

INTERNATIONAL STANDARD

**ISO/IEC
11172-3**

First edition
1993-08-01

Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s —

Part 3: Audio

*Technologies de l'information — Codage de l'image animée et du son
associé pour les supports de stockage numérique jusqu'à environ
1,5 Mbit/s —*

Partie 3: Audio



Contents

Page

Introduction.....	v
Section 1: General.....	1
1.1 Scope.....	1
1.2 Normative references.....	1
Section 2: Technical elements.....	2
2.1 Definitions.....	2
2.2 Symbols and abbreviations.....	10
2.3 Method of describing bitstream syntax.....	12
2.4 Requirements.....	14
Annexes	
A Diagrams.....	38
B Tables.....	45
C The encoding process.....	66
D Psychoacoustic models.....	109
E Bit sensitivity to errors.....	140

© ISO/IEC 1993

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH1211 Genève 20 • Switzerland

Printed in Switzerland.

F	Error concealment.....	142
G	Joint stereo coding	143
H	List of patent holders.....	147

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11172-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Sub-Committee SC 29, *Coded representation of audio, picture, multimedia and hypermedia information*.

ISO/IEC 11172 consists of the following parts, under the general title *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s*:

- Part 1: Systems
- Part 2: Video
- Part 3: Audio
- Part 4: Compliance testing

Annexes A and B form an integral part of this part of ISO/IEC 11172. Annexes C, D, E, F, G and H are for information only.

Introduction

Note: Readers interested in an overview of MPEG Audio should read this Introduction and then proceed to annex A (Diagrams) and annex C (The encoding process) before reading the normative clauses 1 and 2.

To aid in the understanding of the specification of the stored compressed bitstream and its decoding, a sequence of encoding, storage and decoding is described.

0.1 Encoding

The encoder processes the digital audio signal and produces the compressed bitstream for storage. The encoder algorithm is not standardized, and may use various means for encoding such as estimation of the auditory masking threshold, quantization, and scaling. However, the encoder output must be such that a decoder conforming to the specifications of clause 2.4 will produce audio suitable for the intended application.

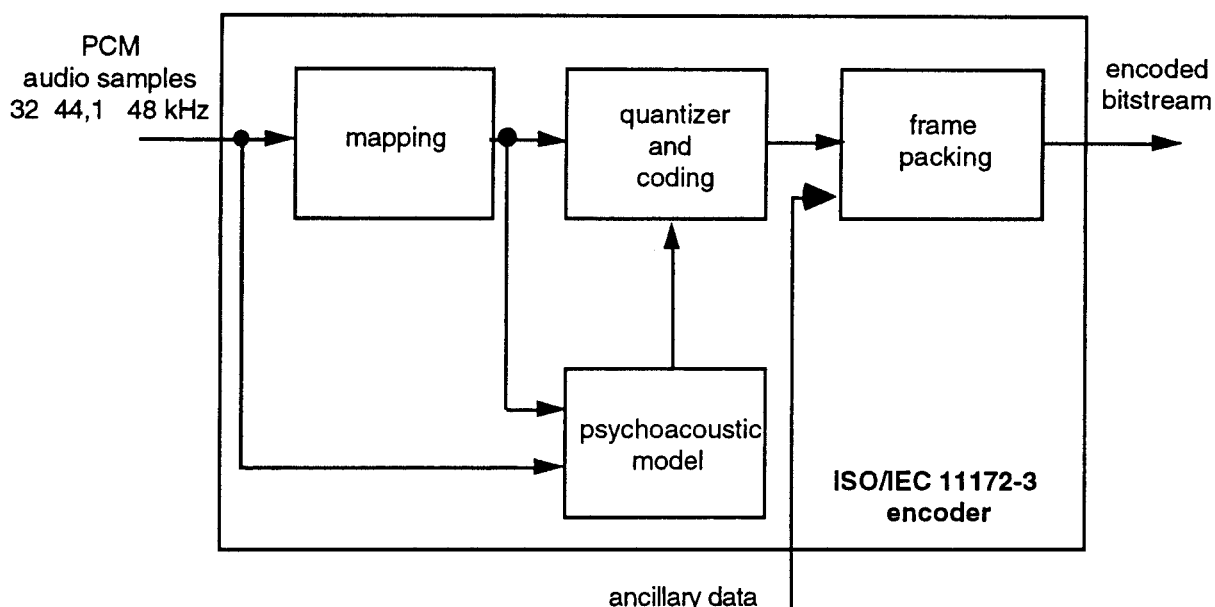


Figure 1 -- Sketch of the basic structure of an encoder

Figure 1 illustrates the basic structure of a audio encoder. Input audio samples are fed into the encoder. The mapping creates a filtered and subsampled representation of the input audio stream. The mapped samples may be called either subband samples (as in Layer I or II, see below) or transformed subband samples (as in Layer III). A psychoacoustic model creates a set of data to control the quantizer and coding. These data are different depending on the actual coder implementation. One possibility is to use an estimation of the masking threshold to do this quantizer control. The quantizer and coding block creates a set of coding symbols from the mapped input samples. Again, this block can depend on the encoding system. The block 'frame packing' assembles the actual bitstream from the output data of the other blocks, and adds other information (e.g. error correction) if necessary.

There are four different modes possible, single channel, dual channel (two independent audio signals coded within one bitstream), stereo (left and right signals of a stereo pair coded within one bitstream), and Joint Stereo (left and right signals of a stereo pair coded within one bitstream with the stereo irrelevancy and redundancy exploited).

0.2 Layers

Depending on the application, different layers of the coding system with increasing encoder complexity and performance can be used. An ISO/IEC 11172-3 Audio Layer N decoder is able to decode bitstream data which has been encoded in Layer N and all layers below N.

Layer I

This layer contains the basic mapping of the digital audio input into 32 subbands, fixed segmentation to format the data into blocks, a psychoacoustic model to determine the adaptive bit allocation, and quantization using block companding and formatting. The theoretical minimum encoding/decoding delay for Layer I is about 19 ms.

Layer II

This layer provides additional coding of bit allocation, scalefactors and samples. Different framing is used. The theoretical minimum encoding/decoding delay for Layer II is about 35 ms.

Layer III

This layer introduces increased frequency resolution based on a hybrid filterbank. It adds a different (nonuniform) quantizer, adaptive segmentation and entropy coding of the quantized values. The theoretical minimum encoding/decoding delay for Layer III is about 59 ms.

Joint Stereo coding can be added as an additional feature to any of the layers.

0.3 Storage

Various streams of encoded video, encoded audio, synchronization data, systems data and auxiliary data may be stored together on a storage medium. Editing of the audio will be easier if the edit point is constrained to coincide with an addressable point.

Access to storage may involve remote access over a communication system. Access is assumed to be controlled by a functional unit other than the audio decoder itself. This control unit accepts user commands, reads and interprets data base structure information, reads the stored information from the media, demultiplexes non-audio information and passes the stored audio bitstream to the audio decoder at the required rate.

0.4 Decoding

The decoder accepts the compressed audio bitstream in the syntax defined in 2.4.1, decodes the data elements according to 2.4.2, and uses the information to produce digital audio output according to 2.4.3.

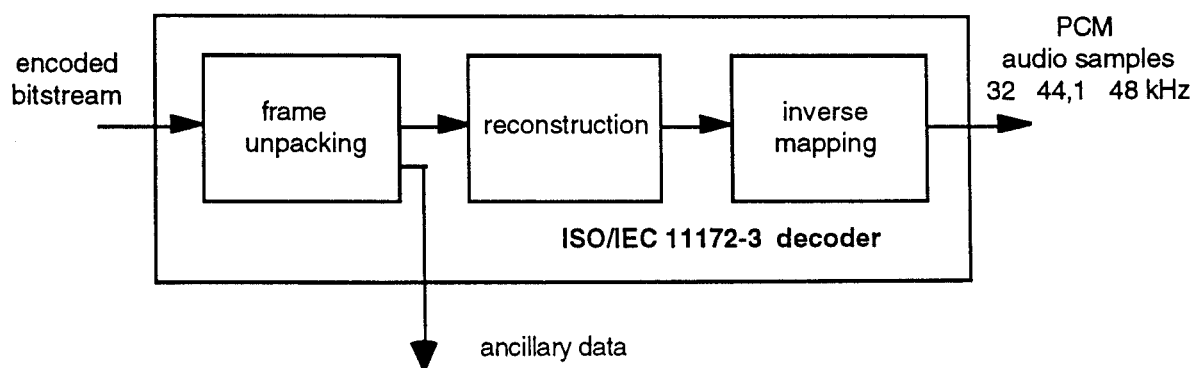


Figure 2 -- Sketch of the basic structure of a decoder

Figure 2 illustrates the basic structure of an audio decoder. Bitstream data is fed into the decoder. The bitstream unpacking and decoding block does error detection if error-check is applied in the encoder (see 2.4.2.4). The bitstream data are unpacked to recover the various pieces of information. The reconstruction block reconstructs the quantized version of the set of mapped samples. The inverse mapping transforms these mapped samples back into uniform PCM.

Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s —

Part 3: Audio

Section 1: General

1.1 Scope

This part of ISO/IEC 11172 specifies the coded representation of high quality audio for storage media and the method for decoding of high quality audio signals. The input of the encoder and the output of the decoder are compatible with existing PCM standards such as standard Compact Disc and Digital Audio Tape.

This part of the ISO/IEC 11172 is intended for application to digital storage media providing a total continuous transfer rate of about 1,5 Mbits/sec for both audio and video bitstreams, such as CD, DAT and magnetic hard disc. The storage media may either be connected directly to the decoder, or via other means such as communication lines and the ISO/IEC 11172 multiplexed stream defined in ISO/IEC 11172-1. This part of ISO/IEC 11172 is intended for sampling rates of 32 kHz, 44,1 kHz, and 48 kHz.

1.2 Normative references

The following International Standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 11172. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 11172 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 11172-1:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 1 Systems.*

ISO/IEC 11172-2:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 2: Video.*

CCIR Recommendation 601-2 *Encoding parameters of digital television for studios.*

CCIR Report 624-4 *Characteristics of systems for monochrome and colour television.*

CCIR Recommendation 648 *Recording of audio signals.*

CCIR Report 955-2 *Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of Advanced Digital System II.*

CCITT Recommendation J.17 *Pre-emphasis used on Sound-Programme Circuits.*

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

IEC publication 908:1987 *CD Digital Audio System.*

Section 2: Technical elements

2.1 Definitions

For the purposes of ISO/IEC 11172, the following definitions apply. If specific to a part, this is noted in square brackets.

2.1.1 ac coefficient [video]: Any DCT coefficient for which the frequency in one or both dimensions is non-zero.

2.1.2 access unit [system]: In the case of compressed audio an access unit is an audio access unit. In the case of compressed video an access unit is the coded representation of a picture.

2.1.3 adaptive segmentation [audio]: A subdivision of the digital representation of an audio signal in variable segments of time.

2.1.4 adaptive bit allocation [audio]: The assignment of bits to subbands in a time and frequency varying fashion according to a psychoacoustic model.

2.1.5 adaptive noise allocation [audio]: The assignment of coding noise to frequency bands in a time and frequency varying fashion according to a psychoacoustic model.

2.1.6 alias [audio]: Mirrored signal component resulting from sub-Nyquist sampling.

2.1.7 analysis filterbank [audio]: Filterbank in the encoder that transforms a broadband PCM audio signal into a set of subsampled subband samples.

2.1.8 audio access unit [audio]: For Layers I and II an audio access unit is defined as the smallest part of the encoded bitstream which can be decoded by itself, where decoded means "fully reconstructed sound". For Layer III an audio access unit is part of the bitstream that is decodable with the use of previously acquired main information.

2.1.9 audio buffer [audio]: A buffer in the system target decoder for storage of compressed audio data.

2.1.10 audio sequence [audio]: A non-interrupted series of audio frames in which the following parameters are not changed:

- ID
- Layer
- Sampling Frequency
- For Layer I and II: Bitrate index

2.1.11 backward motion vector [video]: A motion vector that is used for motion compensation from a reference picture at a later time in display order.

2.1.12 Bark [audio]: Unit of critical band rate. The Bark scale is a non-linear mapping of the frequency scale over the audio range closely corresponding with the frequency selectivity of the human ear across the band.

2.1.13 bidirectionally predictive-coded picture; B-picture [video]: A picture that is coded using motion compensated prediction from a past and/or future reference picture.

2.1.14 bitrate: The rate at which the compressed bitstream is delivered from the storage medium to the input of a decoder.

2.1.15 block companding [audio]: Normalizing of the digital representation of an audio signal within a certain time period.

2.1.16 block [video]: An 8-row by 8-column orthogonal block of pels.

2.1.17 bound [audio]: The lowest subband in which intensity stereo coding is used.

2.1.18 byte aligned: A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.

2.1.19 byte: Sequence of 8-bits.

2.1.20 channel: A digital medium that stores or transports an ISO/IEC 11172 stream.

2.1.21 channel [audio]: The left and right channels of a stereo signal

2.1.22 chrominance (component) [video]: A matrix, block or single pel representing one of the two colour difference signals related to the primary colours in the manner defined in CCIR Rec 601. The symbols used for the colour difference signals are Cr and Cb.

2.1.23 coded audio bitstream [audio]: A coded representation of an audio signal as specified in this part of ISO/IEC 11172.

2.1.24 coded video bitstream [video]: A coded representation of a series of one or more pictures as specified in ISO/IEC 11172-2.

2.1.25 coded order [video]: The order in which the pictures are stored and decoded. This order is not necessarily the same as the display order.

2.1.26 coded representation: A data element as represented in its encoded form.

2.1.27 coding parameters [video]: The set of user-definable parameters that characterize a coded video bitstream. Bitstreams are characterised by coding parameters. Decoders are characterised by the bitstreams that they are capable of decoding.

2.1.28 component [video]: A matrix, block or single pel from one of the three matrices (luminance and two chrominance) that make up a picture.

2.1.29 compression: Reduction in the number of bits used to represent an item of data.

2.1.30 constant bitrate coded video [video]: A compressed video bitstream with a constant average bitrate.

2.1.31 constant bitrate: Operation where the bitrate is constant from start to finish of the compressed bitstream.

2.1.32 constrained parameters [video]: The values of the set of coding parameters defined in 2.4.3.2 of ISO/IEC 11172-2.

2.1.33 constrained system parameter stream (CSPS) [system]: An ISO/IEC 11172 multiplexed stream for which the constraints defined in 2.4.6 of ISO/IEC 11172-1 apply.

2.1.34 CRC: Cyclic redundancy code.

2.1.35 critical band rate [audio]: Psychoacoustic function of frequency. At a given audible frequency it is proportional to the number of critical bands below that frequency. The units of the critical band rate scale are Barks.

2.1.36 critical band [audio]: Psychoacoustic measure in the spectral domain which corresponds to the frequency selectivity of the human ear. This selectivity is expressed in Bark.

2.1.37 data element: An item of data as represented before encoding and after decoding.

2.1.38 dc-coefficient [video]: The DCT coefficient for which the frequency is zero in both dimensions.

2.1.39 dc-coded picture; D-picture [video]: A picture that is coded using only information from itself. Of the DCT coefficients in the coded representation, only the dc-coefficients are present.

2.1.40 DCT coefficient: The amplitude of a specific cosine basis function.

2.1.41 decoded stream: The decoded reconstruction of a compressed bitstream.

2.1.42 decoder input buffer [video]: The first-in first-out (FIFO) buffer specified in the video buffering verifier.

2.1.43 decoder input rate [video]: The data rate specified in the video buffering verifier and encoded in the coded video bitstream.

2.1.44 decoder: An embodiment of a decoding process.

2.1.45 decoding (process): The process defined in ISO/IEC 11172 that reads an input coded bitstream and produces decoded pictures or audio samples.

2.1.46 decoding time-stamp; DTS [system]: A field that may be present in a packet header that indicates the time that an access unit is decoded in the system target decoder.

2.1.47 de-emphasis [audio]: Filtering applied to an audio signal after storage or transmission to undo a linear distortion due to emphasis.

2.1.48 dequantization [video]: The process of rescaling the quantized DCT coefficients after their representation in the bitstream has been decoded and before they are presented to the inverse DCT.

2.1.49 digital storage media; DSM: A digital storage or transmission device or system.

2.1.50 discrete cosine transform; DCT [video]: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in annex A of ISO/IEC 11172-2.

2.1.51 display order [video]: The order in which the decoded pictures should be displayed. Normally this is the same order in which they were presented at the input of the encoder.

2.1.52 dual channel mode [audio]: A mode, where two audio channels with independent programme contents (e.g. bilingual) are encoded within one bitstream. The coding process is the same as for the stereo mode.

2.1.53 editing: The process by which one or more compressed bitstreams are manipulated to produce a new compressed bitstream. Conforming edited bitstreams must meet the requirements defined in this ISO/IEC 11172.

2.1.54 elementary stream [system]: A generic term for one of the coded video, coded audio or other coded bitstreams.

2.1.55 emphasis [audio]: Filtering applied to an audio signal before storage or transmission to improve the signal-to-noise ratio at high frequencies.

2.1.56 encoder: An embodiment of an encoding process.

2.1.57 encoding (process): A process, not specified in ISO/IEC 11172, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in ISO/IEC 11172.

2.1.58 entropy coding: Variable length lossless coding of the digital representation of a signal to reduce redundancy.

2.1.59 fast forward playback [video]: The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.

2.1.60 FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).

2.1.61 filterbank [audio]: A set of band-pass filters covering the entire audio frequency range.

2.1.62 fixed segmentation [audio]: A subdivision of the digital representation of an audio signal into fixed segments of time.

2.1.63 forbidden: The term "forbidden" when used in the clauses defining the coded bitstream indicates that the value shall never be used. This is usually to avoid emulation of start codes.

2.1.64 forced updating [video]: The process by which macroblocks are intra-coded from time-to-time to ensure that mismatch errors between the inverse DCT processes in encoders and decoders cannot build up excessively.

2.1.65 forward motion vector [video]: A motion vector that is used for motion compensation from a reference picture at an earlier time in display order.

2.1.66 frame [audio]: A part of the audio signal that corresponds to audio PCM samples from an Audio Access Unit.

2.1.67 free format [audio]: Any bitrate other than the defined bitrates that is less than the maximum valid bitrate for each layer.

2.1.68 future reference picture [video]: The future reference picture is the reference picture that occurs at a later time than the current picture in display order.

2.1.69 granules [Layer II] [audio]: The set of 3 consecutive subband samples from all 32 subbands that are considered together before quantization. They correspond to 96 PCM samples.

2.1.70 granules [Layer III] [audio]: 576 frequency lines that carry their own side information.

2.1.71 group of pictures [video]: A series of one or more coded pictures intended to assist random access. The group of pictures is one of the layers in the coding syntax defined in ISO/IEC 11172-2.

2.1.72 Hann window [audio]: A time function applied sample-by-sample to a block of audio samples before Fourier transformation.

2.1.73 Huffman coding: A specific method for entropy coding.

2.1.74 hybrid filterbank [audio]: A serial combination of subband filterbank and MDCT.

2.1.75 IMDCT [audio]: Inverse Modified Discrete Cosine Transform.

2.1.76 intensity stereo [audio]: A method of exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on retaining at high frequencies only the energy envelope of the right and left channels.

2.1.77 interlace [video]: The property of conventional television pictures where alternating lines of the picture represent different instances in time.

2.1.78 intra coding [video]: Coding of a macroblock or picture that uses information only from that macroblock or picture.

2.1.79 intra-coded picture; I-picture [video]: A picture coded using information only from itself.

2.1.80 ISO/IEC 11172 (multiplexed) stream [system]: A bitstream composed of zero or more elementary streams combined in the manner defined in ISO/IEC 11172-1.

2.1.81 joint stereo coding [audio]: Any method that exploits stereophonic irrelevance or stereophonic redundancy.

2.1.82 joint stereo mode [audio]: A mode of the audio coding algorithm using joint stereo coding.

2.1.83 layer [audio]: One of the levels in the coding hierarchy of the audio system defined in this part of ISO/IEC 11172.

2.1.84 layer [video and systems]: One of the levels in the data hierarchy of the video and system specifications defined in ISO/IEC 11172-1 and ISO/IEC 11172-2.

2.1.85 luminance (component) [video]: A matrix, block or single pel representing a monochrome representation of the signal and related to the primary colours in the manner defined in CCIR Rec 601. The symbol used for luminance is Y.

2.1.86 macroblock [video]: The four 8 by 8 blocks of luminance data and the two corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the pel data and sometimes to the coded representation of the pel values and other data elements defined in the macroblock layer of the syntax defined in ISO/IEC 11172-2. The usage is clear from the context.

2.1.87 mapping [audio]: Conversion of an audio signal from time to frequency domain by subband filtering and/or by MDCT.

2.1.88 masking [audio]: A property of the human auditory system by which an audio signal cannot be perceived in the presence of another audio signal.

2.1.89 masking threshold [audio]: A function in frequency and time below which an audio signal cannot be perceived by the human auditory system.

2.1.90 MDCT [audio]: Modified Discrete Cosine Transform.

2.1.91 motion compensation [video]: The use of motion vectors to improve the efficiency of the prediction of pel values. The prediction uses motion vectors to provide offsets into the past and/or future reference pictures containing previously decoded pel values that are used to form the prediction error signal.

2.1.92 motion estimation [video]: The process of estimating motion vectors during the encoding process.

2.1.93 motion vector [video]: A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture to the coordinates in a reference picture.

2.1.94 MS stereo [audio]: A method of exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on coding the sum and difference signal instead of the left and right channels.

2.1.95 non-intra coding [video]: Coding of a macroblock or picture that uses information both from itself and from macroblocks and pictures occurring at other times.

2.1.96 non-tonal component [audio]: A noise-like component of an audio signal.

2.1.97 Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.

2.1.98 pack [system]: A pack consists of a pack header followed by one or more packets. It is a layer in the system coding syntax described in ISO/IEC 11172-1.

2.1.99 packet data [system]: Contiguous bytes of data from an elementary stream present in a packet.

2.1.100 packet header [system]: The data structure used to convey information about the elementary stream data contained in the packet data.

2.1.101 packet [system]: A packet consists of a header followed by a number of contiguous bytes from an elementary data stream. It is a layer in the system coding syntax described in ISO/IEC 11172-1.

2.1.102 padding [audio]: A method to adjust the average length in time of an audio frame to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.

2.1.103 past reference picture [video]: The past reference picture is the reference picture that occurs at an earlier time than the current picture in display order.

2.1.104 pel aspect ratio [video]: The ratio of the nominal vertical height of pel on the display to its nominal horizontal width.

2.1.105 pel [video]: Picture element.

2.1.106 picture period [video]: The reciprocal of the picture rate.

2.1.107 picture rate [video]: The nominal rate at which pictures should be output from the decoding process.

2.1.108 picture [video]: Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. The Picture layer is one of the layers in the coding syntax defined in ISO/IEC 11172-2. Note that the term "picture" is always used in ISO/IEC 11172 in preference to the terms field or frame.

2.1.109 polyphase filterbank [audio]: A set of equal bandwidth filters with special phase interrelationships, allowing for an efficient implementation of the filterbank.

2.1.110 prediction [video]: The use of a predictor to provide an estimate of the pel value or data element currently being decoded.

2.1.111 predictive-coded picture; P-picture [video]: A picture that is coded using motion compensated prediction from the past reference picture.

2.1.112 prediction error [video]: The difference between the actual value of a pel or data element and its predictor.

2.1.113 predictor [video]: A linear combination of previously decoded pel values or data elements.

2.1.114 presentation time-stamp; PTS [system]: A field that may be present in a packet header that indicates the time that a presentation unit is presented in the system target decoder.

2.1.115 presentation unit; PU [system]: A decoded audio access unit or a decoded picture.

2.1.116 psychoacoustic model [audio]: A mathematical model of the masking behaviour of the human auditory system.

2.1.117 quantization matrix [video]: A set of sixty-four 8-bit values used by the dequantizer.

2.1.118 quantized DCT coefficients [video]: DCT coefficients before dequantization. A variable length coded representation of quantized DCT coefficients is stored as part of the compressed video bitstream.

2.1.119 quantizer scalefactor [video]: A data element represented in the bitstream and used by the decoding process to scale the dequantization.

2.1.120 random access: The process of beginning to read and decode the coded bitstream at an arbitrary point.

2.1.121 reference picture [video]: Reference pictures are the nearest adjacent I- or P-pictures to the current picture in display order.

2.1.122 reorder buffer [video]: A buffer in the system target decoder for storage of a reconstructed I-picture or a reconstructed P-picture.

2.1.123 requantization [audio]: Decoding of coded subband samples in order to recover the original quantized values.

2.1.124 reserved: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/IEC defined extensions.

2.1.125 reverse playback [video]: The process of displaying the picture sequence in the reverse of display order.

2.1.126 scalefactor band [audio]: A set of frequency lines in Layer III which are scaled by one scalefactor.

2.1.127 scalefactor index [audio]: A numerical code for a scalefactor.

2.1.128 scalefactor [audio]: Factor by which a set of values is scaled before quantization.

2.1.129 sequence header [video]: A block of data in the coded bitstream containing the coded representation of a number of data elements.

2.1.130 side information: Information in the bitstream necessary for controlling the decoder.

2.1.131 skipped macroblock [video]: A macroblock for which no data are stored.

2.1.132 slice [video]: A series of macroblocks. It is one of the layers of the coding syntax defined in ISO/IEC 11172-2.

2.1.133 slot [audio]: A slot is an elementary part in the bitstream. In Layer I a slot equals four bytes, in Layers II and III one byte.

2.1.134 source stream: A single non-multiplexed stream of samples before compression coding.

2.1.135 spreading function [audio]: A function that describes the frequency spread of masking.

2.1.136 start codes [system and video]: 32-bit codes embedded in that coded bitstream that are unique. They are used for several purposes including identifying some of the layers in the coding syntax.

2.1.137 STD input buffer [system]: A first-in first-out buffer at the input of the system target decoder for storage of compressed data from elementary streams before decoding.

2.1.138 stereo mode [audio]: Mode, where two audio channels which form a stereo pair (left and right) are encoded within one bitstream. The coding process is the same as for the dual channel mode.

2.1.139 stuffing (bits); stuffing (bytes) : Code-words that may be inserted into the compressed bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream.

2.1.140 subband [audio]: Subdivision of the audio frequency band.

2.1.141 subband filterbank [audio]: A set of band filters covering the entire audio frequency range. In this part of ISO/IEC 11172 the subband filterbank is a polyphase filterbank.

2.1.142 subband samples [audio]: The subband filterbank within the audio encoder creates a filtered and subsampled representation of the input audio stream. The filtered samples are called subband samples.

From 384 time-consecutive input audio samples, 12 time-consecutive subband samples are generated within each of the 32 subbands.

2.1.143 syncword [audio]: A 12-bit code embedded in the audio bitstream that identifies the start of a frame.

2.1.144 synthesis filterbank [audio]: Filterbank in the decoder that reconstructs a PCM audio signal from subband samples.

2.1.145 system header [system]: The system header is a data structure defined in ISO/IEC 11172-1 that carries information summarising the system characteristics of the ISO/IEC 11172 multiplexed stream.

2.1.146 system target decoder; STD [system]: A hypothetical reference model of a decoding process used to describe the semantics of an ISO/IEC 11172 multiplexed bitstream.

2.1.147 time-stamp [system]: A term that indicates the time of an event.

2.1.148 triplet [audio]: A set of 3 consecutive subband samples from one subband. A triplet from each of the 32 subbands forms a granule.

2.1.149 tonal component [audio]: A sinusoid-like component of an audio signal.

2.1.150 variable bitrate: Operation where the bitrate varies with time during the decoding of a compressed bitstream.

2.1.151 variable length coding; VLC: A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.

2.1.152 video buffering verifier; VBV [video]: A hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an encoder or editing process may produce.

2.1.153 video sequence [video]: A series of one or more groups of pictures. It is one of the layers of the coding syntax defined in ISO/IEC 11172-2.

2.1.154 zig-zag scanning order [video]: A specific sequential ordering of the DCT coefficients from (approximately) the lowest spatial frequency to the highest.

2.2 Symbols and abbreviations

The mathematical operators used to describe this International Standard are similar to those used in the C programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are defined assuming twos-complement representation of integers. Numbering and counting loops generally begin from zero.

2.2.1 Arithmetic operators

+	Addition.
-	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
--	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, $7/4$ and $-7/-4$ are truncated to 1 and $-7/4$ and $7/-4$ are truncated to -1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example $3//2$ is rounded to 2, and $-3//2$ is rounded to -2.
DIV	Integer division with truncation of the result towards $-\infty$.
	Absolute value. $ x = x$ when $x > 0$ $ x = 0$ when $x == 0$ $ x = -x$ when $x < 0$
%	Modulus operator. Defined only for positive numbers.
Sign()	$\text{Sign}(x) = \begin{matrix} 1 & x > 0 \\ 0 & x == 0 \\ -1 & x < 0 \end{matrix}$
NINT ()	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.
√	Square root.
log ₁₀	Logarithm to base ten.
log _e	Logarithm to base e.
log ₂	Logarithm to base 2.

2.2.2 Logical operators

	Logical OR.
&&	Logical AND.

! Logical NOT

2.2.3 Relational operators

> Greater than.

>= Greater than or equal to.

< Less than.

<= Less than or equal to.

= Equal to.

!= Not equal to.

max [,...] the maximum value in the argument list.

min [,...] the minimum value in the argument list.

2.2.4 Bitwise operators

A twos complement number representation is assumed where the bitwise operators are used.

& AND

| OR

>> Shift right with sign extension.

<< Shift left with zero fill.

2.2.5 Assignment

= Assignment operator.

2.2.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in ISO/IEC 11172. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.
ch	Channel. If ch has the value 0, the left channel of a stereo signal or the first of two independent signals is indicated. (Audio)
nch	Number of channels; equal to 1 for single_channel mode, 2 in other modes. (Audio)
gr	Granule of 3 * 32 subband samples in audio Layer II, 18 * 32 sub-band samples in audio Layer III. (Audio)
main_data	The main_data portion of the bitstream contains the scalefactors, Huffman encoded data, and ancillary information. (Audio)
main_data_beg	The location in the bitstream of the beginning of the main_data for the frame. The location is equal to the ending location of the previous frame's main_data plus one bit. It is calculated from the main_data_end value of the previous frame. (Audio)
part2_length	The number of main_data bits used for scalefactors. (Audio)

rpchof	Remainder polynomial coefficients, highest order first. (Audio)
sb	Subband. (Audio)
sblimit	The number of the lowest sub-band for which no bits are allocated. (Audio)
scfsi	Scalefactor selection information. (Audio)
switch_point_l	Number of scalefactor band (long block scalefactor band) from which point on window switching is used. (Audio)
switch_point_s	Number of scalefactor band (short block scalefactor band) from which point on window switching is used. (Audio)
uimsbf	Unsigned integer, most significant bit first.
vlcblf	Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written.
window	Number of the actual time slot in case of block_type==2, $0 \leq \text{window} \leq 2$. (Audio)

The byte order of multi-byte words is most significant byte first.

2.2.7 Constants

π	3,14159265358...
e	2,71828182845...

2.3 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in 2.4.1. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in 2.4.2. The following constructs are used to express the conditions when data elements are present, and are in normal type:

Note this syntax uses the 'C'-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

while (condition) { data_element . . . }	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.
do { data_element . . . } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.
if (condition) { data_element . . . }	If the condition is true, then the first group of data elements occurs next in the data stream.
else { data_element . . . }	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (expr1; expr2; expr3) { expr1 is an expression specifying the initialization of the loop. Normally it specifies the initial state of the counter. expr2 is a condition specifying a test made before each iteration of the loop. The loop terminates when the condition is not true. expr3 is an expression that is performed at the end of each iteration of the loop, normally it increments a counter.

```

data_element
    . . .
  }
```

Note that the most common usage of this construct is as follows:

```

for ( i = 0; i < n; i++) { The group of data elements occurs n times. Conditional constructs
  data_element          within the group of data elements may depend on the value of the
    . . .                loop control variable i, which is set to zero for the first occurrence,
  }                      incremented to one for the second occurrence, and so forth.
```

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} may be omitted when only one data element follows.

data_element [] data_element [] is an array of data. The number of data elements is indicated by the context.

data_element [n] data_element [n] is the n+1th element of an array of data.

data_element [m][n] data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.

data_element [l][m][n] data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.

data_element [m..n] is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that 2.4.3 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include a means to look for start codes in order to begin decoding correctly.

Definition of bytealigned function

The function bytealigned () returns 1 if the current position is on a byte boundary, that is the next bit in the bitstream is the first bit in a byte. Otherwise it returns 0.

Definition of nextbits function

The function nextbits () permits comparison of a bit string with the next bits to be decoded in the bitstream.

Definition of next_start_code function

The next_start_code function removes any zero bit and zero byte stuffing and locates the next start code.

Syntax	No. of bits	Mnemonic
next_start_code() {		
while (!bytealigned())		
zero_bit	1	'0'
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
zero_byte	8	'00000000'
}		

This function checks whether the current position is bytealigned. If it is not, zero stuffing bits are present. After that any number of zero bytes may be present before the start-code. Therefore start-codes are always bytealigned and may be preceded by any number of zero stuffing bits.

2.4 Requirements

2.4.1 Specification of the coded audio bitstream syntax

2.4.1.1 Audio sequence

Syntax	No. of bits	Mnemonic
<pre>audio sequence() { while (nextbits()==syncword) { frame() } }</pre>		

2.4.1.2 Audio frame

Syntax	No. of bits	Mnemonic
<pre>frame() { header() error_check() audio_data() ancillary_data() }</pre>		

2.4.1.3 Header

Syntax	No. of bits	Mnemonic
<pre>header() { syncword ID layer protection_bit bitrate_index sampling_frequency padding_bit private_bit mode mode_extension copyright original/copy emphasis }</pre>	 12 1 2 1 4 2 1 1 2 2 1 1 2	 bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf bslbf

2.4.1.4 Error check

Syntax	No. of bits	Mnemonic
<pre>error_check() { if (protection_bit==0) crc_check }</pre>	16	rpchof

2.4.1.5 Audio data, Layer I

Syntax	No. of bits	Mnemonic
<pre> audio_data() { for (sb=0; sb<bound; sb++) for (ch=0; ch<nch; ch++) allocation[ch][sb] for (sb=bound; sb<32; sb++) { allocation[0][sb] 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] 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] for (sb=bound; sb<32; sb++) if (allocation[0][sb]!=0) sample[0][sb][s] } } </pre>	4 4 6 2..15 2..15	uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf

2.4.1.6 Audio data, Layer II

Syntax	No. of bits	Mnemonic
<pre> audio_data() { for (sb=0; sb<bound; sb++) for (ch=0; ch<nch; ch++) allocation[ch][sb] for (sb=bound; sb<sblimit; sb++) { allocation[0][sb] 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] 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] scalefactor[ch][sb][1] scalefactor[ch][sb][2] } if ((scfsi[ch][sb]==1) (scfsi[ch][sb]==3)) { scalefactor[ch][sb][0] scalefactor[ch][sb][2] } if (scfsi[ch][sb]==2) scalefactor[ch][sb][0] } 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] else for (s=0; s<3; s++) sample[ch][sb][3*gr+s] } for (sb=bound; sb<sblimit; sb++) if (allocation[0][sb]!=0) { if (grouping[0][sb]) samplecode[0][sb][gr] else for (s=0; s<3; s++) sample[0][sb][3*gr+s] } } } </pre>	2..4 2..4 2 6 6 6 6 6 6 5..10 3..16 5..10 3..16	uimbsbf uimbsbf bslbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf uimbsbf

2.4.1.7 Audio data, Layer III

Syntax	No. of bits	Mnemonic
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 ()		
}		

The main data bitstream is defined below. The **main_data** field in the **audio_data()** syntax contains bytes from the main data bitstream. However, because of the variable nature of Huffman coding used in Layer III, the main data for a frame does not generally follow the header and side information for that frame. The **main_data** for a frame starts at a location in the bitstream preceeding the header of the frame at a negative offset given by the value of **main_data_begin**. (See definition of **main_data_begin** and figure A.7.a).

Syntax	No. of bits	Mnemonic
<pre> 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] for (sfb=3; sfb<12; sfb++) for (window=0; window<3; window++) scalefac_s[gr][ch][sfb][window] } else { for (sfb=0; sfb<12; sfb++) for (window=0; window<3; window++) scalefac_s[gr][ch][sfb][window] } } else { if ((scfsi[ch][0]==0) (gr == 0)) for(sfb=0;sfb<6;sfb++) scalefac_l[gr][ch][sfb] if ((scfsi[ch][1]==0) (gr == 0)) for(sfb=6;sfb<11;sfb++) scalefac_l[gr][ch][sfb] if ((scfsi[ch][2]==0) (gr == 0)) for(sfb=11;sfb<16;sfb++) scalefac_l[gr][ch][sfb] if ((scfsi[ch][3]==0) (gr == 0)) for(sfb=16;sfb<21;sfb++) scalefac_l[gr][ch][sfb] } Huffmancodebits() } } for (b=0; b<no_of_ancillary_bits; b++) ancillary_bit } </pre>	0..4 0..4 0..4 0..4 0..4 0..4 0..3 0..3 1	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf bslbf

Syntax	No. of bits	Mnemonic
Huffmancodebits() {		
for (l=0; l<big_values*2; l+=2) {		
hcod[lx][ly]	0..19	bslbf
if (lx==15 && linbits>0)		
linbitsx	1..13	uimsbf
if (x != 0)		
signx	1	bslbf
if (ly==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[lv][lw][lx][ly]	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 Ancillary data

Syntax	No. of bits	Mnemonic
<pre> ancillary_data() { if ((layer == 1) (layer == 2)) for (b=0; b<no_of_ancillary_bits; b++) ancillary_bit } } </pre>	1	bslbf

2.4.2 Semantics for the audio bitstream syntax

2.4.2.1 Audio sequence general

frame -- Layer I and Layer II: Part of the bitstream that is decodable by itself. In Layer I it contains information for 384 samples and in Layer II for 1 152 samples. It starts with a syncword, and ends just before the next syncword. It consists of an integer number of slots (four bytes in Layer I, one byte in Layer II).

-- Layer III: Part of the bitstream that is decodable with the use of previously acquired main information. In Layer III it contains information for 1 152 samples. Although the distance between the start of consecutive syncwords is an integer number of slots (one byte in Layer III), the audio information belonging to one frame is generally not contained between two successive syncwords.

2.4.2.2 Audio frame

header -- Part of the bitstream containing synchronization and state information.

error_check -- Part of the bitstream containing information for error detection.

audio_data -- Part of the bitstream containing information on the audio samples.

ancillary_data -- Part of the bitstream that may be used for ancillary data.

2.4.2.3 Header

The first 32 bits (four bytes) are header information which is common to all layers.

syncword -- The bit string '1111 1111 1111'.

ID -- One bit to indicate the ID of the algorithm. Equals '1' for ISO/IEC 11172-3 audio, '0' is reserved.

Layer -- 2 bits to indicate which layer is used, according to the following.

Layer	
'11'	Layer I
'10'	Layer II
'01'	Layer III
'00'	reserved

To change the layer, a reset of the audio decoder may be required.

protection_bit -- One bit to indicate whether redundancy has been added in the audio bitstream to facilitate error detection and concealment. Equals '1' if no redundancy has been added, '0' if redundancy has been added.

bitrate_index -- Indicates the bitrate. The all zero value indicates the 'free format' condition, in which a fixed bitrate which does not need to be in the list can be used. Fixed means that a frame contains either N or N+1 slots, depending on the value of the padding bit. The bitrate_index is an index to a table, which is different for the different layers.

The bitrate_index indicates the total bitrate irrespective of the mode (stereo, joint_stereo, dual_channel, single_channel).

bitrate_index	bitrate specified (kbits/s)		
	Layer I	Layer II	Layer III
'0000'	free	free	free
'0001'	32	32	32
'0010'	64	48	40
'0011'	96	56	48
'0100'	128	64	56
'0101'	160	80	64
'0110'	192	96	80
'0111'	224	112	96
'1000'	256	128	112
'1001'	288	160	128
'1010'	320	192	160
'1011'	352	224	192
'1100'	384	256	224
'1101'	416	320	256
'1110'	448	384	320
'1111'	forbidden	forbidden	forbidden

In order to provide the smallest possible delay and complexity, the decoder is not required to support a continuously variable bitrate when in Layer I or II. Layer III supports variable bitrate by switching the bitrate_index. The switching of the bitrate_index can be used either to optimize storage requirements on DSM or to interpolate any mean data rate by switching between nearby values in the bitrate table. However, in free format, fixed bitrate is required. The decoder is also not required to support bitrates higher than 448 kbits/s, 384 kbits/s, 320 kbits/s in respect to Layer I, II and III when in free format mode.

For Layer II, not all combinations of total bitrate and mode are allowed. See the following table.

bit rate (kbits/s)	Allowed modes
free format	all modes
32	single_channel
48	single_channel
56	single_channel.
64	all modes
80	single_channel
96	all modes
112	all modes
128	all modes
160	all modes
192	all modes
224	stereo, intensity stereo, dual channel
256	stereo, intensity stereo, dual channel
320	stereo, intensity stereo, dual channel
384	stereo, intensity stereo, dual channel

sampling_frequency -- Indicates the sampling frequency, according to the following table.

sampling_frequency	frequency specified (kHz)
'00'	44,1
'01'	48
'10'	32
'11'	reserved

A reset of the audio decoder may be required to change the sampling rate.

padding_bit -- If this bit equals '1', the frame contains an additional slot to adjust the mean bitrate to the sampling frequency, otherwise this bit will be '0'. Padding is necessary with a sampling frequency of 44,1 kHz. Padding may also be required in free format.

The padding should be applied to the bitstream such that the accumulated length of the coded frames, after a certain number of audio frames does not deviate more than (+0, -1 slot) from the following computed value:

$$\text{accumulated frame length} = \sum_{\text{first frame}}^{\text{current frame}} (\text{frame_size} * \text{bitrate} / \text{sampling frequency})$$

where frame_size = 384 for Layer I,
1 152 for Layer II or III.

The following method can be used to determine whether or not to use padding:

```
for 1st audio frame:
    rest = 0;
    padding = no;

for each subsequent audio frame:
    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;
    }
    else padding = no;
```

private_bit -- Bit for private use. This bit will not be used in the future by ISO/IEC.

mode -- Indicates the mode according to the following table. In Layer I and II the joint_stereo mode is intensity_stereo. In Layer III it is intensity_stereo and/or ms_stereo.

mode	mode specified
'00'	stereo
'01'	joint_stereo (intensity_stereo and/or ms_stereo)
'10'	dual_channel
'11'	single_channel

In Layer I, in all modes except joint stereo, the value of bound equals 32. In layer II, in all modes except joint_stereo, the value of bound equals sblimit. In joint_stereo mode the bound is determined by the mode_extension.

mode_extension -- These bits are used in joint_stereo mode. In Layer I and II they indicate which subbands are in intensity_stereo. All other subbands are coded in stereo.

mode_extension	
'00'	subbands 4-31 in intensity_stereo, bound=4
'01'	subbands 8-31 in intensity_stereo, bound=8
'10'	subbands 12-31 in intensity_stereo, bound=12
'11'	subbands 16-31 in intensity_stereo, bound=16

In Layer III they indicate which type of joint stereo coding method is applied. The frequency ranges over which the intensity_stereo and ms_stereo modes are applied are implicit in the algorithm. For more information see 2.4.3.4.

mode_extension	intensity_stereo	ms_stereo
'00'	off	off
'01'	on	off
'10'	off	on
'11'	on	on

Note that the mode "stereo" is used if the mode bits specify stereo or equivalently if the mode bits specify joint stereo and the mode_extension specifies intensity_stereo "off" and ms_stereo "off".

copyright -- If this bit equals '0', there is no copyright on the ISO/IEC 11172-3 bitstream, '1' means copyright protected.

original/copy -- This bit equals '0' if the bitstream is a copy, '1' if it is an original.

emphasis -- Indicates the type of de-emphasis that shall be used.

emphasis	emphasis specified
'00'	none
'01'	50/15 microseconds
'10'	reserved
'11'	CCITT J.17

2.4.2.4 Error check

crc_check -- A 16 bit parity-check word is used for optional error detection within the encoded bitstream.

2.4.2.5 Audio data, Layer I

allocation[ch][sb] -- Indicates the number of bits used to code the samples in subband sb of channel ch. For subbands in intensity_stereo mode the bitstream contains only one allocation data element per subband.

allocation[ch][sb]	bits per sample
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
14	15
15	forbidden

Note: For code '0000' no samples are transferred.

scalefactor[ch][sb] -- Indicates the factor of subband sb of channel ch by which the requantized samples of subband sb in channel ch shall be multiplied. The six bits constitute an unsigned integer, index to table B.1 "Layer I, II scalefactors".

sample[ch][sb][s] -- Coded representation of the s-th sample in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the sample is valid for both channels.

2.4.2.6 Audio data, Layer II

allocation[ch][sb] -- Contains information concerning the quantizers used for the samples in subband sb in channel ch, whether the information on three consecutive samples has been grouped to one code, and on the number of bits used to code the samples. The meaning and length of this field depends on the number of the subband, the bitrate, and the sampling frequency. The bits in this field form an unsigned integer used as an index to the relevant table in table B.2 "Layer II bit allocation tables", which gives the number of levels used for quantization. For subbands in intensity_stereo mode the bitstream contains only one allocation data element per subband.

scfsi[ch][sb] -- Scalefactor selection information. This gives information on the number of scalefactors transferred for subband sb in channel ch and for which parts of the signal in this frame they are valid. The frame is divided into three equal parts of 12 subband samples each per subband.

scfsi[sb]	
'00'	three scalefactors transmitted, for parts 0,1,2 respectively.
'01'	two scalefactors transmitted, first one valid for parts 0 and 1, second one for part 2.
'10'	one scalefactor transmitted, valid for all three parts.
'11'	two scalefactors transmitted, first one valid for part 0, the second one for parts 1 and 2.

scalefactor[ch][sb][p] -- Indicates the factor by which the requantized samples of subband sb in channel ch and of part p of the frame should be multiplied. The six bits constitute an unsigned integer, index to table B.1 "Layer I, II scalefactors".

grouping[ch][sb] -- Is a function that determines, whether grouping is in effect for coding of samples in subband sb of channel ch. Grouping means, that three consecutive samples (a triplet) of the current subband sb in channel ch in the current granule gr are coded and transmitted using one common codeword and not using three separate codewords. Grouping[ch][sb] is true, if in the Bit Allocation table currently in use (see B.2) the value found under the sb (row) and the allocation[sb] (column) is either 3, 5, or 9. Otherwise it is false. For subbands in intensity_stereo mode the grouping is valid for both channels.

samplecode[ch][sb][gr] -- Coded representation of the three consecutive samples in the granule gr in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the samplecode is valid for both channels.

sample[ch][sb][s] -- Coded representation of the s-th sample in subband sb of channel ch. For subbands in intensity_stereo mode the coded representation of the sample is valid for both channels.

2.4.2.7 Audio data, Layer III

main_data_begin -- The value of main_data_begin is used to determine the location of the first bit of main data of a frame. The main_data_begin value specifies the location as a negative offset in bytes from the first byte of the audio sync word. The number of bytes belonging to the header and side information is not taken into account. For example, if main_data_begin == 0, then main data starts after the side information. Examples are given in figure A.7.a and figure A.7.b.

private_bits -- Bits for private use. These bits will not be used in the future by ISO/IEC. The number of private_bits depends on the number of channels. The number of bits allocated for private_bits is determined to equalize the total number of bits used for side-information.

scfsi[ch][scfsi_band] -- In Layer III, the scalefactor selection information works similarly to audio Layer II. The main difference is the use of the variable scfsi_band to apply scfsi to groups of scalefactors instead of single scalefactors. The application of scalefactors to granules is controlled by scfsi.

scfsi[scfsi_band]	
'0'	scalefactors are transmitted for each granule
'1'	scalefactors transmitted for granule 0 are also valid for granule 1

If short windows are switched on, i.e. `block_type==2` for one of the granules, then `scfsi` is always 0 for this frame.

scfsi_band controls the use of the scalefactor selection information for groups of scalefactors (scfsi_bands).

scfsi_band	scalefactor bands (see table 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] -- This value contains the number of main_data bits used for scalefactors and Huffman code data. Because the length of the side information is always the same, this value can be used to calculate the beginning of the main information for the next granule or the position of the ancillary information (if used). Note that single channel audio frames contain 17 bytes of side information and dual channel audio frames contain 32 bytes of side information (see 2.4.1.7 Audio Data, Layer III - syntax for `audio_data()`).

big_values[gr][ch] -- The spectral values of each granule are coded with different Huffman code tables. The full frequency range from zero to the Nyquist frequency is divided into several regions, which then are coded using different tables. Partitioning is done according to the maximum quantized values. This is done with the assumption that values at higher frequencies are expected to have lower amplitudes or do not need to be coded at all. Starting at high frequencies, the pairs of quantized values equal to zero are counted. This number is named "rzero". Then, quadruples of quantized values with absolute value not exceeding 1 (i.e. only 3 possible quantization levels) are counted. This number is named "count1". Again an even number of values remain. Finally, the number of pairs of values in the region of the spectrum which extends down to zero is named "big_values". The maximum absolute value in this range is constrained to 8191. The following figure shows the partitioning:

```

xxxxxxxxxxxxxxxx-----00000000000000000000000000000000
|                               |                               |
1          bigvalues*2          bigvalues*2+count1*4          iblen

```

The values 000 are all zero. Their number is a multiple of 2.
The values --- are -1,0 or +1. Their number is a multiple of 4.
The values xxx are not bound. Their number is a multiple of 2.
Iblen is 576.

global_gain[gr][ch] -- The quantizer step size information is transmitted in the side information variable **global_gain**. It is logarithmically quantized. For the application of **global_gain**, refer to the formula in 2.4.3.4, "Formula for requantization and all scaling".

scalefac_compress[gr][ch] -- Selects the number of bits used for the transmission of the scalefactors according to the following table:

if **block_type** is 0, 1, or 3:

slen1: length of scalefactors for the scalefactor bands 0 to 10
slen2: length of scalefactors for the scalefactor bands 11 to 20

if block type is 2 and mixed_block_flag is 0:

slen1: length of scalefactors for the scalefactor bands 0 to 5
slen2: length of scalefactors for the scalefactor bands 6 to 11

if **block_type** is 2 and **mixed_block_flag** is 1:

slen1: length of scalefactors for the scalefactor bands 0 to 7 (long window scalefactor band) and 3 to 5 (short window scalefactor band) Note: Scalefactor bands 0-7 are from the "long window

scalefactor band" table, and scalefactor bands 3 to 11 from the "short window scalefactor band" table. This combination of partitions is contiguous and spans the entire frequency spectrum.
 slen2: length of scalefactors for the scalefactor bands 6 to 11

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
12	3	2
13	3	3
14	4	2
15	4	3

window_switching_flag[gr][ch] -- Signals that the block uses an other than normal (type 0) window.

If **window_switching_flag** is set, several other variables are set by default:

region0_count = 7 (in case of **block_type**==1 or **block_type**==3
 or **block_type**==2 and **mixed_block_flag**)
region0_count = 8 (in case of **block_type**==2 and not **mixed_block_flag**)
region1_count = 36 Thus all remaining values in the big_value region are contained in
 region 1.

If **window_switching_flag** is not set, then the value of **block_type** is zero.

block_type[gr][ch] -- Indicates the window type for the granule (see description of the filterbank, Layer III).

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

Block_type and **mixed_block_flag** give the information about assembling of values in the block and about length and count of the transforms (see figure A.4 for a schematic, annex C for an analytic description). If **window_switching_flag**==1, then the **mixed_block_flag** indicates whether lower frequency polyphase filter subbands are coded using normal window type. The polyphase filterbank is described in 2.4.3.

In the case of long blocks (**block_type** not equal to 2 or in the lower subbands of **block_type** 2 if the **mixed_block_flag** is set) the IMDCT generates an output of 36 values every 18 input values. The output is windowed depending on the **block_type** and the first half is overlapped with the second half of the block before. The resulting vector is the input of the synthesis part of the polyphase filterbank of one band.

In the case of short blocks (in the upper subbands of a type 2 block if the **mixed_block_flag** is set, or in all subbands of a type 2 block if **mixed_block_flag** is not set), three transforms are performed producing 12 output values each. The three vectors are windowed and overlapped each. Concatenating 6 zeros on both ends of the resulting vector gives a vector of length 36, which is processed like the output of a long transform.

mixed_block_flag[gr][ch] -- Indicates that lower frequencies are transformed with a window type that is different than that which is used at higher frequencies. If **mixed_block_flag** is zero, then all blocks are transformed as indicated by **block_type[gr][ch]**. If **mixed_block_flag** is one, then the frequency lines corresponding to the two lowest frequency polyphase subbands are transformed with normal window (**block_type**==0), while the remaining 30 subbands are transformed as **block_type[gr][ch]**.

table_select[gr][ch][region] -- Different Huffman code tables are used depending on the maximum quantized value and the local statistics of the signal. There are a total of 32 possible tables given in table B.7.

subblock_gain[gr][ch][window] -- Indicates the gain offset (quantization: factor 4) from the global gain for one subblock. Used only with block type 2 (short windows). The values of the subblock have to be divided by $4^{(\text{subblock_gain}[\text{window}])}$ in the decoder. See 2.4.3.4 - Formula for requantization and all scaling.

region0_count[gr][ch] -- A further partitioning of the spectrum is used to enhance the performance of the Huffman coder. It is a subdivision of the region which is described by **big** values. The purpose of this subdivision is to get better error robustness and better coding efficiency. Three regions are used, they are named: region 0, 1 and 2. Each region is coded using a different Huffman code table depending on the maximum quantized value and the local signal statistics.

The values **region0_count** and **region1_count** are used to indicate the boundaries of the regions. The region boundaries are aligned with the partitioning of the spectrum into scale factor bands.

The field **region0_count** contains one less than the number of scalefactor bands in region 0. In the case of short blocks, each scale factor band is counted three times, once for each short window, so that a **region0_count** value of 8 indicates that region1 begins at scalefactor band number 3.

If **block_type**==2 and **mixed_block_flag**==0, the total amount of scalefactor bands for the granule in this case is $12 \times 3 = 36$. If **block_type**==2 and **mixed_block_flag**==1, the amount of scalefactor bands is $8 + 9 \times 3 = 35$. If **block_type**!=2, the amount of scalefactor bands is 21.

region1_count[gr][ch] -- **region1_count** counts one less than the number of scalefactor bands in region 1. Again, if **block_type**==2 the scalefactor bands representing different time slots are counted separately.

preflag[gr][ch] -- This is a shortcut for additional high frequency amplification of the quantized values. If **preflag** is set, the values of a table are added to the scalefactors (see table B.6). This is equivalent to multiplication of the requantized scalefactors with table values. If **block_type**==2 (short blocks) **preflag** is never used.

scalefac_scale[gr][ch] -- The scalefactors are logarithmically quantized with a step size of 2 or $(\sqrt{2})$ depending on **scalefac_scale**. The following table indicates the scale factor multiplier used in the requantization equation for each stepsize.

scalefac_scale[gr]	scalefac_multiplier
0	0,5
1	1

count1table_select[gr][ch] -- This flag selects one of two possible Huffman code tables for the region of quadruples of quantized values with magnitude not exceeding 1.

count1table_select[gr]	
0	Table B.7 - A
1	Table B.7 - B

scalefac_l[gr][ch][sfb], **scalefac_s[gr][ch][sfb][window]**, **is_pos[sfb]** -- The scalefactors are used to colour the quantization noise. If the quantization noise is coloured with the right shape, it is masked completely. Unlike Layers I and II, the Layer III scalefactors say nothing about the local maximum of the quantized signal. In Layer III, scalefactors are used in the decoder to get division factors for groups of values. In the case of Layer III, the groups stretch over several frequency lines. These groups are called scalefactor bands and are selected to resemble critical bands as closely as possible.

The `scalefac_compress` table shows that the scalefactors 0...10 have a range of 0 to 15 (maximum length 4 bits) and the scalefactors 11...21 have a range of 0 to 7 (maximum length 3 bits).

If `intensity_stereo` is enabled (`modebit_extension`) the scalefactors of the "zero_part" of the difference (right) channel are used as `intensity_stereo` positions, `is_pos[sfb]` (see 2.4.3.4, `MS_stereo` mode). `is_pos[sfb]` is the intensity stereo position for scalefactor band `sfb`.

The subdivision of the spectrum into scalefactor bands is fixed for every block length and sampling frequency and stored in tables in the coder and decoder (see table B.8). The scale factor for frequency lines above the highest line in the tables is zero, which means that the actual multiplication factor is 1,0.

The scalefactors are logarithmically quantized. The quantization step is set with `scalefac_scale`.

huffmancodebits() -- Huffman encoded data.

The syntax for `huffmancodebits()` shows how quantized values are encoded. Within the `big_values` partition, pairs of quantized values with an absolute value less than 15 are directly coded using a Huffman code. The codes are selected from Huffman tables 0 through 31 in table B.7. Always pairs of values (x,y) are coded. If quantized values of magnitude greater than or equal to 15 are coded, the values are coded with a separate field following the Huffman code. If one or both values of a pair is not zero, one or two sign bits are appended to the code word.

The Huffman tables for the `big_values` partition are comprised of three parameters:

<code>hcod[x][y]</code>	is the Huffman code table entry for values x,y.
<code>hlen[x][y]</code>	is the Huffman length table entry for values x,y.
<code>linbits</code>	is the length of <code>linbitsx</code> or <code>linbitsy</code> when they are coded.

The syntax for `huffmancodebits` contains the following fields and parameters:

signv	is the sign of v (0 if positive, 1 if negative).
signw	is the sign of w (0 if positive, 1 if negative).
signx	is the sign of x (0 if positive, 1 if negative).
signy	is the sign of y (0 if positive, 1 if negative).
linbitsx	is used to encode the value of x if the magnitude of x is greater or equal to 15. This field is coded only if <code> x </code> in <code>hcod</code> is equal to 15. If <code>linbits</code> is zero, so that no bits are actually coded when <code> x ==15</code> , then the value <code>linbitsx</code> is defined to be zero.
linbitsy	is the same as <code>linbitsx</code> but for y.
is[l]	Is the quantized value for frequency line number l.

The `linbitsx` or `linbitsy` fields are only used if a value greater or equal to 15 needs to be encoded. These fields are interpreted as unsigned integers and added to 15 to obtain the encoded value. The `linbitsx` and `linbitsy` fields are never used if the selected table is one for blocks with a maximum quantized value less than 15. Note that a value of 15 can still be encoded with a Huffman table for which `linbits` is zero. In this case, the `linbitsx` or `linbitsy` fields are not actually coded, since `linbits` is zero.

Within the `count1` partition, quadruples of values with magnitude less than or equal to one are coded. Again magnitude values are coded using a Huffman code from tables A or B in table B.7. Again, for each non-zero value, a sign bit is appended after the Huffman code symbol.

The Huffman tables for the `count1` partition are comprised of the following parameters:

<code>hcod[v][w][x][y]</code>	is the Huffman code table entry for values v,w,x,y.
<code>hlen[v][w][x][y]</code>	is the Huffman length table entry for values v,w,x,y.

Huffman code table B is not really a 4-dimensional code because it is constructed from the trivial code: 0 is coded with a 1, and 1 is coded with a 0.

Quantized values above the `count1` partition are zero, so they are not encoded.

For clarity, the parameter "count1" is used in this document to indicate the number of Huffman codes in the `count1` region. However, unlike the `bigvalues` partition, the number of values in the `count1` partition is not

explicitly coded by a field in the syntax. The end of the count1 partition is known only when all bits for the granule (as specified by part2_3_length), have been exhausted, and the value of count1 is known implicitly after decoding the count1 region.

The order of the Huffman data depends on the block_type of the granule. If block_type is 0, 1 or 3 the Huffman encoded data is ordered in terms of increasing frequency.

If block_type==2 (short blocks) the Huffman encoded data is ordered in the same order as the scalefactor values for that granule. The Huffman encoded data is given for successive scalefactor bands, beginning with scalefactor band 0. Within each scalefactor band, the data is given for successive time windows, beginning with window 0 and ending with window 2. Within each window, the quantized values are then arranged in order of increasing frequency.

2.4.2.8 Ancillary data

Ancillary_bit -- User definable.

The number of ancillary bits (no_of_ancillary_bits) equals the available number of bits in an audio frame minus the number of bits actually used for header, error check and audio data. In Layer I and II the no_of_ancillary_bits corresponds to the distance between the end of the audio data and the beginning of the next header. In Layer III the no_of_ancillary_bits corresponds to the distance between the end of the Huffman_code_bits and the location in the bitstream where the next frame's main_data_begin pointer points to.

2.4.3 The audio decoding process

2.4.3.1 General

The first action is synchronization of the decoder to the incoming bitstream. Just after startup this may be done by searching in the bitstream for the 12 bit syncword. In some applications the ID, layer, and protection status are already known to the decoder, and thus the first 16 bits of the header should be regarded as a 16 bit syncword, thereby allowing a more reliable synchronization. The position of consecutive syncwords can be calculated from the information provided by the seven bits after the protection_bit: the bitstream is subdivided in slots. The distance between the start of two consecutive syncwords is equal to "N" or "N+1" slots. The value of "N" depends on the layer.

For Layer I the following equation is valid:

$$N = 12 * \frac{\text{bitrate}}{\text{sampling_frequency}}.$$

For Layers II and III the equation becomes:

$$N = 144 * \frac{\text{bitrate}}{\text{sampling_frequency}}.$$

If this calculation does not give an integer number the result is truncated and 'padding' is required. In this case the number of slots in a frame will vary between N and N+1. The padding bit is set to '0' if the number of slots equals N, and to '1' otherwise. This knowledge of the position of consecutive syncwords greatly facilitates synchronization.

If the bitrate index equals '0000', the exact bitrate is not indicated. N can be determined from the distance between consecutive syncwords and the value of the padding bit.

The mode bits in the bitstream shall be read and if their value is '01' the mode_extension bits shall also be read. The mode_extension bits set the 'bound' as shown in 2.4.2.3 and thus indicate which subbands are coded in joint_stereo mode.

If the protection bit in the header equals '0', a CRC-check word has been inserted in the bitstream just after the header. The error detection method used is 'CRC-16' whose generator polynomial is:

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

The bits included into the CRC-check are given by table B.5.

The method is depicted in figure A.9 "CRC-check diagram". The initial state of the shift register is '1111 1111 1111 1111'. Then all the bits included into the CRC-check are input to the circuit shown in figure A.9 "CRC-check diagram". After each bit is input the shift register is shifted by one bit. After the last shift operation, the outputs $b_{15}...b_0$ constitute a word to be compared with the CRC-check word in the bitstream. If the words are not identical, a transmission error has occurred in the protected field of the bitstream. To avoid annoying distortions, application of a concealment technique, such as muting of the actual frame or repetition of the previous frame, is recommended.

2.4.3.2 Layer I

After the part of the decoding which is common to all layers (see 2.4.3.1) the bit allocation information has to be read for all subbands, and the scalefactors read for all subbands with a nonzero bit allocation. The decoder flowchart is given in figure A.1 "Layer I and II decoder flow chart".

2.4.3.2.1 Requantization of subband samples

From the bit allocation the number of bits n_b that has to be read for the samples in each subband is known. The order of the samples is given in 2.4.1.5 for each mode. After the bits for one sample have been gathered from the bitstream, the first bit has to be inverted. The resulting number can be considered as a two's

complement fractional number, where the MSB represents the value -1. The requantized value can be obtained by applying a linear formula :

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

where

s''' is the fractional number;
 s'' is the requantized value;
 nb is the number of bits allocated to samples in the subband.

Samples in subbands which are in intensity_stereo mode must be copied to both channels. The requantized value has to be rescaled. The multiplication factor can be found in the table B.1 "Layer I, II scalefactors". The rescaled value s' is calculated as :

$$s' = \text{factor} * s''$$

2.4.3.2.2 Synthesis subband filter

If a subband has no bits allocated to it, the samples in that subband are set to zero. Each time the subband samples for all 32 subbands of one channel have been calculated, they can be applied to the synthesis subband filter and 32 consecutive audio samples can be calculated. The actions in flow diagram figure A.2 "Synthesis subband filter flow chart" show the reconstruction operation. The coefficients N_{ik} for the matrixing operation are given by

$$N_{ik} = \cos \left[(16 + i) (2k+1) \frac{\pi}{64} \right] \quad 0 \leq i \leq 63, 0 \leq k \leq 31$$

The coefficients D_i for the windowing operation can be found in table B.3 "Coefficients D_i of the synthesis window". The coefficients have been derived by numerical optimization. One frame contains $12 * 32 = 384$ subband samples, which result, after filtering, in 384 audio samples.

2.4.3.3 Layer II

Layer II is a more efficient but more complex coding scheme than Layer I. The flowchart in figure A.1 "Layer I and II decoder flow chart" applies to both Layers I and II. The first step is to perform the decoding which is common to all three layers (see 2.4.3.1).

2.4.3.3.1 Bit allocation decoding

For different combinations of bitrate and sampling frequency different bit allocation tables exist (table B.2 "Layer II bit allocation tables"). Note that the bitrates given in the table headers are per channel. If the mode is not single_channel, the bitrate should be divided by two to obtain the bitrate per channel. The decoding of the bit allocation table is done in a three-step approach. The first step consists of reading 'nbal' (2,3, or 4) bits of information for one subband from the bitstream. The value of 'nbal' is given in the second column of the relevant table B.2 "Layer II bit allocation tables". These bits shall be interpreted as an unsigned integer number. The second step uses this number and the number of the subband as indices to point to a value in the table. This value represents the number of levels 'nlevels' used to quantize the samples in the subband. As a third step, using table B.4 "Layer II classes of quantization", the number of bits used to code the quantized samples, the requantization coefficients, and whether the codes for three consecutive subband samples have been grouped to one code can be determined. It can be seen from the bit allocation tables that some of the highest subbands will never have bits allocated. The number of the lowest subband that will not have bits allocated to it is assigned to the identifier 'sblimit'.

2.4.3.3.2 Scalefactor selection information decoding

The 36 samples in one subband within a frame are divided in three equal parts of 12 subband samples. Each part can have its own scalefactor. The number of scalefactors that has to be read from the bitstream depends on $scfsi[sb]$. The scalefactor selection information $scfsi[sb]$ is read from the bitstream for the subbands that have a nonzero bit allocation. If $scfsi[sb]$ equals '00' three scalefactors are transmitted, for parts 0,1,2 respectively. If $scfsi[sb]$ equals '01' two scalefactors are transmitted, the first one valid for parts 0 and 1, the

second one for part 2. If $scfsi[sb]$ equals '10' one scalefactor is transmitted, valid for all three parts. If $scfsi[sb]$ equals '11' two scalefactors are transmitted, the first one valid for part 0, the second one for parts 1 and 2.

2.4.3.3.3 Scalefactor decoding

For every subband with a nonzero bit allocation the coded scalefactors for that subband are read from the bitstream. The number of coded scalefactors and the part of the subband samples they refer to is defined by $scfsi[sb]$. The 6 bits of a coded scalefactor should be interpreted as an unsigned integer index to table B.1 "Layer I, II scalefactors". This table gives the scalefactor by which the relevant subband samples should be multiplied after requantization.

2.4.3.3.4 Requantization of subband samples

Next the coded samples are read. As can be seen from 2.4.1.6, the coded samples appear as triplets, the code contains three consecutive samples at a time. From table B.4 "Layer II classes of quantization" it is known how many bits have to be read for one triplet from the bitstream for each subband. Also from table B.4 "Layer II classes of quantization", it is known whether this code consists of three consecutive separable codes for each sample or of one combined code for the three samples (grouping). In the last case degrouping must be performed. The combined code has to be regarded as an unsigned integer, called 'c'. The following algorithm will supply the three separate codes $s[0]$, $s[1]$, $s[2]$.

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

where $nlevels$ is the number of steps as shown in table B.2 "Layer II bit allocation table".

The first bit of each of the three codes has to be inverted, and the resulting numbers should be regarded as two's complement fractional numbers, where the MSB represents the value -1. The requantized values can be obtained by applying a linear formula :

$$s'' = C * (s''' + D)$$

where

s''' is the fractional number;
 s'' is the requantized value.

The values of the constants C and D are given in table B.4 "Layer II classes of quantization". The requantized values have to be rescaled. The multiplication factors can be found in the table B.1 "Layer I, II scalefactors", as described above. The rescaled value s' is calculated as :

$$s' = \text{factor} * s''$$

2.4.3.3.5 Synthesis subband filter

If a subband has no bits allocated to it, the samples in that subband are set to zero. Each time the subband samples for all 32 subbands of one channel have been calculated, they can be applied to the synthesis subband filter and 32 consecutive audio samples can be calculated. For that purpose, the actions in the flow diagram of figure A.2 "Synthesis subband filter flow chart" have to be carried out. The coefficients N_{ik} for the matrixing operation are given by

$$N_{ik} = \cos \left[(16 + i) (2k+1) \frac{\pi}{64} \right] \quad 0 \leq i \leq 63, 0 \leq k \leq 31$$

The coefficients D_i for the windowing operation can be found in table B.3 "Coefficients D_i of the synthesis window". One frame contains $36 * 32 = 1152$ subband samples, which result after filtering in 1152 audio samples.

2.4.3.4 Layer III

Additional frequency resolution is provided by the use of an hybrid filterbank. Every band is split into 18 frequency lines by use of an MDCT. The window length of the MDCT is 36. Adaptive window switching is used to control time artifacts (pre-echoes), see the description in annex C. The frequency above which shorter blocks (better time resolution) are used can be selected. Parts of the signal below a frequency depending on "mixed_block_flag" are coded with better frequency resolution, parts of the signal above are coded with better time resolution.

The frequency components are quantized using a nonuniform quantizer and coded using a Huffman encoder. The Huffman coder uses one of 18 different tables (see table B.7). A buffer is used to help enhance the coding efficiency of the Huffman coder and to help in the case of pre-echo conditions (see the description in annex C). The size of the input buffer is the size of one frame at the bitrate of 160 kbits/s per channel for Layer III. The short term buffer technique used is called "bit reservoir" because it uses short-term variable bitrate with a maximum integral offset from the mean bitrate.

Each frame holds the data from 2 granules. The audio data in a frame is allocated in the following way:

- main_data_begin pointer
- side info for both granules (scfsi)
- side info granule 1
- side info granule 2

The header and this part of the audio data constitute the side information stream.

- scalefactors and Huffman code data granule 1
- scalefactors and Huffman code data granule 2
- ancillary data

These data constitute the main data stream. The main_data_begin pointer specifies a negative offset from the position of the first byte of the header.

2.4.3.4.1 Decoding

The first action is the synchronization of the decoder to the incoming bitstream. This is done as in the other layers. The header information (first 32 bits including syncword) is read in just as in the other layers. The information about sampling frequency is used to select the scalefactor_band table (see annex B.8).

2.4.3.4.2 Side information

The side information must be extracted from the bitstream and stored for use during the decoding of the associated frame. The table select information is used to select the Huffman decoder table and the number of ESC-bits (linbits), according to table B.7.

2.4.3.4.3 Start of main_data

The main_data (scalefactors, Huffman coded data and ancillary information) are not necessarily located adjacent to the side information. This is described in figure A.7.a and figure A.7.b. The beginning of the main data part is located by using the main_data_begin pointer of the current frame. The allocation of the main data is done in a way that all main data are resident in the input buffer when the header of the next frame is arriving in the input buffer. The decoder has to skip Header and side information when decoding the main data. It knows their positions from the bitrate_index and padding_bit. The length of the Header is always 4 bytes, the length of the side information is 17 bytes in mode single_channel and 32 bytes in the other modes. Main data can span more than one block of header and side information (see figure A.7.b).

2.4.3.4.4 Buffer considerations

The following rule can be used to calculate the maximum number of bits used for one granule:

The buffer length is 7 680 bits. This value is used as the maximum buffer at every bitrate. At the highest possible bitrate of Layer III (320 kbits/s per stereo signal) and sampling frequency 48 kHz the mean frame length is $(320\,000/48\,000) \times 1\,152 = 7\,680$ bits. Therefore the frames must be of constant length at this bitrate and sampling frequency. At 64 kbits/s (128 kbits/s stereo) the mean granule length is

$(64\,000/48\,000) \cdot 576 = 768$ bit at 48 kHz sampling frequency. This means that there is a maximum deviation (short time buffer) of $7\,680 - 4 \cdot 768 = 4\,608$ bits is allowed at 64 kbits/s. The actual deviation is equal to the number of bytes denoted by the `main_data_begin` offset pointer. The actual maximum deviation is $2^9 \cdot 8 \text{ bit} = 4\,096$ bits. For intermediate bitrates the delay and buffer length can be calculated accordingly. The exchange of buffer between the left and right channel in a stereo bitstream is allowed without restrictions. Because of the constraint on the buffer size `main_data_begin` is always set to 0 in the case of `bitrate_index == 14`, i.e. data rate 320 kbits/s per stereo signal. In this case all data are allocated between adjacent header words.

At sampling frequencies lower than 48 kHz the buffer should be constrained such that the same physical buffer size is sufficient as the one calculated for the 48 kHz case above.

2.4.3.4.5 Scalefactors

The scalefactors are decoded according to the `slen1` and `slen2` which themselves are determined from the values of `scalefac_compress`. The decoded values can be used as entries into a table or used to calculate the factors for each scalefactor band directly. When decoding the second granule, the `scfsi` has to be considered. For the bands in which the corresponding `scfsi` is set to 1, the scalefactors of the first granule are also used for the second granule, therefore they are not transmitted for the second granule.

The number of bits used to encode scalefactors is called `part2_length`, and is calculated as follows.

For `block_type == 0, 1, or 3` (long blocks):

$$\text{part2_length} = 11 \cdot \text{slen1} + 10 \cdot \text{slen2}.$$

For `block_type == 2` (short blocks) and `mixed_block_flag == 0`:

$$\text{part2_length} = 18 \cdot \text{slen1} + 18 \cdot \text{slen2}.$$

For `block_type == 2` (short blocks) and `mixed_block_flag == 1`:

$$\text{part2_length} = 17 \cdot \text{slen1} + 18 \cdot \text{slen2}.$$

These formulas are valid if `gr == 0` or if `gr == 1` and `scfsi[ch][scfsi_band] == 0` for all `scfsi_bands`, i.e. scalefactor selection information is not used.

2.4.3.4.6 Huffman decoding

All necessary information including the table which realizes the Huffman code tree can be generated from the tables in table B.7. First the `big_values` data are decoded, using the tables with the number `table_select[gr][ch][region]`. The frequency lines in region 0, region 1 and region 2 are Huffman decoded in pairs until `big_values` number of line-pairs have been decoded. The remaining Huffman code bits are decoded using the table according to `count1table_select[gr][ch]`. Decoding is done until all Huffman code bits have been decoded or until quantized values representing 576 frequency lines have been decoded, whichever comes first. If there are more Huffman code bits than necessary to decode 576 values they are regarded as stuffing bits and discarded. The variable `count1` is implicitly derived as the number of quadruples of decoded values using `count1table_select`.

2.4.3.4.7 Requantizer

The nonuniform quantizer uses a power law. For each output value, "is", from the Huffman decoder, " $|is|^{4/3}$ " is calculated. This can be done either by table lookup or by explicit calculation.

2.4.3.4.7.1 Formula for requantization and all scaling

One complete formula describes all the processing from the Huffman decoded values to the input of the synthesis filterbank. All necessary scaling factors are contained within this formula. The output data are reconstructed from requantized samples. Global gain and subblock gain values affect all values within one time window (in the case of `block_type == 2`). Scalefactors and preflag further adjust the gain within each scalefactor band. An illustration can be found in figure A.8.

The following is the requantization equation for short windows. The Huffman decoded value at buffer index i is called is_i , the input to the synthesis filterbank at index i is called xr_i :

$$xr_i = \text{sign}(is_i) * |is_i|^{\frac{4}{3}} * 2^{\frac{1}{4}} (\text{global_gain}[gr] - 210 - 8 * \text{subblock_gain}[\text{window}][gr]) \\ * 2^{-(\text{scalefac_multiplier} * \text{scalefac_s}[gr][ch][sfb][\text{window}])}$$

For long blocks, the formula is:

$$xr_i = \text{sign}(is_i) * |is_i|^{\frac{4}{3}} * 2^{\frac{1}{4}} (\text{global_gain}[gr] - 210) \\ * 2^{-(\text{scalefac_multiplier} * (\text{scalefac_l}[sfb][ch][gr] + \text{preflag}[gr] * \text{pretab}[sfb]))}$$

Pretab[sfb] is a value given in the preemphasis table B.6. The constant 210 in this formula is needed to scale the output appropriately. It is a system constant. The synthesis filterbank is assumed to be implemented according to the formulas below. The range of the output values of the decoder (PCM samples) is between - 1,0 and + 1,0.

2.4.3.4.8 Reordering

If short blocks are used ($\text{block_type} = 2$), the rescaled data $xr[\text{scf_band}][\text{window}][\text{freq_line}]$ (as described in `huffmancodebits()` in 2.4.1.7) shall be reordered in subband order, $xr[\text{subband}[\text{window}][\text{freq_line}]$, prior to the IMDCT operation.

2.4.3.4.9 Stereo Processing

After requantization, the reconstructed values are processed for MS or intensity stereo modes or both, before going to the synthesis filterbank. In MS_stereo mode, both channels of a granule must have the same block_type .

2.4.3.4.9.1 MS_stereo mode

This mode switch (found in the header: `mode_extension`) allows switching from "independent stereo" to MS_stereo. If MS_stereo is enabled but intensity stereo is not enabled the entire spectrum is decoded in MS_stereo. If both MS_Stereo and intensity stereo are enabled, the upper bound of the scalefactor bands decoded in MS_stereo is derived from the "zero_part" of the difference (right) channel. In this case the scalefactor band in which the last non-zero (right channel) frequency line occurs is the last scalefactor band to which the MS_stereo equations apply. Above this bound intensity stereo may be applied if enabled in the header. The "zero_part" of the difference channel is the part of the spectrum from " $\text{bigvalues} * 2 + \text{count1} * 4$ " (see 2.4.2.7) to the Nyquist rate.

2.4.3.4.9.2 MS matrix

In MS stereo mode the values of the normalized middle/side channels M_i/S_i are transmitted instead of the left/right channel values L_i/R_i . Thus L_i/R_i are reconstructed using

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

The values M_i are transmitted in the left, values S_i are transmitted in the right channel.

If window switching occurs, then the M and S channels must switch synchronously.

2.4.3.4.9.3 Intensity stereo mode

This mode switch (found in the header: `mode_extension`) allows switching from 'normal stereo' to intensity stereo. In Layer III, intensity stereo is not done using a pair of scalefactor s as in Layers I and II, but by specifying the magnitude (via the scalefactors of the right channel as normal) and a stereo position $\text{is_possb}[sfb]$. $\text{is_possb}[sfb]$ is transmitted instead of scalefactors for the right channels. The stereo position is used to derive the left and right channel signals according to the formulas below. The lower bound of the scalefactor bands decoded in intensity stereo is derived from the "zero_part" of the right

channel. Above this bound decoding of intensity stereo is applied using the scalefactors of the right channel as intensity stereo positions. An intensity stereo position of 7 in one scalefactor band indicates that this scalefactor band is not decoded as intensity stereo.

Scalefactor bands :									
<--- nonzero_part of spectrum (right chan) --->					<----- zero_part of spectrum ----->				
<----- m/s or l/r stereo coded part ----->					<- intensity stereo coded part ->				

For each scalefactor band (sb) coded in intensity stereo, the following steps are executed:

- 1) the intensity stereo position is_pos_{sb} is read from the scalefactor of the right channel.
- 2) if ($is_pos_{sb} == 7$) do not perform the following steps (illegal is_pos).
- 3) $is_ratio = \tan(is_pos_{sb} * \frac{\pi}{12})$.
- 4) $L_i := L_i * \frac{is_ratio}{1 + is_ratio}$ for all indices i within the actual scalefactor band sb .
- 5) $R_i := L_i * \frac{1}{1 + is_ratio}$ for all indices i within the actual scalefactor band sb .

2.4.3.4.10 Synthesis filterbank

Figure A.4. shows a block diagram including the synthesis filterbank. The frequency lines are preprocessed by the "alias reduction" scheme (see the block diagrams in figure A.5 and in table B.9. for the coefficients) and fed into the IMDCT matrix, each 18 into one transform block. The first half of the output values are added to the stored overlap values from the last block. These values are new output values and are input values for the polyphase filterbank. The second half of the output values is stored for overlap with the next data granule. For every second subband of the polyphase filterbank every second input value is multiplied by -1 to correct for the frequency inversion of the polyphase filterbank.

2.4.3.4.10.1 Alias reduction

For long block_type granules (block_type != 2) the input to the synthesis filterbank is processed for alias reduction before processing by the IMDCT. The following pseudo code describes the alias reduction computation:

```

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]
  }

```

The indices of arrays $xar[]$ and $xr[]$ label the frequency lines in a granule, arranged in order of lowest frequency to highest frequency, with zero being the index of the lowest frequency line, and 575 being the index of the highest. The coefficients: $Cs[i]$ and $Ca[i]$ can be found in table B.9. Figures A.5 and A.6 illustrate the alias reduction computation.

Alias reduction is not applied for granules with block_type == 2 (short block).

2.4.3.4.10.2 IMDCT

In the following, n is the number of windowed samples (for short blocks n is 12, for long blocks n is 36). In the case of a block of type "short", each of the three short blocks is transformed separately. $n/2$ values X_k are transformed to n values x_i . The analytical expression of the IMDCT is:

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

2.4.3.4.10.3 Windowing

Depending on the *block_type* different shapes of windows are used.

a) *block_type*=0 (normal window)

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

b) *block_type*=1 (start block)

$$z_i = \begin{cases} x_i \sin\left(\frac{\pi}{36}\left(i + \frac{1}{2}\right)\right) & \text{for } i = 0 \text{ to } 17 \\ x_i & \text{for } i = 18 \text{ to } 23 \\ x_i \sin\left(\frac{\pi}{12}\left(i - 18 + \frac{1}{2}\right)\right) & \text{for } i = 24 \text{ to } 29 \\ 0 & \text{for } i = 30 \text{ to } 35 \end{cases}$$

c) *block_type*=3 (stop block)

$$z_i = \begin{cases} 0 & \text{for } i = 0 \text{ to } 5 \\ x_i \sin\left(\frac{\pi}{12}\left(i - 6 + \frac{1}{2}\right)\right) & \text{for } i = 6 \text{ to } 11 \\ x_i & \text{for } i = 12 \text{ to } 17 \\ x_i \sin\left(\frac{\pi}{36}\left(i + \frac{1}{2}\right)\right) & \text{for } i = 18 \text{ to } 35 \end{cases}$$

d) *block_type*=2 (short block)

Each of the three short blocks is windowed separately.

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

The windowed short blocks must be overlapped and concatenated.

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

2.4.3.4.10.4 Overlapping and adding with previous block

The first half of the block of 36 values is overlapped with the second half of the previous block. The second half of the actual block is stored to be used in the next block:

$$\begin{aligned} \text{result}_i &= z_i + s_i & \text{for } i = 0 \text{ to } 17 \\ s_i &= z_{i+18} & \text{for } i = 0 \text{ to } 17 \end{aligned}$$

2.4.3.4.10.5 Compensation for frequency inversion of polyphase filterbank

The output of the overlap add consists of 18 time samples for each of the 32 polyphase subbands. If the time samples are labeled 0 through 17, with 0 being the earliest time sample, and subbands are labeled 0 through 31, with 0 being the lowest subband, then every odd time sample of every odd subband is multiplied by -1 before processing by the polyphase filter bank.

Annex A

(normative)

Diagrams

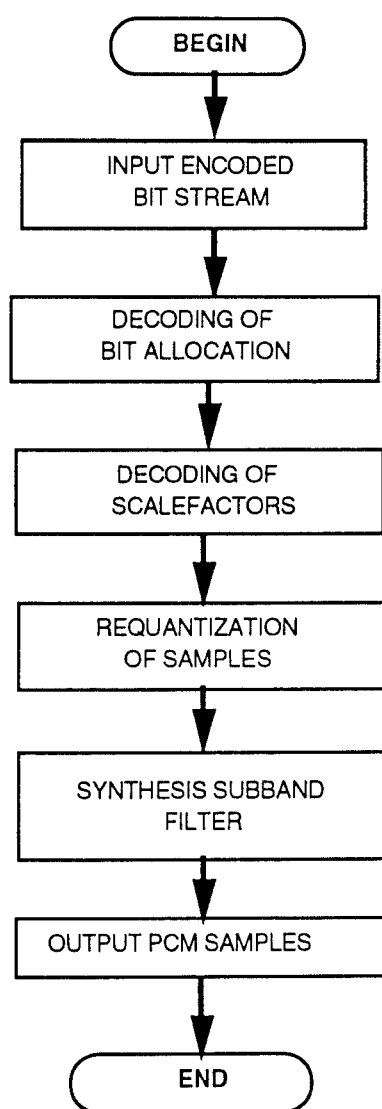


Figure A.1 -- Layer I and II decoder flow chart

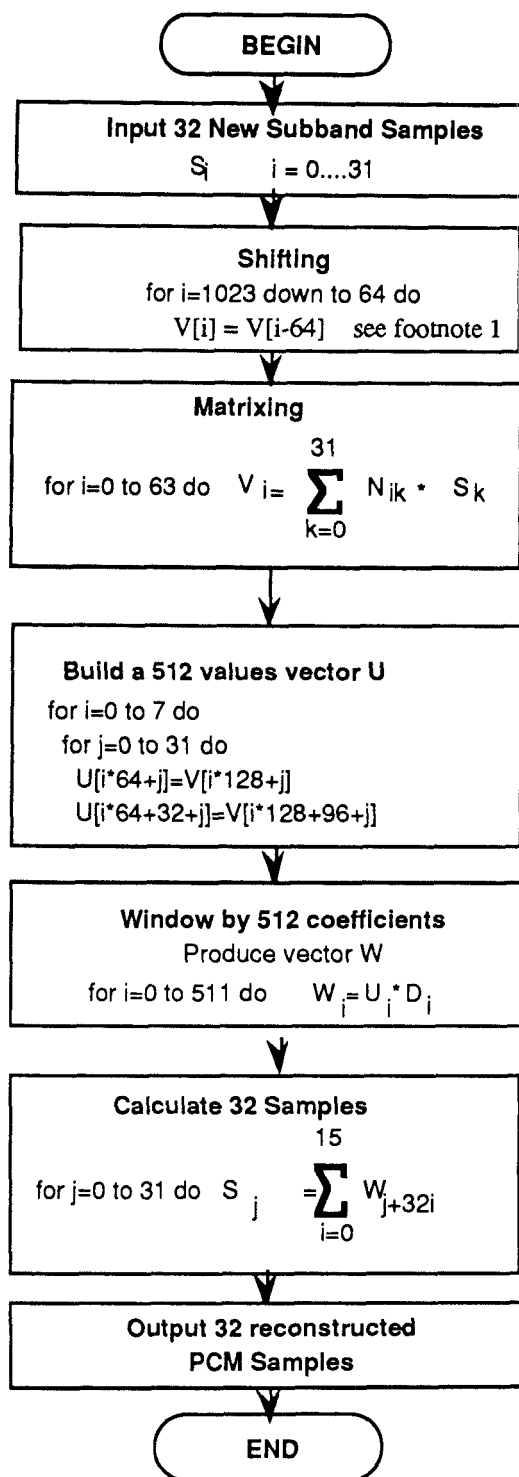


Figure A.2 -- Synthesis subband filter flow chart

¹ V to be initialized with zeroes during startup.

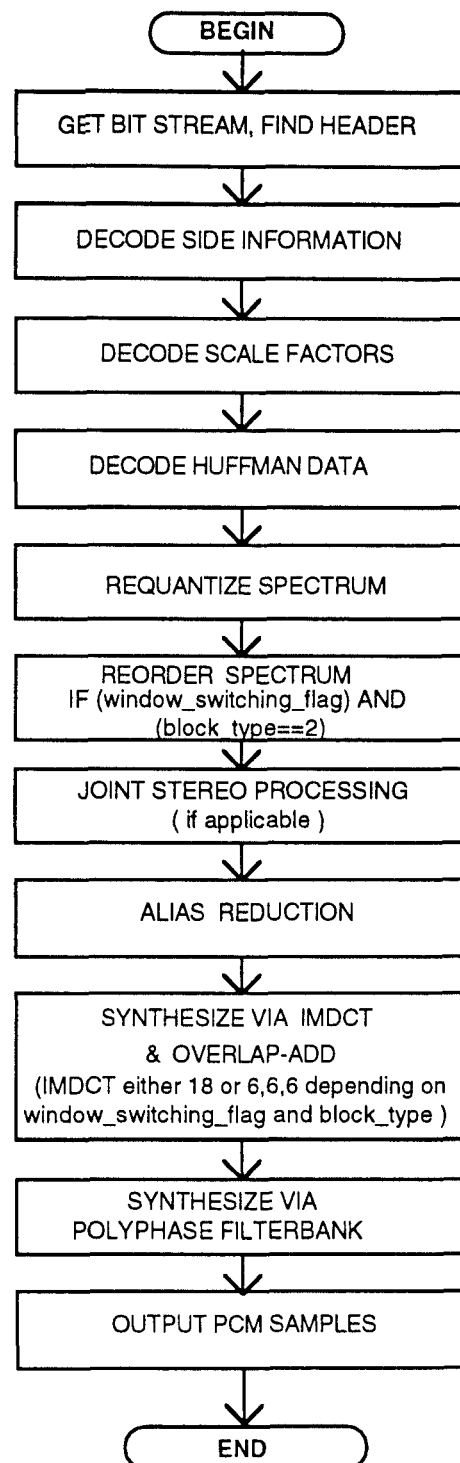


Figure A.3 -- Layer III decoder flow chart

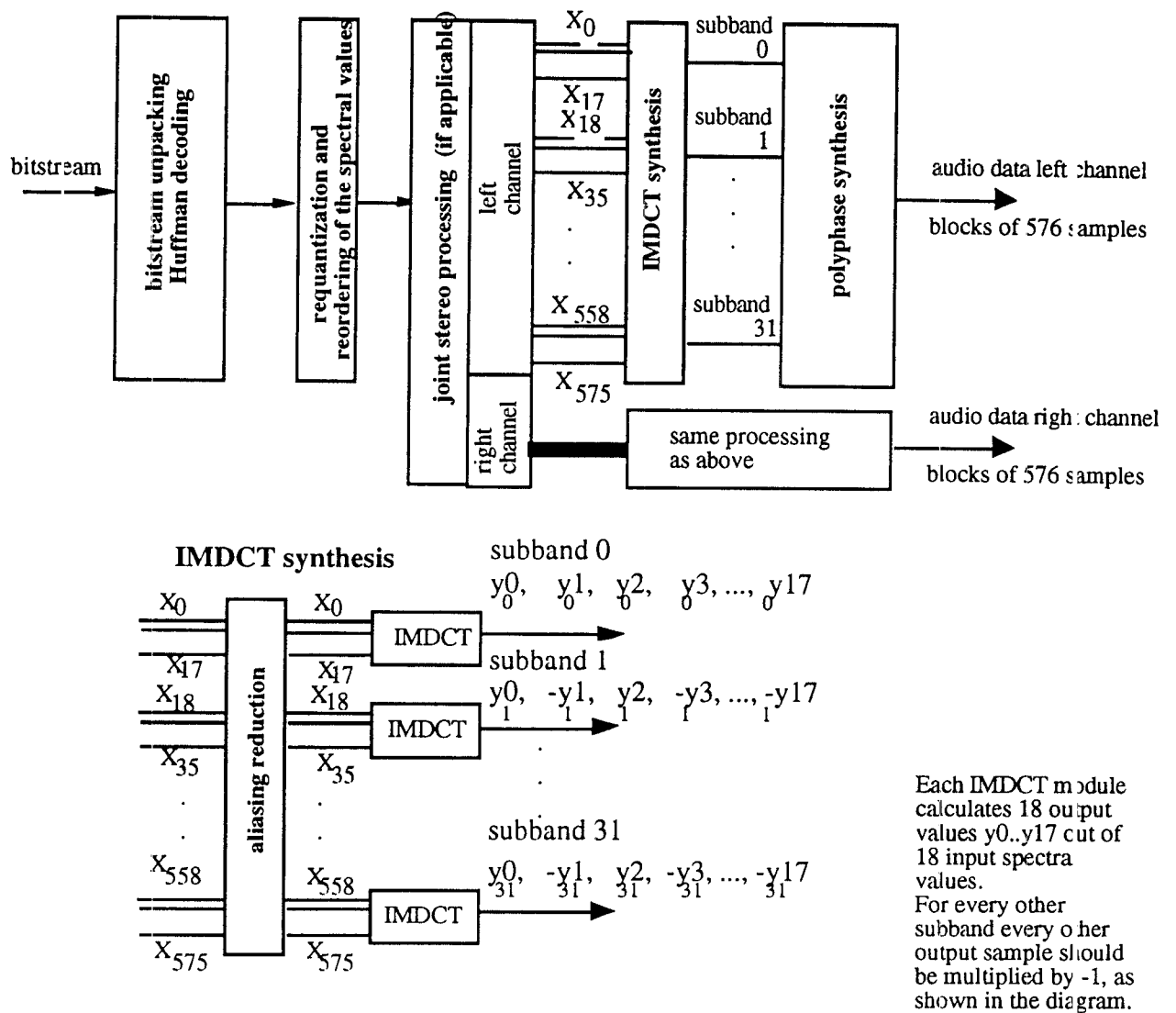


Figure A.4 -- Layer III decoder diagram

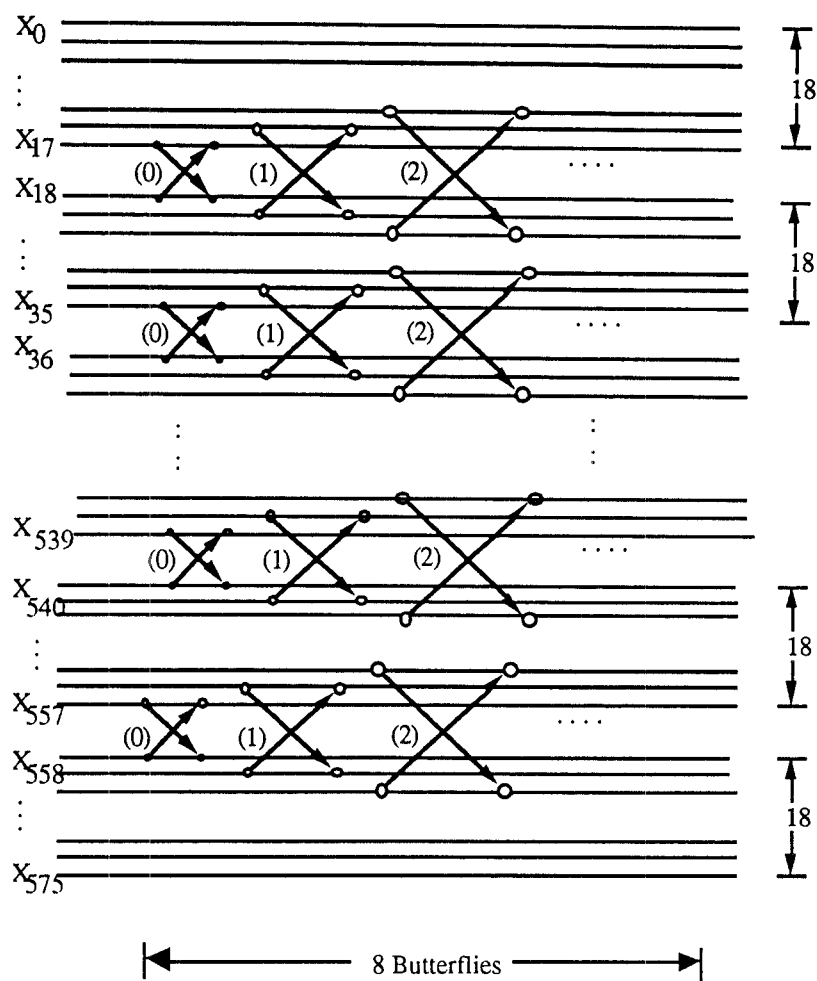


Figure A.5 -- Layer III aliasing reduction decoder diagram

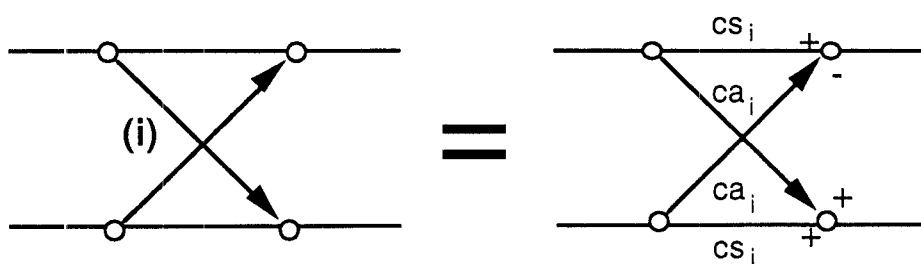


Figure A.6 -- Layer III aliasing-butterfly, decoder