TITLE PAGE PROVIDED BY ISO

CD 11172-3
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO FOR DIGITAL STORAGE MEDIA AT UP TO ABOUT 1.5 MBIT/s
Part 3 AUDIO
CONTENTS

FOREWORD

This standard is a committee draft that was submitted for approval to ISO-IEC/JTC1 SC29 on 22 November 1991. It was prepared by SC29/WG11, also known as MPEG (Moving Pictures Expert Group). MPEG was formed in 1988 to establish a standard for the coded representation of moving pictures and associated audio stored on digital storage media.

This standard is published in four parts. Part 1 - systems - specifies the system coding layer of the standard. It defines a multiplexed structure for combining audio and video data and means of representing the timing information needed to replay synchronized sequences in real-time. Part 2 - video - specifies the coded representation of video data and the decoding process required to reconstruct pictures. Part 3 - audio - specifies the coded representation of audio data. Part 4 - conformance testing - is still in preparation. It will specify the procedures for determining the

characteristics of coded bit streams and for testing compliance with the requirements stated in Parts 1, 2 and 3.

In Part 1 of this standard all annexes are informative and contain no normative requirements.

In Part 2 of this standard 2-Annex A, 2-Annex B and 2-Annex C contain normative requirements and are an integral part of this standard. 2-Annex D and 2-Annex E are informative and contain no normative requirements.

In Part 3 of this standard 3-Annex A and 3-Annex B contain normative requirements and are an integral part of this standard. All other annexes are informative and contain no normative requirements.

INTRODUCTION

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.

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.

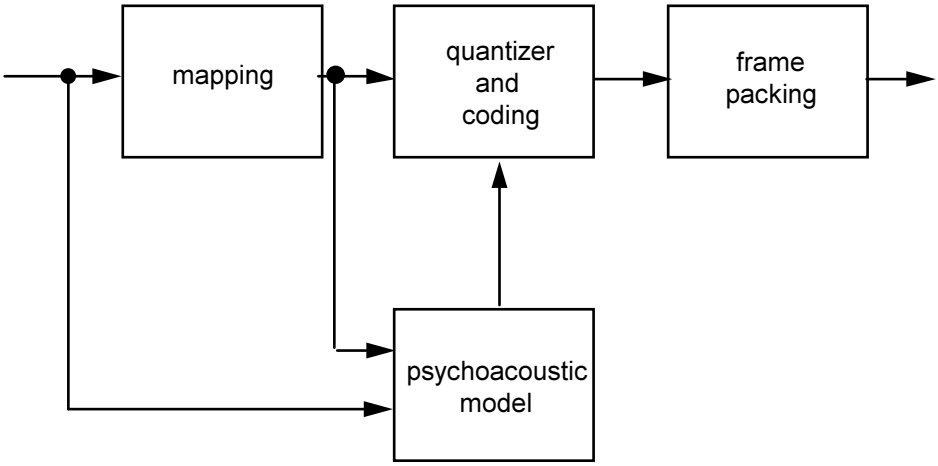

Figure I-1 Sketch of a basic encoder

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, 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.

Layers

Depending on the application, different layers of the coding system with increasing encoder complexity and performance can be used. An ISO MPEG 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.

Layer II:
This layer provides additional coding of bit allocation, scalefactors and samples. Different framing is used.

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 .

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

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.

Decoding

The decoder, subject to the application-dependent parameters of clause 2.4.1, accepts the compressed audio bitstream in the syntax defined in clause 2.4.2, decodes the data elements according to clause 2.4.3, and uses the information to produce digital audio output according to clause 2.4.4.

Figure I-2 Sketch of the basic structure of the 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 clause 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.

1. GENERAL NORMATIVE ELEMENTS

1.1 Scope

This standard 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 standard is intended for application to digital storage media providing a total continuous transfer rate of about 1.5 Mbit/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 11172 multiplex stream defined in Part1 of this standard. This standard is intended for sampling rates of 32 kHz, 44.1 kHz, and 48 kHz.

1.2 References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard 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.

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

Volume X
Rec. 953 "Encoding parameters of digital audio".

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

2. TECHNICAL NORMATIVE ELEMENTS

2.1 Definitions

For the purposes of this International Standard, the following definitions apply.

AC coefficient: Any DCT coefficient for which the frequency in one or both dimensions is non-zero.

access unit: 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.

Adaptive segmentation: A subdivision of the digital representation of an audio signal in variable segments of time.

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

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

Alias: Mirrored signal component resulting from sub-Nyquist sampling.

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

Audio Access Unit: 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".

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

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

Bark: Unit of critical band rate.

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

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

Block companding: Normalizing of the digital representation of an audio signal within a certain time period.

block: An 8-row by 8-column orthogonal block of pels.

Bound: The lowest subband in which intensity stereo coding is used.

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.

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

chrominance (component): A matrix, block or sample of pels 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.

coded audio bitstream: A coded representation of an audio signal as specified in this International Standard.

coded video bitstream: A coded representation of a series of one or more pictures as specified in this International Standard.

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

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

coding parameters: The set of user-definable parameters that characterise a coded video bitstream. Bit-streams are characterised by coding parameters. Decoders are characterised by the bitstreams that they are capable of decoding.

component: A matrix, block or sample of pel data from one of the three matrices (luminance and two chrominance) that make up a picture.

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

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

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

Constrained Parameters: In the case of the video specification, the values of the set of coding parameters defined in Part 2 Clause 2.4.4.4.

constrained system parameter stream (CSPS): An ISO 11172 multiplexed stream for which the constraints defined in Part 1 Clause 2.4.6 apply.

CRC: Cyclic redundancy code.

Critical Band Rate: Psychoacoustic measure in the spectral domain which corresponds to the frequency selectivity of the human ear.

Critical Band: Part of the spectral domain which corresponds to a width of one Bark.

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

DC-coefficient: The DCT coefficient for which the frequency is zero in both dimensions.

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

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

decoded stream: The decoded reconstruction of a compressed bit stream.

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

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

decoder: An embodiment of a decoding process.

decoding process: The process defined in this International Standard that reads an input coded bitstream and outputs decoded pictures or audio samples.

decoding time-stamp; DTS: 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.

Dequantization [Audio]: Decoding of coded subband samples in order to recover the original quantized values.

dequantization: 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.

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

discrete cosine transform; DCT: 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 2-Annex A of Part 2.

display order: 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.

editing: The process by which one or more compressed bitstreams are manipulated to produce a new compressed bitstream. Conforming editted bitstreams must meet the requirements defined in this International Standard.

elementary stream: A generic term for one of the coded video, coded audio or other coded bit streams.

encoder: An embodiment of an encoding process.

encoding process: A process, not specified in this International Standard, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this International Standard.

Entropy coding: Variable length noiseless coding of the digital representation of a signal to reduce redundancy.

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

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

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

Fixed segmentation: A subdivision of the digital representation of an audio signal in to fixed segments of time.

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.

forced updating: 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.

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

Frame [audio]: A part of the audio signal that corresponds to a fixed number of audio PCM samples.

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

Granules [Layer II]: 96 subband samples, 3 consecutive subband samples for all 32 subbands that are considered together before quantisation

Granules [Layer III]: 576 frequency lines that carry their own side information.

group of pictures: A series of one or more pictures intended to assist random access. The group of pictures is one of the layers in the coding syntax defined in Part 2 of this International Standard.

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

Huffman coding: A specific method for entropy coding.

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

IMDCT: Inverse Modified Discrete Cosine Transform.

Intensity stereo: 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.

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

intra coding: Compression coding of a block or picture that uses information only from that block or picture.

intra-coded picture; I-picture: A picture coded using information only from itself.

ISO 11172 (multiplexed) stream: A bitstream composed of zero or more elementary streams combined in the manner defined in Part 1 of this International Standard.

Joint stereo coding: Any method that exploits stereophonic irrelevance or stereophonic redundancy.

Joint stereo mode: A mode of the audio coding algorithm using joint stereo coding.

layer [audio]: One of the levels in the coding hierarchy of the audio system defined in this International Standard.

layer [video and systems]: One of the levels in the data hierarchy of the video and system specifications defined in Parts 1 and 2 of this International Standard.

luminance (component): A matrix, block or sample of pels 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.

macroblock: 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 and other data elements defined in the macroblock layer of the syntax defined in Part 2 of this International Standard. The usage is clear from the context.

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

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

Masking: property of the human auditory system by which an audio signal cannot be perceived in the presence of another audio signal .

MDCT: Modified Discrete Cosine Transform.

motion compensation: 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 frames containing previously decoded pels that are used to form the prediction.

motion vector estimation: The process of estimating motion vectors during the encoding process.

motion vector: 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.

MS stereo: 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.

non-intra coding: Coding of a block or picture that uses information both from itself and from blocks and pictures occurring at other times.

Non-tonal component: A noise-like component of an audio signal.

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

pack: A pack consists of a pack header followed by one or more packets. It is a layer in the system coding syntax described in Part 1 of this standard.

packet data: Contiguous bytes of data from an elementary stream present in a packet.

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

packet: 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 Part 1 of this International Standard.

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

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

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

pel: An 8-bit sample of luminance or chrominance data.

picture period: The reciprocal of the picture rate.

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

picture: Source or reconstructed image data. A 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 Part 2 of this International Standard. NOTE: the term "picture" is always used in this standard in preference to the terms field or frame.

Polyphase filterbank: A set of equal bandwidth filters with special phase interrelationships, allowing for an efficient implementation of the filterbank.

prediction: The use of predictor to provide an estimate of the pel or data element currently being decoded.

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

predictor: A linear combination of previously decoded pels or data elements.

presentation time-stamp; PTS: 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.

presentation unit: A decoded audio access unit or a decoded picture.

Psychoacoustic model: A mathematical model of the masking behaviour of the human auditory system.

quantization matrix: A set of sixty-four 8-bit scaling values used by the dequantizer.

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

quantizer scale factor: A data element represented in the bitstream and used by the decoding process to scale the dequantization.

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

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

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

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 defined extensions.

reverse play: The process of displaying the picture sequence in the reverse of display order.

Scalefactor band: A set of frequency lines in Layer III which are scaled by one scalefactor.

Scalefactor index: A numerical code for a scalefactor.

Scalefactor: Factor by which a set of values is scaled before quantization.

sequence header: A block of data in the coded bitstream containing the coded representation of a number of data elements. It is one of the layers of the coding syntax defined in Part 2 of this International Standard.

Side information: Information in the bitstream necessary for controlling the decoder.

skipped macroblock: A macroblock for which no data is stored.

slice: A series of macroblocks. It is one of the layers of the coding syntax defined in Part 2 of this International Standard.

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.

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

Spreading function: A function that describes the frequency spread of masking.

start codes: 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.

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

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.

Subband [audio]: Subdivision of the audio frequency band.

Subband filterbank: A set of band filters covering the entire audio frequency range. In Part 3 of this International Standard the subband filterbank is a polyphase filterbank.

Syncword: A 12-bit code embedded in the audio bitstream that identifies the start of a frame.

Synthesis filterbank: Filterbank in the decoder that reconstructs a PCM audio signal from subband samples.

system header: The system header is a data structure defined in Part 1 of this International Standard that carries information summarising the system characteristics of the ISO 11172 multiplexed stream.

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

time-stamp: A term that indicates the time of an event.

Tonal component: A sinusoid-like component of an audio signal.

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

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.

video buffering verifier; VBV: 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.

video sequence: A series of one or more groups of pictures.

zig-zag scanning order: 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 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 two's-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 toward -8.
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign(x) = 1 x > 0 0 x == 0 -1 x < 0
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
log10	Logarithm to base ten
loge	Logarithm to base e

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

& 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 the standard. 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.

uimsbf Unsigned integer, most significant bit first.

vlclbf Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written in Annex B.

ch channel.

gr granule of three sub-band samples in audio layer 2, twelve sub-band samples in audio layer 3.

rpchof remainder polynomial coefficients, highest order first.

sb sub-band.

scfsi scale-factor selector information.

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

2.2.7 Constants

pi 3.14159265359...
e 2.71828182846...

2.3 Method of Describing Bit Stream Syntax

The bit stream retrieved by the decoder is described in clause 2.4.2. Each data item in the bit stream 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 bit stream 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 clause 2.4.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:

while (condition) { If the condition is true, then the group of data elements occurs next

data_element in the data stream. This repeats until the condition is not true.
. . .
}

do {
data_element The data element always occurs at least once.
. . .
} while (condition) The data element is repeated until the condition is not true.

if (condition) { If the condition is true, then the first group of data elements occurs
data_element next in the data stream.
. . .
}
else { If the condition is not true, then the second group of data elements
data_element occurs next in the data stream.
. . .
}

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 {} are omitted when only one data element follows.

data_element [n]data_element [n] is the n+1th element of an 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 clause 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, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardized.

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 bit stream is the first bit in a byte.

Definition of nextbits function

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

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.

next_start_code() {
while (!bytealigned())
zero_bit 1 "0"
while (nextbits() != '0000 0000 0000 0000 0000 0001')
zero_byte 8 "00000000"
}

2.4 Requirements

2.4.1 Specification of the Coded Audio Bitstream Syntax

2.4.1.1 Audio Sequence

```
audio sequence()
 {
  while (true)
  {
      frame()
  }
 }
```

2.4.1.2 Audio Frame

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

2.4.1.3 Header

```
header()
{
   syncword                                    12   bits
   bslbf
   ID                                          1    bit
   bslbf
   layer                                       2    bits
   bslbf
   protection_bit                              1    bit
   bslbf
   bitrate_index                               4    bits
   bslbf
   sampling_frequency                          2    bits
   bslbf
   padding_bit                                 1    bit
   bslbf
   private_bit                                 1    bit
   bslbf
   mode                                        2    bits
   bslbf
   mode_extension                              2    bits
   bslbf
   copyright                                   1    bit
   bslbf
   original/home                               1    bit
   bslbf
   emphasis                                    2    bits
   bslbf
}
```

2.4.1.4 Error check

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

2.4.1.5 Audio data, Layer I

```
audio_data()
{
```

```
if (mode==single_channel)
{
 for (sb=0; sb<32; sb++)
  allocation[sb]                                    4     bits
  uimsbf
 for (sb=0; sb<32; sb++)
  if (allocation[sb]!=0)
   scalefactor[sb]                                  6     bits
  uimsbf
 for (s=0; s<12; s++)
  for (sb=0; sb<32; sb++)
   if (allocation[sb]!=0)
    sample[sb][s]                                   2..15 bits
  uimsbf
}
if (mode==stereo) || (mode==dual_channel)
{
 for (sb=0; sb<32; sb++)
  for (ch=0; ch<2; ch++)
   allocation[ch][sb]                               4     bits
  bsmsbf
 for (sb=0; sb<32; sb++)
  for (ch=0; ch<2; ch++)
   if (allocation[ch][sb]!=0)
    scalefactor[ch][sb]                             6     bits
  uimsbf
 for (s=0; s<12; s++)
  for (sb=0; sb<32; sb++)
   for (ch=0; ch<2; ch++)
    if (allocation[ch][sb]!=0)
     sample[ch][sb][s]                              2..15 bits
  uimsbf
 }
 if (mode==intensity_stereo)
 {
  for (sb=0; sb<bound; sb++)
   for (ch=0; ch<2; ch++)
    allocation[ch][sb]                              4     bits
  uimsbf
  for (sb=bound; sb<32; sb++)
   allocation[sb]                                   4     bits
  uimsbf
  for (sb=0; sb<bound; sb++)
   for (ch=0; ch<2; ch++)
    if (allocation[ch][sb]!=0)
     scalefactor[ch][sb]                            6     bits
  uimsbf
  for (sb=bound; sb<32; sb++)
   for (ch=0; ch<2; ch++)
    if (allocation[sb]!=0)
     scalefactor[ch][sb]                            6     bits
  uimsbf
     for (s=0; s<12; s++)
  {
   for (sb=0; sb<bound; sb++)
    for (ch=0; ch<2; ch++)
     if (allocation[ch][sb]!=0)
      sample[ch][sb][s]                             2..15 bits
  uimsbf
   for (sb=bound; sb<32; sb++)
    if (allocation[sb]!=0)
     sample[sb][s]                                  2..15 bits
  uimsbf
  }
 }
}
```

2.4.1.6 Audio data, Layer II

```
audio_data()
```

```
{
 if (mode==single_channel)
 {
  for (sb=0; sb<sblimit; sb++)
   allocation[sb]                                  2..4 bits
   uimsbf
  for (sb=0; sb<sblimit; sb++)
   if (allocation[sb]!=0)
    scfsi[sb]                                      2    bits
   bslbf
  for (sb=0; sb<sblimit; sb++)
   if (allocation[sb]!=0)
   {
    if (scfsi[sb]==0)
    { scalefactor[sb][0]                           6    bits
   uimsbf
      scalefactor[sb][1]                           6    bits
   uimsbf
      scalefactor[sb][2] }                         6    bits
   uimsbf
    if (scfsi[sb]==1) || (scfsi[sb]==3)
    { scalefactor[sb][0]                           6    bits
   uimsbf
      scalefactor[sb][2] }                         6    bits
   uimsbf
    if (scfsi[sb]==2)
      scalefactor[sb][0]                           6    bits
   uimsbf
   }
   for (gr=0; gr<12; gr++)
    for (sb=0; sb<sblimit; sb++)
     if (allocation[sb]!=0)
     {
      if (grouping[sb])
       samplecode[sb][gr]                          5..10bits
   uimsbf
      else for (s=0; s<3; s++)
       sample[sb][3*gr+s]                          2..16bits
   uimsbf
     }
 }

if (mode==stereo) || (mode==dual_channel)
 {
  for (sb=0; sb<sblimit; sb++)
   for (ch=0; ch<2; ch++)
     allocation[ch][sb]                            2..4 bits
   uimsbf
  for (sb=0; sb<sblimit; sb++)
   for (ch=0; ch<2; ch++)
    if (allocation[ch][sb]!=0)
     scfsi[ch][sb]                                 2    bits
   bslbf
    for (sb=0; sb<sblimit; sb++)
   for (ch=0; ch<2; ch++)
    if (allocation[ch][sb]!=0)
    {
     if (scfsi[ch][sb]==0)
     { scalefactor[ch][sb][0]                      6    bits
   uimsbf
       scalefactor[ch][sb][1]                      6    bits
   uimsbf
       scalefactor[ch][sb][2] }                    6    bits
   uimsbf
     if (scfsi[ch][sb]==1) || (scfsi[ch][sb]==3)
     { scalefactor[ch][sb][0]                      6    bits
   uimsbf
       scalefactor[ch][sb][2] }                    6    bits
   uimsbf
```

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

if (mode==intensity_stereo)
  {
   for (sb=0; sb<bound; sb++)
    for (ch=0; ch<2; ch++)
     allocation[ch][sb]                                2..4 bits
    uimsbf
    for (sb=bound; sb<sblimit; sb++)
     allocation[sb]                                    2..4 bits
    uimsbf
    for (sb=0; sb<bound; sb++)
     for (ch=0; ch<2; ch++)
     if (allocation[ch][sb]!=0)
       scfsi[ch][sb]                                   2    bits
    bslbf
    for (sb=bound; sb<sblimit; sb++)
     for (ch=0; ch<2; ch++)
     if (allocation[sb]!=0)
       scfsi[ch][sb]                                   2    bits
    bslbf
    for (sb=0; sb<bound; sb++)
     for (ch=0; ch<2; ch++)
     if (allocation[ch][sb]!=0)
      {
       if (scfsi[ch][sb]==0)
       { scalefactor[ch][sb][0]                        6    bits
    uimsbf
         scalefactor[ch][sb][1]                        6    bits
    uimsbf
         scalefactor[ch][sb][2] }                      6    bits
    uimsbf
       if (scfsi[ch][sb]==1) || (scfsi[ch][sb]==3)
       { scalefactor[ch][sb][0]                        6    bits
    uimsbf
         scalefactor[ch][sb][2] }                      6    bits
    uimsbf
       if (scfsi[ch][sb]==2)
         scalefactor[ch][sb][0]                        6    bits
    uimsbf
      }
    for (sb=bound; sb<sblimit; sb++)
     for (ch=0; ch<2; ch++)
     if (allocation[sb]!=0)
      {
       if (scfsi[ch][sb]==0)
       { scalefactor[ch][sb][0]                        6    bits
    uimsbf
         scalefactor[ch][sb][1]                        6    bits
    uimsbf
         scalefactor[ch][sb][2] }                      6    bits
    uimsbf
       if (scfsi[ch][sb]==1) || (scfsi[ch][sb]==3)
```

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

2.4.1.7 Audio data, Layer III

```
audio_data()
{
 if (mode == single_channel)
 {
  main_data_end                                       9     bits
                                                      uimsbf
  private_bits                                        5     bits
                                                      bslbf
  for (scfsi_band=0; scfsi_band<4; scfsi_band++)
   scfsi[scfsi_band]                                  1     bits
                                                      bslbf
  for (gr=0; gr<2; gr++)
   {
    part2_3_length[gr]                                12    bits
                                                      uimsbf
    big_values[gr]                                    9     bits
                                                      uimsbf
    global_gain[gr]                                   8     bits
                                                      uimsbf
    scalefac_compress[gr]                             4     bits
                                                      bslbf
    blocksplit_flag[gr]                               1     bit
                                                      bslbf
    if (blocksplit_flag[gr])
      {
       block_type[gr]                                 2     bits
                                                      bslbf
       switch_point[gr]                               1     bits
                                                      uimsbf
       for (region=0; region<2; region++)
```

```
          table_select[region][gr]                                5    bits
                                                                  bslbf
        for (window=0; window<3; window++)
        subblock_gain[window][gr]                                 3    bits
                                                                  uimsbf
      }
    else
      {
        for (region=0; region<3; region++)
          table_select[region][gr]                                5    bits
                                                                  bslbf
        region_address1[gr]                                       4    bits
                                                                  bslbf
        region_address2[gr]                                       3    bits
                                                                  bslbf
      }
    preflag[gr]                                                   1    bit
                                                                  bslbf
    scalefac_scale[gr]                                            1    bit
                                                                  bslbf
    count1table_select[gr]                                        1    bit
                                                                  bslbf
   }

    /***
    The main_data follows.  It does not follow the above side information
    in the bitstream.  The main_data ends at a location in the main_data
    bitstream preceding the frame header of the following frame at an
    offset given by the value of main_data_end (see definition of
    main_data_end and 3-Annex Fig.3-A.7.1)
    ***/
  for (gr=0; gr<2; gr++)
    if (blocksplit_flag[gr] == 1 && block_type[gr] == 2)
    {
     for (cb=0; cb<switch_point_l[gr]; cb++)
      if (scfsi[cb]==0) || (gr==0)
       scalefac[cb][gr]                                           0..4  bits
                                                                  uimsbf
     for (cb=switch_point_s[gr]; cb<cblimit_short; cb++)
      for (window=0; window<3; window++)
       if (scfsi[cb]==0) || (gr==0)
        scalefac[cb][window][gr]                                  0..4  bits
                                                                  uimsbf
    }
    else
     for (cb=0; cb<cblimit; cb++)
      if (scfsi[cb]==0) || (gr==0)
       scalefac[cb][gr]                                           0..4  bits
                                                                  uimsbf
    Huffmancodebits
(part2_3_length-part2_length)                                     bits  bslbf
    while (position != main_data_end)
    {
       ancillary_bit                                              1     bit
   bslbf
    }

 }

if (mode==stereo) || (mode==dual_channel) || (mode==ms_stereo)
 {
  main_data_end                                                         9 bits
                                                                  uimsbf
  private_bits                                                          3 bits
                                                                  bslbf
  for (ch=0; ch<2; ch++)
   for (scfsi_band=0; scfsi_band<4; scfsi_band++)
    scfsi[scfsi_band][ch]                                               1 bits
                                                                  bslbf
```

```
 for (gr=0; gr<2; gr++)
  for (ch=0; ch<2; ch++}
   part2_3_length[gr][ch]                                  12    bits
                                                           uimsbf
   big_values[gr][ch]                                      9     bits
                                                           uimsbf
   global_gain[gr][ch]                                     8     bits
                                                           uimsbf
   scalefac_compress[gr][ch]                               4     bits
                                                           bslbf
   blocksplit_flag[gr][ch]                                 1     bit
                                                           bslbf
   if (blocksplit_flag[gr][ch])
     {
      block_type[gr][ch]                                   2     bits
                                                           bslbf
      switch_point[gr][ch]                                 1     bits
                                                           uimsbf
      for (region=0; region<2; region++)
        table_select[region][gr][ch]                       5     bits
                                                           bslbf
      for (window=0; window<3; window++)
        subblock_gain[window][gr][ch]                      3     bits
                                                           uimsbf
     }
   else
     {
      for (region=0; region<3; region++)
        table_select[region][gr][ch]                       5     bits
                                                           bslbf
       region_address1[gr][ch]                             4     bits
                                                           bslbf
       region_address2[gr][ch]                             3     bits
                                                           bslbf
     }
   preflag[gr][ch]                                         1     bit
                                                           bslbf
   scalefac_scale[gr][ch]                                  1     bit
                                                           bslbf
   count1table_select[gr][ch]                              1     bit
                                                           bslbf

   /***
   The main_data follows. It does not follow the above side information
   in the bitstream. The main_data ends at a location in the main_data
   bitstream preceding the frame header of the following frame at an
   offset given by the value of main_data_end.
   ***/
 for (gr=0; gr<2; gr++)
   for (ch=0; ch<2; ch++) {
    if (blocksplit_flag[gr][ch] == 1 && block_type[gr][ch] == 2)
    {
     for (cb=0; cb<switch_point_l[gr][ch]; cb++)
      if (scfsi[cb]==0) || (gr==0)
       scalefac[cb][gr][ch]                                0..4  bits
                                                           uimsbf
     for (cb=switch_point_s[gr][ch]; cb<cblimit_short; cb++)
      for (window=0; window<3; window++)
       if (scfsi[cb]==0) || (gr==0)
        scalefac[cb][window][gr][ch]                       0..4  bits
                                                           uimsbf
    }
    else
     for (cb=0; cb<cblimit; cb++)
      if (scfsi[cb]==0) || (gr==0)
       scalefac[cb][gr][ch]                                0..4  bits
                                                           uimsbf
    Huffmancodebits                                        (part2_3_length-
part2_length)                                              bits  bslbf
```

```
    while (position != main_data_end)
    {
       ancillary_bit                                          1      bit
   bslbf
    }

   }
 }
}
```

2.4.1.8 Ancillary data

```
if(layer == 1 || layer == 2)
{
  ancillary_data()
  {
   while (nextbits() != syncword)
   {
   ancillary_bit                                              1      bit
   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 1152 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 side and main information. In Layer III it contains information for 1152 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 MPEG audio, '0' is reserved.
`Layer` - 2 bits to indicate which layer is used, according to the following table.

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

To change the layer, a reset of the decoder is 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.
bit_rate_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 bit_rate_index is an index to a table, which is different for the different Layers.
The bit_rate_index indicates the total bitrate irrespective of the mode (stereo, joint_stereo, dual_channel, single_channel).
For Layer II, not all combinations of total bitrate and mode are allowed. See 3-Annex B, Table 3-B.2 "LAYER II BIT ALLOCATION TABLES".

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

In order to provide the smallest possible delay and complexity, the decoder is not required to support a continuously variable bitrate when in LayerI or II. Layer III supports variable bitrate by switching the bit_rate_index. However, in free format, fixed bitrate is required.

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

'00'	44.1 kHz
'01'	48 kHz
'10'	32 kHz
'11'	reserved

A reset of the decoder is 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 only necessary with a sampling frequency of 44.1kHz.

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

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

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.

'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.

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

copyright - if this bit equals '0' there is no copyright on the coded bitstream, '1' means copyright protected.

original/home - 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.

'00'	no emphasis
'01'	50/15 microsec. emphasis
'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[sb] - indicates the number of bits used to code the samples in subband sb. Valid for single_channel subbands and for subbands in intensity_stereo mode. In the latter case the allocation is valid for both channels.

CODE	BITS
'0000'	0
'0001'	2
'0010'	3
'0011'	4
'0100'	5
'0101'	6
'0110'	7
'0111'	8
'1000'	9
'1001'	10
'1010'	11
'1011'	12
'1100'	13
'1101'	14
'1110'	15
'1111'	invalid

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

allocation[ch][sb] - same as allocation[sb] but now for the channel ch in stereo or dual_channel mode.

scalefactor[sb] - indicates the factor of subband sb by which the requantized samples of subband sb shall be multiplied. The six bits constitute an unsigned integer, index to 3-Annex B, Table 3-B.1 "LAYER I, II SCALEFACTORS". Valid for single_channel mode.

scalefactor[ch][sb] - same as scalefactor[sb] but now for one of the channels in stereo, intensity_stereo, or dual_channel mode.

sample[sb][s] - coded representation of the s-th sample in subband sb. Valid for single_channel subbands and for subbands in intensity_stereo mode. In the latter case the value is valid for both channels.

sample[ch][sb][s] - same as sample[sb][s] but for the channel ch in stereo or dual_channel mode.

2.4.2.6 Audio data, Layer II

allocation[sb] - contains information concerning the quantizers used for the samples in subband sb, 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 3-Annex B, Table 3-B.2 "LAYER II BIT ALLOCATION TABLES", which gives the number of levels used for quantization. 3-Annex B, Table 3-B.4 "LAYER II CLASSES OF QUANTIZATION" gives additional information concerning each possible quantizer: the requantization coefficients, whether grouping has been used, the number of samples per codeword, and the number of bits per codeword. Several tables exist for different combinations of bitrate and sampling frequency, see 3-Annex B, Table 3-B.2 "LAYER II BIT ALLOCATION TABLES". This is valid for single_channel subbands or for subbands in intensity_stereo mode. In the latter case the allocation is valid for both channels.

allocation[ch][sb] - same as allocation[sb] but now for channel ch in stereo or dual_channel mode.

gr - index to a granule. In Layer II, a granule consists of 3 consecutive samples from each of the 32 subbands.

scfsi[sb] - scalefactor selection information. This gives information on the number of scalefactors transferred for subband sb 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.

'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.

Valid in single_channel mode.

scfsi[ch][sb] - same as scfsi[sb] but now for the channel ch in stereo, intensity_stereo, or dual_channel mode.

scalefactor[sb][p] - indicates the factor by which the requantized samples of subband sb and of part p of the frame should be multiplied. The six bits constitute an unsigned integer, index to 3-Annex B, Table 3-B.1 "LAYER I, II SCALEFACTORS'. Valid in single_channel mode.

scalefactor[ch][sb][p] - same as scalefactor[sb][p] but now for one of the channels ch in stereo, intensity_stereo, or dual_channel mode.

samplecode[sb][gr] - coded representation of the three consecutive samples in the granule gr in subband sb. Valid for single_channel subbands and for subbands in intensity_stereo mode. In the latter case, the code is valid for both channels.

samplecode[ch][sb][gr] - same as samplecode[sb][gr] but now for channel ch in stereo or dual_channel mode.

sample[sb][s] - coded representation of the s-th sample in subband sb. Valid for single_channel subbands and for subbands in intensity_stereo mode. In the latter case the value is valid for both channels.

sample[ch][sb][s] - same as sample[sb][s] but now for the channel ch in stereo or dual_channel mode.

2.4.2.7 Audio data, Layer III

gr - the granules in Layer III consist of 18 * 32 subband samples. Each frame holds the data from 2 granules. The audio data in a frame is allocated in the following way:

- main_data_end pointer
- side info for both granules (scfsi)
- side info granule 1
- side info granule 2
- scalefactors and Huffman code data granule 1
- scalefactors and Huffman code data granule 2

main_data_end - The value of main_data_end is used to determine the location in the bitstream of the last bit of main_data for the frame. The main_data_end value specifies the location as a negative offset in bytes from the next frame's frame header location in the main_data portion of the bitstream. This is explained in 3-Annex Fig. 3- A.7.1.

private_bits - bits for private use. These bits will not be used in the future by ISO.

main_data_beg - This gives 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.

main_data - The main_data portion of the bitstream contains the scalefactors, Huffman encoded data, and ancillary information.

scfsi[scfsi_band] - in Layer III the scalefactor selection information works similarly to Layers I and II. The main difference is the use of the variable scfsi_band to apply scfsi to groups of scalefactors instead of single scalefactors. scfsi controls the use of scalefactors to the granules.

```
'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[scfsi_band][ch] - same as scfsi[scfsi_band] but for use in stereo, joint_stereo or dual_channel mode

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

```
scfsi_band  scalefactor bands (see 3-Annex B, Table 3-B.8)
 0           0,1,2,3,4,5,
1    6,7,8,9,10,
2    11 ... 15
3    16 ... 20
```

part2_3_length[gr] - 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 each granule and the position of ancillary information (if used).

part2_3_length[gr][ch] - same as part2_3_length[gr] but for use in stereo, joint_stereo or dual_channel mode

part2_length - this value contains the number of main_data bits used for scalefactors. Its value is given as follows:
For switch_point == 0,
part2_length = 11 * slen1 + 10 * slen2 for long blocks (block_type 0, 1 or 3), and
part2_length = 18 * slen1 + 18 * slen2 for short blocks (block_type==2).

For switch_point == 1,

part2_length = 17 * slen1 + 18 * slen2 (block_type == 2), and

for long blocks (block type 0, 1, or 3) the value of part2_length is the same as that for switch_point == 0.

big_values[gr] - 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 don't 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 remains. 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 figure shows the partitioning:

```
xxxxxxxxxxxxx-----------------000000000000000000000000000000
|           |                 |                             |
1           bigvalues*2       bigvalues*2+count1*4          iblen
The values 000 are all zero.
The values --- are -1,0 or +1. Their number is a multiple of 4.
The values xxx are not bound.
Iblen is 576.
```

big_values[gr][ch] - same as big_values[gr] but for use in stereo, joint_stereo or dual_channel mode

global_gain[gr] - 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".

global_gain[gr][ch] - same as global_gain[gr] but for use in stereo, joint_stereo or dual_channel mode

scalefac_compress[gr] - 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 switch_point 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 switch_point is 1:
slen1: length of scalefactors for the scalefactor bands 0 to 7 (long window scalefactor band) and 4 to 5 (short window scalefactor band) Note: Scalefactor bands 0-7 are from the "long window scalefactor band" table, and scalefactor bands 4-11 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	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

scalefac_compress[gr][ch] - same as scalefac_compress[gr] but for use in stereo, joint_stereo or dual_channel mode

blocksplit_flag[gr] - signals that the block uses an other than normal (type 0) window. If blocksplit_flag is set, several other variables are set by default:

region_address1 = 8 (in case of block_type==1 or block_type==3)
region_address1 = 9 (in case of block_type==2)
region_address2 = 0 In this case the length of region 2 is zero.

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

blocksplit_flag[gr][ch] - same as blocksplit_flag[gr] but for use in stereo, joint_stereo or dual_channel mode

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

type 0	reserved
type 1	start block
type 2	3 short windows
type 3	end block

Block_type and switch_point give the information about assembling of values in the block and about length and count of the transforms (see 3-Annex A, Figure 3-A.4 for a schematic, 3-Annex C for an analytic description). In the case of block_type=2 the switch_point indicates whether some polyphase filter subbands are coded using long transforms even in case of block_type 2. The polyphase filterbank is described in the clause 2.4.3.2 Layer I.
- In the case of long blocks (block_type not equal to 2 or in the lower subbands of block_type 2) 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) 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.

block_type[gr][ch] - same as block_type[gr] but for use in stereo, joint_stereo or dual_channel mode

switch_point[gr] - signals the split point of short/long transforms. The following table shows the number of the scalefactor band above which window switching (i.e. block_type different from 0) is used.

switch_point	switch_point_l (No of sb)	switch_point_s (No of sb)

´0´	0	0	; switching of the whole spectrum
´1´	8	3	; switching of higher frequencies only

switch_point[gr][ch] - same as switch_point[gr] but for use in stereo, joint_stereo or dual_channel mode

switch_point_l - Number of scalefactor band (long block scalefactor band) from which point on window switching is used.

switch_point_s - Number of scalefactor band (short block scalefactor band) from which point on window switching is used.

cb_limit - Number of scalefactor bands for long blocks (block_type != 2). This is a constant, 21, for Layer III in all modes and at all sampling frequencies.

cb_limit_short - Number of scalefactor bands for short blocks (block_type=2). This is a constant, 12, for Layer III in all modes and at all sampling frequencies.

window - Number of actual time slot in case of block_type==2, 0 = window = 2.

table_select[region][gr] - 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 3-Annex B Table 3-B.7.

table_select[region][gr][ch] - same as table_select[region][gr] but for use in stereo, joint_stereo or dual_channel mode

subblock_gain[window][gr] - 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.^subblock_gain(window) in the decoder.

subblock_gain[window][gr][ch] - same as subblock_gain[window][gr] but for use in stereo, joint_stereo or dual_channel mode

region_address1[gr] - 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. Each region is coded using a different Huffman code table depending on the maximum quantized value and the local signal statistics.
The values region_address[1,2] are used to point to the boundaries of the regions. The region boundaries are aligned with the partitioning of the spectrum into critical bands.
In case of block_type==2 (short blocks) the scalefactor bands representing the different time slots are counted separately. If switch_point==0, the total amount of scalefactor bands for the granule in this case is 12*3=36. If block_type==2 and switch_point==1, the amount of scalefactor bands is 8+9*3=35. region_address1 counts the number of scalefactor bands until the upper edge of the first region:

region_address1	upper edge of region is upper edge of scalefactor band number:
0	0 (no first region)
1	1
2	2
...	...
15	15

region_address1[gr][ch] - same as region_address1[gr] but for use in stereo, joint_stereo or dual_channel mode

region_address2[gr] - region_address2 counts the number of scalefactor bands which are partially or totally in region 3. Again if block_type==2 the scalefactor bands representing different time slots are counted separately.

region_address2[gr][ch] - same as region_address2[gr] but for use in stereo, joint_stereo or dual_channel mode

preflag[gr] - 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 3-Annex B, Table 3-B.6). This is equivalent to multiplication of the requantized scalefactors with table values. preflag is never used if block_type==2 (short blocks).

preflag[gr][ch] - same as preflag[gr] but for use in stereo, joint_stereo or dual_channel mode

scalefac_scale[gr] - the scalefactors are logarithmically quantized with a step size of 2 or (|2) depending on scalefac_scale.

```
scalefac_scale = 0        stepsize sqrt(2)
scalefac_scale = 1        stepsize 2
```

scalefac_scale[gr][ch] - same as scalefac_scale[gr] but for use in stereo, joint_stereo or dual_channel mode

count1table_select[gr] - 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 = 0	Table A of 3-Annex B.7
count1table_select = 1	Table B of 3-Annex B.7

count1table_select[gr][ch] - same as count1table_select[gr] but for use in stereo, joint_stereo or dual_channel mode

scalefac[cb][gr] - the scalefactors are used to colour the quantization noise. If the quantization noise is colored with the right shape, it is masked completely. Unlike LayersI and II, the LayerIII 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 blocks of values. In the case of Layer III, the blocks stretch over several frequency lines. These blocks are called scalefactor bands and are selected to resemble critical bands as close 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 (see clause 2.4.3.4, MS_stereo mode).

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 3-Annex Table 3-B.8).

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

scalefac[cb][window][gr] - same as scalefac[cb][gr] but for different windows if block_type==2

scalefac[cb][gr][ch] - same as scalefac[ch][gr] but for use in stereo, joint_stereo or dual_channel mode

scalefac[cb][window][gr][ch] - same as scalefac[cb][window][gr] but for use in stereo, joint_stereo or dual_channel mode

Huffman_code_bits

To get a clear picture of the Huffman code syntax some pseudo-functions and structures have to be defined:
All quantized values of absolute value 15 and less are directly coded using a Huffman code. Always pairs of values (x,y) are coded. If quantized values of magnitude greater than 15 are found, ESC-codes are used to flag these values. If one or both values of a pair is not zero, one or two sign bits are appended to the Huffman code word.

```
hcod[|x|][|y|]          is the Huffman code table entry for values
                        x,y
hlen[|x|][|y|]          is the Huffman length table entry for values
                        x,y
max_table_entry         is the maximum table entry index. This is a
                        system constant (15, maximum number of
                        entries in a single table is 256)
signx                   sign of the 1st. value (0 if positive, 1 if
                        negative)
signy                   sign of the 2nd. value (0 if positive, 1 if
                        negative)

struct coded_word {
         codeword       hcod[|x|][|y|], length is hlen[|x|][|y|]
         linbitsx       If (x=max_table_entry) this constitutes an
                        ESC-code. In this case the length of this
                        field is linbits, else zero. The unsigned
                        integer contained in this field is added to
                        max_table_entry -1 to establish the absolute
                        value of the encoded data.
         signx          sign of x (transmitted only if x not equal
                        0)
         linbitsy       See linbitsx.
         signy          sign of y (transmitted only if y not equal
                        0)
         }
```

The ESCaped codes linbitsx or linbitsy are only used if a value greater or equal to the table maximum is actually flagged. They are never used if the selected table is one for blocks with a maximum quantized value equal or less than max_table_entry. The sign bits are transmitted only if the value of x with respect to y is different from zero.

For the higher end of the spectrum quadruples of values are coded using one of two special tables. Again magnitude values are coded using a Huffman code. One of the two codes is not really a 4-dimensional code because it is constructed from the trivial code: 0 is coded with a 1 (no sign bit needed), 1 is coded with a 0 (sign bit added). In both cases the Huffman code is assembled as follows:

```
struct quad_word {
         codeword       hcod[|v|][|w|][|x|][|y|],
                        hlen[|v|][|w|][|x|][|y|]
         signv          only if v not equal 0
         signw          only if w not equal 0
         signx          only if x not equal 0
         signy          only if y not equal 0
         }
```

At the high end of the spectrum the number of pairs of zeroes is simply counted. As this value is implicitly known when the other values have been decoded, it is not transmitted.

Ordering of Huffman encoded data:

If the block_type is 0, 1 or 3 the Huffman encoded data are ordered in terms of increasing frequency.

If the block_type is 2 (short blocks) then the Huffman encoded data are ordered in a pattern similar to that of the scalefactor values (see clause 2.4.2.7):

The Huffman encoded data are given for successive scalefactor bands, beginning with scalefactor band 0 and ending with scalefactor band 11. Within each scalefactor band, the data is given for successive time windows, beginning with window 0 and ending with window 2. The data values within each window are arranged in order of increasing frequency.

2.4.2.8 Ancillary data

`Ancillary_bit` - user definable

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 just after the syncword : the bitstream is subdivided in slots. The distance between the start of two consecutive syncwords is constant and equals "N" slots. The value of "N" depends on the Layer.
For Layer I the following equation is valid:

N = 12 * bit_rate / sampling_frequency.

For Layers II and III the equation becomes:

N = 144 * bit_rate / 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 clause 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:

$$G(X) = X16 + X15 + X2 + 1$$

The bits included into the CRC-check are:

- 16 bits of header(), starting with bit_rate_index and ending with emphasis
- a number of bits of audio_data(), starting with the first bit. This number is given by 3-AnnexB,Table 3-B.5 "NUMBER OF PROTECTED AUDIO_DATA BITS".

The method is depicted in 3-Annex A, Figure 3-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 3-Annex A, Figure 3-A.9 "CRC-CHECK DIAGRAM". The outputs b15...b0 constitute a word to be compared with the CRC-check word in the bitstream. If the words are not identical, a transmission error has occured 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 clause 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 3-Annex A, Figure 3-A.1 "LAYER I AND II DECODER FLOW CHART".

Requantization of subband samples

From the bit allocation the number of bits nb that has to be read for the samples in each subband is known. The order of the samples is given in clause 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"= * (s"' + 2-nb+1)

where:	s"'	is the fractional number,
	s"	the requantized value,
	nb	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 3-Annex B, Table 3-B.1 "LAYER I, II SCALEFACTORS". The rescaled value s' is calculated as :

s'= factor * s".

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 3-Annex A, Figure 3-A.2 "SYNTHESIS SUBBAND FILTER FLOW CHART" show the reconstruction operation. The coefficients Nik for the matrixing operation are given by

Nik = cos[(16 + i)(2k + 1)p/64], for i = 0 to 63, and k = 0 to 31.

The coefficients Di for the windowing operation can be found in 3-Annex B, Table 3-B.3 "COEFFICIENTS Di OF THE SYNTHESIS WINDOW". One frame contains 12 * 32 = 384 subband samples, which result, after filtering, in 384 audio samples.

2.4.3.3 Layer II

LayerII is a more efficient but more complex coding scheme than LayerI. The flowchart in 3-Annex A, Figure 3-A.1 "LAYER I AND II DECODER FLOW CHART" applies to both LayersI and II. The first step is to perform the decoding which is common to all three layers (see clause 2.4.3.1).

Bit allocation decoding

For different combinations of bitrate and sampling frequency different bit allocation tables exist (3-Annex B, Table3-B.2 "LAYER II BIT ALLOCATION TABLES"). 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 3-Annex B, Table 3-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 3-Annex B, Table 3-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'.

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.

Scalefactor decoding

For every subband with a nonzero bit allocation the coded scalefactor 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 3-Annex B, Table 3-B.1 "LAYER I, II SCALEFACTORS". This table gives the scalefactor by which the relevant subband samples should be multiplied after requantization.

Requantization of subband samples

Next the coded samples are read. As can be seen from clause 2.4.1.6, the coded samples appear as triplets, the code contains three consecutive samples at a time. From 3-Annex B, Table 3-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 3-Annex B, Table 3-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 3-Annex B, Table 3-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'' the requantized value.

The values of the constants C and D are given in 3-Annex B, Table 3-B.4 "LAYER II CLASSES OF QUANTIZATION". The requantized values have to be rescaled. The multiplication factors can be found in the 3-Annex B, Table 3-B.1 "LAYER I, II SCALEFACTORS". as described above. The rescaled value s' is calculated as:

$$s' = factor * s''.$$

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 3-Annex A, Figure 3-A.2 "SYNTHESIS SUBBAND FILTER FLOW CHART" have to be carried out. The coefficients Nik for the matrixing operation are given by

$$Nik = \cos[(16 + i)(2k + 1)p/64], \quad \text{for } i = 0 \text{ to } 63, \text{ and } k = 0 \text{ to } 31.$$

The coefficients Di for the windowing operation can be found in 3-Annex B, Table 3-B.3 "COEFFICIENTS Di 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 a MDCT. The window length of the MDCT is 36. Adaptive window switching is used to control time artifacts (pre-echoes), see the description in 3-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 'switch_point' 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 3-Annex 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 3-Annex C). The size of the input buffer is the size of one frame at the bitrate of 160 kbit/s per channel for LayerIII. 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.

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 3-Annex B.8).

Side information

Decoding of the side information requires storage of the decoded parameters. The table select information is used to select the decoder table and the number of ESC-bits (linbits), according to the table in the 3-Annex B-B.7.

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 Fig. 3-Annex A.7.1 and 3-Annex A.7.2. The begin of the main data part is located by using the main_data_end pointer of the preceding 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 position from the bit_rate_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 Fig. 3-Annex A.7.2).

MS_stereo mode

This mode switch (found in the header: mode_extension) allows switching from 'independent stereo' to MS_stereo. The upper bound of the scalefactor bands decoded in ms stereo is derived from the "zero_part" of the difference (right) channel. Above this bound intensity stereo can be applied if enabled in the header.
The "zero_part" of the difference channel is the part of the spectrum from "bigvalues*2+count1*4" (see clause 2.4.2.7) to the Nyquist rate.

- MS matrix

In MS stereo mode the values of the normalized middle/side channels Mi/Si are transmitted instead of the left/right channel values Li/Ri. Thus Li/Ri 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 Mi are transmitted in the left, values Si are transmitted in the right channel

Intensity stereo mode

This mode switch (found in the header: mode_extension) allows switching from 'normal stereo' to intensity stereo. 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 (left 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:

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

Scalefactors

The scalefactors are decoded according to the actual slen1 and slen2 which themselves are decoded from scalefac_select. 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.

Huffman decoding

Huffman decoding is done using a state machine. The state machine works from a ROM table, where each entry is the information for one node in the decoder tree. All necessary information including the table which realizes the Huffman code tree can be generated from the tables in 3-Annex B, Table 3-B.7. Decoding is done until all Huffmancodebits have been decoded or until quantized values representing 576 frequency lines have been decoded, whichever comes first. If there are more Huffmancodebits than necessary to decode 576 values they are regarded as stuffing bits and discarded.

Requantizer

The nonuniform quantizer uses a power law. For each output valueY from the Huffman decoder Y4/3 is calculated. This can be done either by table lookup or by explicit calculation.

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 3-Annex 3-A.8.

The following example is given for the example of a granule containing data with block_type==2 (short blocks). It can accordingly be used for other block types. 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):

$$xr(i) = is(i)^{4/3} * 2^{.25*(global\_gain[gr] - 64 - 8 * subblock\_gain[window][gr])} *$$
$$2^{.25*(-2 * (1+scalefac\_scale[gr]) * scalefac[cb][window][gr] - 2 * preflag[gr] * (1+scalefac\_scale[gr]) * pretab[cb])}$$

The constant 64 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. If an implementation with a different power transfer characteristic is chosen (different global scaling) then the constant has to be changed accordingly.

Synthesis filterbank

3-Annex A, Figure 3-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 in 3-Annex A Figure 3-A.5 and in 3-Annex B Table 3-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.

Buffer considerations

The following rule can be used to calculate the maximum number of bits used for one granule:
At the highest possible bitrate of Layer III (320 kbit/s per stereo signal) the frames must be of constant length, i.e. one buffer length is

320000 * .024 bit = 7680 bit.

This value is used as the maximum buffer per channel at the lower bitrates. At 64 kbit/s (128 kbit/s stereo) the mean granule length is 64000/48000*576 = 768 bit at 48 kHz sampling frequency. This means that there is a maximum deviation (short time buffer) of 7680 - 4*768 = 4608 bits is allowed at 64 kbit/s. The actual deviation is equal to the number of bytes denoted by the main_data_end offset pointer. The actual maximum deviation is 2**9 * 8 bit = 4096 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_end is always set to 0 in the case of bit_rate_index==14, i.e. data rate 320 kbps per stereo signal. In this case all data are allocated between adjacent header words.

IMDCT

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

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

Windowing

Depending on the block_type different shapes of windows are used.

a) block_type=0

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

b) block_type=1

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

c) block_type=3

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

d) block_type=2:

Each of the three small blocks is windowed separately.

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

The windowed small blocks must be overlapped and concatenated.

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

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:

$$x_i = y_i + s_i \qquad \text{for i=0 to 17}$$
$$s_i = y_{i+18} \qquad \text{for i=0 to 17}$$