等的部分，每部分 12 个子带采样。各个部分可以有自己的比例因子。必须从比特流中读取的比例因子的数目取决于 scfsi[sb]。从比特流中读取比例因子选择信息 scfsi[sb]给有非零比特分配的子带。如果 scfsi[sb]等于“00”则传输三个比例因子，分别对应第 0、1、2 部分。如果 scfsi[sb]等于“01”则传输两个比例因子，第一个对第 0、1 部分有效，第二个则给第 2 部分。如果 scfsi[sb]等于“10”则只传输一个比例因子，对所有三个部分都有效。如果 scfsi[sb]等于“11”则传输两个比例因子，第一个对第 0 部分有效，第二个则给第 1、2 部分。

- **比例因子解码**

对每个分配了非零比特数的子带，该子带已编码的比例因子可从比特流中读取。涉及到的已编码的比例因子数目和子带采样部分由 scfsi[sb]定义。比例因子的 6 个比特的编码应该被解释为一个无符号整数，是附录 3-B，表 3-B.1“Layer I、II 比例因子”的索引。该表给出相关子带采样在解量化后应该被乘以的比例因子。

- **解量化子带采样**

接下来是读取编码的采样。如2.4.1.6节中所示，编码的采样以三个一组出现，编码中一次包含三个连续采样。从附录 3-B，表 3-B.4“Layer II量化种类”中可知，需要从比特流中读取多少个比特给每个子带中的一个三元组。从附录 3-B，表 3-B.4“Layer II量化种类”也可得知，当前编码是否由三个连续可分离的码元（对每个采样）组成还是三个采样的组合代码（分组）。在后一种情况下必须去除分组。组合的编码必须被认为是一个无符号整数，称为“c”。下面的算法将产生三个分离的码元，s[0]、s[1]、s[2]：

```
for (i=0; i<3; i++) {
    s[i]= c % nlevels
    c = c DIV nlevels
}
```

其中 nlevels 是附录 3-B，表 3-B.2“Layer II 比特分配表”中表示的阶数。

三个码元各自的第一个比特必须被反转，并且作为结果的数字应该被认为是二进制补码表示的分数，其中 MSB 代表值-1。解量化的值可以通过使用下面的线性公式获得：

$$s'' = C \times (s''' + D)$$

其中，s''' 是分数；

s'' 是解量化的值。

常数 C 和 D 的值在附录 3-B，表 3-B.4“Layer II 量化种类”中给出。解量化的值必须被重新缩放。乘数因子可以在附录 3-B，表 3-B.1“Layer I、II 比例因子”中找到。如前所述，重新缩放的值 s' 计算如下：

$$s' = factor \times s''$$

- **综合子带滤波**

如果一个子带没有分配比特，则该子带中的采样被设置为 0。每次一个声道所有 32 个子带的子带采样都被计算，它们可以被应用到综合子带滤波器，并且 32 个连续的音频采样可以被计算得出。正因如此，流程图附录 3-A，图 3-A.2“综合子带滤波器流程图”中的步骤必须被执行。矩阵运算时的系数 N_{ik} 由下面的公式得出：

$$N_{ik} = \cos[(16+i)(2k+1)\frac{\pi}{64}], \text{for i} = 0 \text{ to } 63, \text{k} = 0 \text{ to } 31$$

加窗操作的系数 D_i 可以在附录 3-B，表 3-B.3“综合加窗的系数 D_i”中找到。一个帧包含 36×32 = 1152 个子带采样，而经过滤波后的结果，是 1152 个音频采样。

2.4.3.4 Layer III解码流程

混合滤波器组的应用，使得频率精度得到提高。应用 MDCT 将每个频带再划分为 18 个频率线。MDCT 窗的长度是 36。用自适应窗选择来控制时间假象（前回音），见附录 3-C 中的描述。以上的频率也可以选用较短的块来获得较高的时间精度。信号中低于一个频率之下（由“mixed_block_flag”确定）的部分编码的频率精度较高，在这之上的部分信号则编码的时间精度较高。

频率成分使用一个不均匀的量化器量化并用哈夫曼编码器编码。哈夫曼编码器使用了 18 个不同的表（见附录 3-B.7）。这个过程中使用了一个缓冲来提高哈夫曼编码的效率，并在前回音（见附录 3-C 中的描述）情况下起到帮助效果。对 Layer III 来说，输入缓冲的大小和在每声道 160kbps 码率下一个帧的大小一样。使用的短期缓冲技术被称为“比特池”，因为它使用了来自平均比特率中的一个最大整体偏移量的短期可变特率。

每个帧保存了两个颗粒的数据。一帧中的音频数据以下面的方式分配：

```
--main_data_begin 指针；
--两个颗粒的附属信息（scfsi）；
--颗粒 1 的附属信息；
--颗粒 2 的附属信息。
```

帧头和这部分的音频数据组成了附属信息流：

```
--比例因子和颗粒 1 的哈夫曼编码数据
--比例因子和颗粒 2 的哈夫曼编码数据
--填充数据
```

这些数据组成了主要数据流。main_data_begin 指针指定了一个距帧头第一个字节处的偏移量。

- **解码**

第一步就是让解码器和输入比特流同步，这步的完成和其他层一样。帧头信息（包括同步字的前 32 个比特）也和其它层一样读入。采样频率的信息可以用来选择 scalefactor_band 表（见附录 3-B.8）。

- **附属信息**

解码附属信息需要存储解码参数。表的选择信息被用来选择解码查询表和 ESC-bits（linbits）的数量，根据附录 3-B.7 中的表。

- **主要数据的开始**

main_data（比例因子、哈夫曼编码数据和填充信息）并不一定位于附属信息之后。这在附图 3-A.7.1 和附图 3-A.7.2 中有描述。主要数据部分的开始是用当前帧的 main_data_begin 指针来定位。主要数据的配置是以这样的方式完成的：当下一帧的帧头到达输入缓冲时（当前帧的）所有主要数据都存在于输入缓冲中。在解码主要数据时，解码器必须略过帧头和附属信息。可以通过比特率和填充位来计算主数据的位置。帧头的长度总是 4 个字节，单声道模式下附属信息长度为 17 个字节，其他模式则为 32 个字节。主要数据可以跨越不止一个帧头和附属信息（见附图 3-A.7.2）。

- **缓冲的考虑**

下面的规则可以用来计算一个颗粒使用的最大比特数：

缓冲长度为 7680 个比特。这个值用作任意比特率下每声道的最大缓冲。在 Layer III 的最高可能的比特率（320 kbit/s per stereo）且采样频率为 48KHz 时，平均帧长度是：

$$320000 / 48000 \times 1152 = 7680 \quad \text{bits}$$

因此在这个比特率和采样频率条件下帧必定是定长的。在 64 kbit/s（128 kbit/s stereo）48KHz 采样频率的情况下，平均颗粒长度是 64000 / 48000×576＝768 bit。这就意味着在 64 kbit/s 的比特率下允许有 7680－4×768＝4608 bits 的最大偏差（短期缓冲）。实际上最大的偏差是 2^9×8＝4096 bits。因此也可以根据这个规则来计算中间的比特率的延时和缓冲长度。立体声比特流中左右声道交换缓冲是允许而没有限制的。因为受限于缓冲大小，在 bitrate_index == 14，数据速率为 320Kbit/s 每立体声信号的情况下，main_data_begin 的值总是被置为 0。这种情况下所有的数据都分配在帧头之后。

当采样频率低于 48KHz 时应该限制缓冲，以便同样物理大小的缓冲能足够用于上面计算的 48KHz 的情况。

- **比例因子**

比例因子可根据实际的 slen1 和 slen2 来解码，而 slen1 和 slen2 则是通过解码 scalefac_compresss 得到。解码出来的值可以作为查表的索引，也可以用来直接计算每个比例因子带的因数。当解码第二个颗粒时，必须考虑 scfsi 的值。对于 scfsi 被设为 1 的带，第一个颗粒的比例因子也用到第二个颗粒上，因此它们没有传输给第二个颗粒。

用来编码比例因子的比特数被称为 part2_length，用下面的方式计算得出：

对于 block_type == 0，1，或 3（长块）：

$$part2\_length = 11 \times slen1 + 10 \times slen2$$

对于 block_type == 2（短块）且 mixed_block_flag == 0：

$$part2\_length = 18 \times slen1 + 18 \times slen2$$

对于 block_type == 2（短块）且 mixed_block_flag == 1：

$$part2\_length = 17 \times slen1 + 18 \times slen2$$

这些公式在 gr == 0 或 gr == 1 且 scfsi[ch][scfsi_band] == 0 的条件下对所有 scfsi_band 都有效。比例因子选择信息并未使用。

- **哈夫曼解码**

所有需要的信息，包括实现哈夫曼编码树的表格可以从附录 3-B，表 3-B.7 中生成。首先使用数 table_select[gr][ch][region]索引的表，来解码 big_values 数据。在区域 0，区域 1 和区域 2 中的频率线是成对哈夫曼解码的，直到 big_values 数目的频率线对被解码完毕。剩下的哈夫曼编码比特则使用根据 count1table_select[gr][ch]索引的表来解码。当所有哈夫曼编码比特被解码，或者代表 576 个频率线的量化值被解码，这两个条件无论谁先完成则解码完成。如果需要解码 576 个值并且（这之外）还有多余的哈夫曼编码比特，则认为它们是填充位而丢弃。变量 count1 可以由使用 count1table_selest 得到的解码值的四倍推算得出。

- **解量化器**

非均匀量化器使用一个指数关系（来量化）。对从哈夫曼解码器的每个输出值，“is”，需要计算“$|is|^{4/3}$”。这个可以通过查表或者直接计算来完成。

解量化和全局缩放的公式：

一个完整的公式描述了从哈夫曼解码出来的值到输入综合滤波器组之前的全部处理过程。所有需要的缩放因子都包含在这个公式中。输出数据是从解量化的采样重建而来。全局增益和子块增益的值在一个时间窗里面（在 block_type == 2 的情况下）影响所有的值。在每个比例因子带内比例因子和 preflag 更深层的调整增益。在附录 3-A.8 中可以找到相应的图示。

下面是短块（block_type == 2）的解量化等式。在缓冲中第 i 个哈夫曼解码值被称为 is_i，综合滤波器组中索引为 i 的输入被称为 xr_i：

$$xr_i = sign(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(global\_gain[gr]-210-8\times subblock\_gain[window][gr])} \times 2^{-(saclefac\_multiplier \times scalefac\_s[gr][ch][sfb][window])}$$

对长块，公式为：

$$xr_i = sign(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(global\_gain[gr]-210)} \times 2^{-(scalefac\_multiplier \times (scalefac\_l[sfb][ch][gr] + preflag[gr] \times pretab[sfb]))}$$

pretab[sfb]是表 3-B.6 预加重表中给出的值。公式中的常数 210 是用来将输出缩放到合适的范围。它是一个系统常量。综合滤波器组被假定为根据下面的公式来执行。解码器输出值的范围（PCM 采样）在-1.0 和+1.0 之间。

- **重排序**

如果使用了短块（block_type == 2），解码数据需要根据 huffmancodebits( )定义中所描述的顺序进行重排，以将频率线分离到不同的窗口。

- **立体声处理**

在解量化之后，重建的值将在送至综合滤波器组之前进行 MS 或强度立体声模式处理。在 MS 立体声模式，一个颗粒的两个声道必须有相同的 block_type（块类型）。

MS 立体声模式：

这个模式的选择（帧头中：mode_extension）允许从“独立立体声”到MS立体声的切换。MS立体声解码出的比例因子上界是从差分（右）声道的“zero_part”中得来的。在这个界限之上可以使用强度立体声，如果帧头中使能了该模式。差分声道的“zero_part”是频谱中从“bigvalues×2 + count1×4”（见2.4.2.7节）到奈奎斯特采样频率的部分。

MS 矩阵：

在 MS 立体声模式中传输归一化的 middle/side 声道值 M_i/S_i，而不是左右声道的值 L_i/R_i。可以通过下面的公式重建 L_i/R_i：

$$L_i = \frac{M_i + S_i}{\sqrt{2}} \text{ 而 } R_i = \frac{M_i - S_i}{\sqrt{2}}$$

值 M_i 在左声道中传输，S_i 在右声道中传输。

如果出现窗切换的情况，那么 M 和 S 声道必须同步切换。

强度立体声模式：

这种模式选择（帧头中：mode_extension）允许从“常规立体声”到强度立体声的切换。强度立体声中解码出的比例因子下界是从右声道的“zero_part”得来。在这个界限之上通过用右声道的比例因子决定强度立体声位置的方式来使用强度立体声。在一个比例因子带中强度立体声的位置为 7 表示这个比例因子带不和强度立体声一样解码。

```
Scalefactor bands:
|   |   |   |   |   |   |   |   |   |   |   |   |   |. . .  |   |
|<--- nonzero_part 频谱(左声道) --->|<----- zero_part 频谱 ----->|
|<----- m/s 或 l/r 立体声编码部分----->|<---- 强度立体声编码部分---->|
```

对于每个以强度立体声编码的比例因子带（sb），执行下面的步骤：

1) 从右声道的比例因子中读出强度立体声位置 is_pos_{sb}；

2) 如果 $is\_pos_{sb} == 7$，不执行下面的步骤（非法的 is_pos_{sb} 值）；

3) $is\_ratio = \tan(is\_pos_{sb} \times \frac{\pi}{12})$；

4) $L_i := L_i \times \frac{is\_ratio}{1 + isratio}$，对所有在当前比例因子带 sb 内的 i；

5) $R_i := L_i \times \frac{1}{1 + is\_ratio}$，对所有在当前比例因子带 sb 内的 i；

- **综合滤波器组**

附录 3-A，图 3-A.4 表示了包含综合滤波器组在内的模块框图。频率线根据“抗混叠”方案进行预处理（见附录 3-A，图 3-A.5 的模块方案图和附录 3-B，表 3-B.9 的系数），然后送进 IMDCT 矩阵，每 18 个值送入一个转换模块。输出的前半部分被被添加和存储起来，与上一个块的数据进行交迭。这些值作为新输出值并且是多相滤波器组的输入值。输出值的另外一半存储起来以备和下一个颗粒的数据进行交迭。将多相滤波器组的每第二子带的每第二个值乘以-1 以修正多相滤波器的频率反转。

抗混叠：

对于常规块类型颗粒（block_type == 0），到综合滤波器组的输入需要在 IMDCT 处理之前进行抗混叠处理。下面的伪代码描述了抗混叠处理的计算：

```
for (sb=1; sb<32; sb++)
    for (i=0; i<8; i++) {
        xar[18*sb-1-i] = xr[18*sb-1-i]Cs[i] - xr[18*sb+i]Ca[i]
        xar[18*sb+i] = xr[18*sb+i]Cs[i] + xr[18*sb-1-i]Ca[i]
    }
```

数组 xar[]和 xr[]的索引将一个颗粒内的频率线分类，按从低到高的顺序排列，索引为 0 表示最低的频率线，索引为 575 则是最高频率线的索引。系数 Cs[i]和 Cr[i]可以在表 3-B.9 和图示了抗混叠运算的图 3-A.5 和 3-A.6 中找到。

抗混叠并没有在 block_type == 2（短块）的颗粒中使用。

IMDCT：

接下来，n 是加窗的采样数（短块 n 为 12，长块 n 为 36）。在块类型为“短块”情况下，每 3 个短块各自转换。n/2 个 X_k 的值转换为 n 个 x_i 的值。IMDCT 的解析表达式如下：

$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos(\frac{\pi}{2n}(2i+1+\frac{n}{2})(2k+1)) \quad \text{for i = 0 to n - 1}$$

加窗：

根据 block_type 而选用不同外形的窗：

a）block_type = 0（normal window）

$$z_i = x_i \sin(\frac{\pi}{36}(i+\frac{1}{2})) \quad \text{for i = 0 to 35}$$

b）block_type = 1（start block）

$$z_i = \begin{cases} x_i \sin(\frac{\pi}{36}(i+\frac{1}{2})) & \text{for i = 0 to 17} \\ x_i & \text{for i = 18 to23} \\ x_i \sin(\frac{\pi}{12}(i-18+\frac{1}{2})) & \text{for i = 24 to 29} \\ 0 & \text{for i = 30 to 35} \end{cases}$$

c）block_type = 3（stop block）

$$z_i = \begin{cases} 0 & \text{for i = 0 to 5} \\ x_i \sin(\frac{\pi}{12}(i-6+\frac{1}{2})) & \text{for i = 6 to11} \\ x_i & \text{for i = 12 to 17} \\ x_i \sin(\frac{\pi}{36}(i+\frac{1}{2})) & \text{for i = 18 to 35} \end{cases}$$

d）block_type = 2（short block）

三个小块各自加窗。

$$y_i^{(j)} = x_i^{(j)} \sin(\frac{\pi}{12}(i+\frac{1}{2})) \quad \text{for i = 0 to 11, j = 0 to 2}$$

加窗的短块必须交迭和连接。

$$z_i = \begin{cases} 0 & \text{for i = 0 to 5} \\ y_{i-6}^{(1)} & \text{for i = 6 to 11} \\ y_{i-6}^{(1)} + y_{i-12}^{(2)} & \text{for i = 12 to 17} \\ y_{i-12}^{(2)} + y_{i-18}^{(3)} & \text{for i = 18 to 23} \\ y_{i-18}^{(3)} & \text{for i = 24 to 29} \\ 0 & \text{for i = 30 to 35} \end{cases}$$

交叠并加上前一个块：

块的 36 个值的前一半与前一个块的后一半交迭。当前块的后一半存储起来，用来和下一个块叠加。

$$result_i = z_i + s_i \quad \text{for i = 0 to 17}$$
$$s_i = z_{i+18} \quad \text{for i = 0 to 17}$$

多相滤波器组的频率反转补偿：

交迭的输出（需要）对 32 个多相子带的每一个都加上 18 个时间采样。如果时间采样从 0 到 17 进行标号，以 0 表示最早的时间采样；子带以 0 到 31 进行标号，以 0 表示最低的子带，那么在多相滤波器组处理之前，每个奇数子带的每个奇数时间采样需要乘以-1。