INTERNATIONAL STANDARD

ISO/IEC 11172-2

First edition
1993-08-01

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

Part 2:
Video

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

Partie 2: Vidéo

Reference number
ISO/IEC 11172-2:1993(E)

Contents

Page

Foreword

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

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

International Standard ISO/IEC 11172-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Sub-Committee SC 29, *Coded representation of audio, picture, multimedia and hypermedia information.*

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

— *Part 1: Systems*

— *Part 2: Video*

— *Part 3: Audio*

— *Part 4: Compliance testing*

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

Introduction

Note -- Readers interested in an overview of the MPEG Video layer should read this Introduction and then proceed to annex D, before returning to clauses 1 and 2.

0.1 Purpose

This part of ISO/IEC 11172 was developed in response to the growing need for a common format for representing compressed video on various digital storage media such as CDs, DATs, Winchester disks and optical drives. This part of ISO/IEC 11172 specifies a coded representation that can be used for compressing video sequences to bitrates around 1,5 Mbit/s. The use of this part of ISO/IEC 11172 means that motion video can be manipulated as a form of computer data and can be transmitted and received over existing and future networks. The coded representation can be used with both 625-line and 525-line television and provides flexibility for use with workstation and personal computer displays.

This part of ISO/IEC 11172 was developed to operate principally from storage media offering a continuous transfer rate of about 1,5 Mbit/s. Nevertheless it can be used more widely than this because the approach taken is generic.

0.1.1 Coding parameters

The intention in developing this part of ISO/IEC 11172 has been to define a source coding algorithm with a large degree of flexibility that can be used in many different applications. To achieve this goal, a number of the parameters defining the characteristics of coded bitstreams and decoders are contained in the bitstream itself. This allows for example, the algorithm to be used for pictures with a variety of sizes and aspect ratios and on channels or devices operating at a wide range of bitrates.

Because of the large range of the characteristics of bitstreams that can be represented by this part of ISO/IEC 11172, a sub-set of these coding parameters known as the "Constrained Parameters" has been defined. The aim in defining the constrained parameters is to offer guidance about a widely useful range of parameters. Conforming to this set of constraints is not a requirement of this part of ISO/IEC 11172. A flag in the bitstream indicates whether or not it is a Constrained Parameters bitstream.

Summary of the Constrained Parameters:

Horizontal picture size	Less than or equal to 768 pels
Vertical picture size	Less than or equal to 576 lines
Picture area	Less than or equal to 396 macroblocks
Pel rate	Less than or equal to 396x25 macroblocks/s
Picture rate	Less than or equal to 30 Hz
Motion vector range	Less than -64 to +63,5 pels (using half-pel vectors) [backward_f_code and forward_f_code <= 4 (see table D.7)]
Input buffer size (in VBV model)	Less than or equal to 327 680 bits
Bitrate	Less than or equal to 1 856 000 bits/s (constant bitrate)

0.2 Overview of the algorithm

The coded representation defined in this part of ISO/IEC 11172 achieves a high compression ratio while preserving good picture quality. The algorithm is not lossless as the exact pel values are not preserved during coding. The choice of the techniques is based on the need to balance a high picture quality and compression ratio with the requirement to make random access to the coded bitstream. Obtaining good picture quality at the bitrates of interest demands a very high compression ratio, which is not achievable with intraframe coding alone. The need for random access, however, is best satisfied with pure intraframe coding. This requires a careful balance between intra- and interframe coding and between recursive and non-recursive temporal redundancy reduction.

A number of techniques are used to achieve a high compression ratio. The first, which is almost independent from this part of ISO/IEC 11172, is to select an appropriate spatial resolution for the signal. The algorithm then uses block-based motion compensation to reduce the temporal redundancy. Motion compensation is used for causal prediction of the current picture from a previous picture, for non-causal prediction of the current picture from a future picture, or for interpolative prediction from past and future pictures. Motion vectors are defined for each 16-pel by 16-line region of the picture. The difference signal, the prediction error, is further compressed using the discrete cosine transform (DCT) to remove spatial correlation before it is quantized in an irreversible process that discards the less important information. Finally, the motion vectors are combined with the DCT information, and coded using variable length codes.

0.2.1 Temporal processing

Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intra-coded pictures (I-Pictures) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only a moderate compression ratio. Predictive coded pictures (P-Pictures) are coded more efficiently using motion compensated prediction from a past intra or predictive coded picture and are generally used as a reference for further prediction. Bidirectionally-predictive coded pictures (B-Pictures) provide the highest degree of compression but require both past and future reference pictures for motion compensation. Bidirectionally-predictive coded pictures are never used as references for prediction. The organisation of the three picture types in a sequence is very flexible. The choice is left to the encoder and will depend on the requirements of the application. Figure 1 illustrates the relationship between the three different picture types.

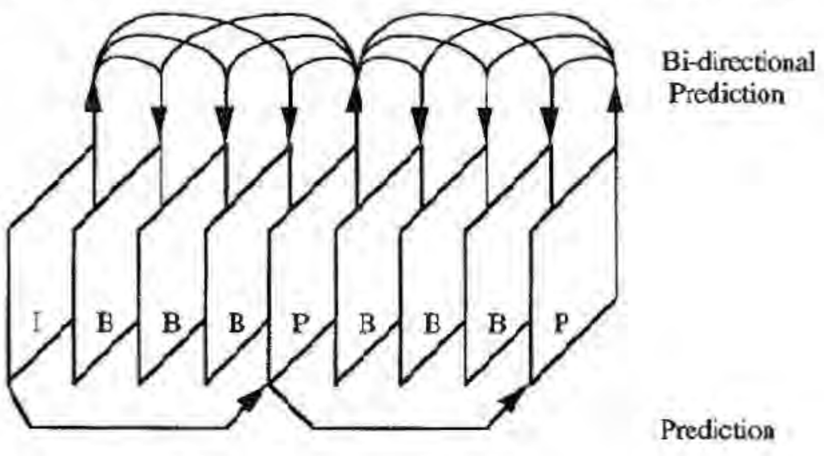


Figure 1 -- Example of temporal picture structure

The fourth picture type defined in this part of ISO/IEC 11172, the D-picture, is provided to allow a simple, but limited quality, fast-forward playback mode.

0.2.2 Motion representation - macroblocks

The choice of 16 by 16 macroblocks for the motion-compensation unit is a result of the trade-off between increasing the coding efficiency provided by using motion information and the overhead needed to store it. Each macroblock can be one of a number of different types. For example, intra-coded, forward-predictive-coded, backward-predictive coded, and bidirectionally-predictive-coded macroblocks are permitted in bidirectionally-predictive coded pictures. Depending on the type of the macroblock, motion vector information and other side information are stored with the compressed prediction error signal in each macroblock. The motion vectors are encoded differentially with respect to the last coded motion vector, using variable-length codes. The maximum length of the vectors that may be represented can be programmed, on a picture-by-picture basis, so that the most demanding applications can be met without compromising the performance of the system in more normal situations.

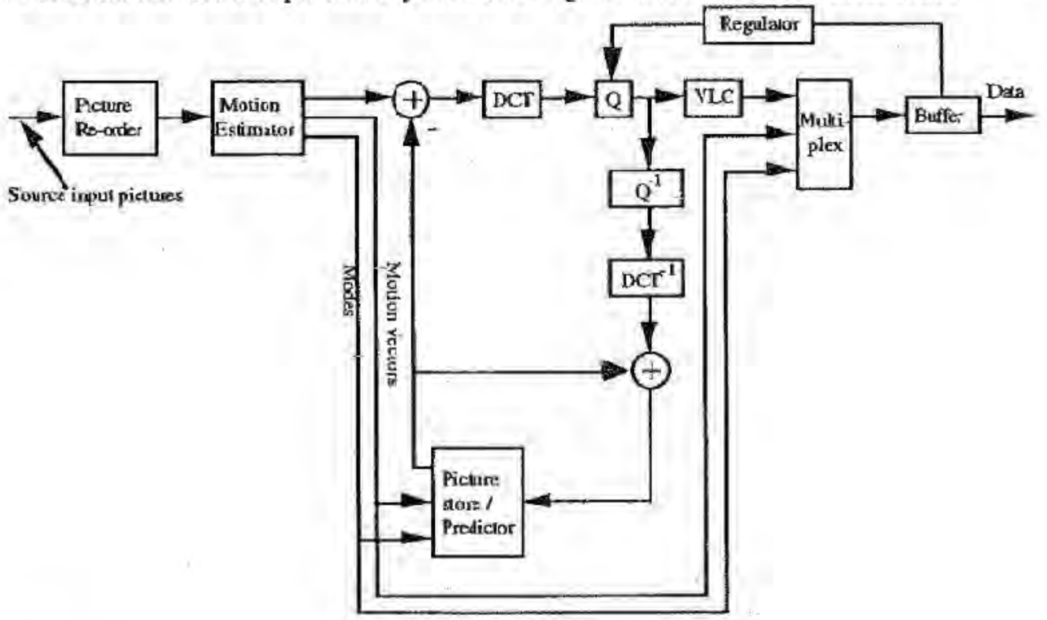
It is the responsibility of the encoder to calculate appropriate motion vectors. This part of ISO/IEC 11172 does not specify how this should be done.

0.2.3 Spatial redundancy reduction

Both original pictures and prediction error signals have high spatial redundancy. This part of ISO/IEC 11172 uses a block-based DCT method with visually weighted quantization and run-length coding. Each 8 by 8 block of the original picture for intra-coded macroblocks or of the prediction error for predictive-coded macroblocks is transformed into the DCT domain where it is scaled before being quantized. After quantization many of the coefficients are zero in value and so two-dimensional run-length and variable length coding is used to encode the remaining coefficients efficiently.

0.3 Encoding

This part of ISO/IEC 11172 does not specify an encoding process. It specifies the syntax and semantics of the bitstream and the signal processing in the decoder. As a result, many options are left open to encoders to trade-off cost and speed against picture quality and coding efficiency. This clause is a brief description of the functions that need to be performed by an encoder. Figure 2 shows the main functional blocks.



where

DCT is discrete cosine transform
DCT^{-1} is inverse discrete cosine transform
Q is quantization
Q^{-1} is dequantization
VLC is variable length coding

Figure 2 -- Simplified video encoder block diagram

The input video signal must be digitized and represented as a luminance and two colour difference signals (Y, C_b, C_r). This may be followed by preprocessing and format conversion to select an appropriate window, resolution and input format. This part of ISO/IEC 11172 requires that the colour difference signals (C_b and C_r) are subsampled with respect to the luminance by 2:1 in both vertical and horizontal directions and are reformatted, if necessary, as a non-interlaced signal.

The encoder must choose which picture type to use for each picture. Having defined the picture types, the encoder estimates motion vectors for each 16 by 16 macroblock in the picture. In P-Pictures one vector is needed for each non-intra macroblock and in B-Pictures one or two vectors are needed.

If B-Pictures are used, some reordering of the picture sequence is necessary before encoding. Because B-Pictures are coded using bidirectional motion compensated prediction, they can only be decoded after the subsequent reference picture (an I or P-Picture) has been decoded. Therefore the pictures are reordered by the encoder so that the pictures arrive at the decoder in the order for decoding. The correct display order is recovered by the decoder.

The basic unit of coding within a picture is the macroblock. Within each picture, macroblocks are encoded in sequence, left to right, top to bottom. Each macroblock consists of six 8 by 8 blocks: four blocks of luminance, one block of Cb chrominance, and one block of Cr chrominance. See figure 3. Note that the picture area covered by the four blocks of luminance is the same as the area covered by each of the chrominance blocks. This is due to subsampling of the chrominance information to match the sensitivity of the human visual system.

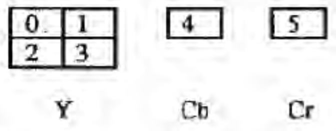


Figure 3 -- Macroblock structure

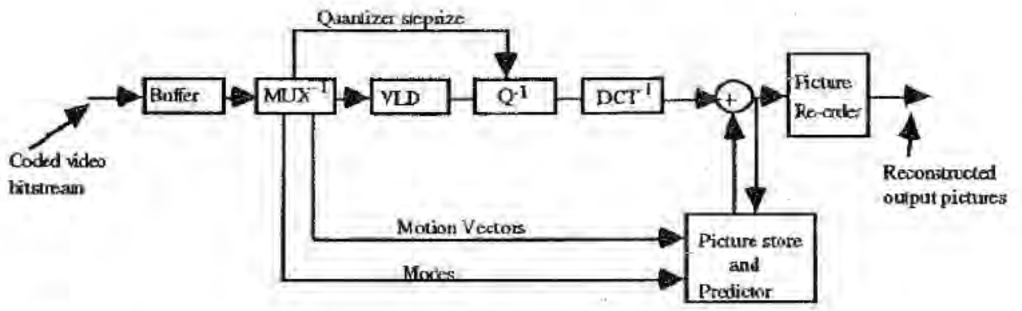
Firstly, for a given macroblock, the coding mode is chosen. It depends on the picture type, the effectiveness of motion compensated prediction in that local region, and the nature of the signal within the block. Secondly, depending on the coding mode, a motion compensated prediction of the contents of the block based on past and/or future reference pictures is formed. This prediction is subtracted from the actual data in the current macroblock to form an error signal. Thirdly, this error signal is separated into 8 by 8 blocks (4 luminance and 2 chrominance blocks in each macroblock) and a discrete cosine transform is performed on each block. Each resulting 8 by 8 block of DCT coefficients is quantized and the two-dimensional block is scanned in a zig-zag order to convert it into a one-dimensional string of quantized DCT coefficients. Fourthly, the side-information for the macroblock (mode, motion vectors etc) and the quantized coefficient data are encoded. For maximum efficiency, a number of variable length code tables are defined for the different data elements. Run-length coding is used for the quantized coefficient data.

A consequence of using different picture types and variable length coding is that the overall data rate is variable. In applications that involve a fixed-rate channel, a FIFO buffer may be used to match the encoder output to the channel. The status of this buffer may be monitored to control the number of bits generated by the encoder. Controlling the quantization process is the most direct way of controlling the bitrate. This part of ISO/IEC 11172 specifies an abstract model of the buffering system (the Video Buffering Verifier) in order to constrain the maximum variability in the number of bits that are used for a given picture. This ensures that a bitstream can be decoded with a buffer of known size.

At this stage, the coded representation of the picture has been generated. The final step in the encoder is to regenerate I-Pictures and P-Pictures by decoding the data so that they can be used as reference pictures for subsequent encoding. The quantized coefficients are dequantized and an inverse 8 by 8 DCT is performed on each block. The prediction error signal produced is then added back to the prediction signal and limited to the required range to give a decoded reference picture.

0.4 Decoding

Decoding is the inverse of the encoding operation. It is considerably simpler than encoding as there is no need to perform motion estimation and there are many fewer options. The decoding process is defined by this part of ISO/IEC 11172. The description that follows is a very brief overview of one possible way of decoding a bitstream. Other decoders with different architectures are possible. Figure 4 shows the main functional blocks.



Where

DCT^{-1}	is inverse discrete cosine transform
Q^{-1}	is dequantization
MUX^{-1}	is demultiplexing
VLD	is variable length decoding

Figure 4 -- Basic video decoder block diagram

For fixed-rate applications, the channel fills a FIFO buffer at a constant rate with the coded bitstream. The decoder reads this buffer and decodes the data elements in the bitstream according to the defined syntax.

As the decoder reads the bitstream, it identifies the start of a coded picture and then the type of the picture. It decodes each macroblock in the picture in turn. The macroblock type and the motion vectors, if present, are used to construct a prediction of the current macroblock based on past and future reference pictures that have been stored in the decoder. The coefficient data are decoded and dequantized. Each 8 by 8 block of coefficient data is transformed by an inverse DCT (specified in annex A), and the result is added to the prediction signal and limited to the defined range.

After all the macroblocks in the picture have been processed, the picture has been reconstructed. If it is an I-picture or a P-picture it is a reference picture for subsequent pictures and is stored, replacing the oldest stored reference picture. Before the pictures are displayed they may need to be re-ordered from the coded order to their natural display order. After reordering, the pictures are available, in digital form, for post-processing and display in any manner that the application chooses.

0.5 Structure of the coded video bitstream

This part of ISO/IEC 11172 specifies a syntax for a coded video bitstream. This syntax contains six layers, each of which either supports a signal processing or a system function:

Layers of the syntax	Function
Sequence layer	Random access unit: context
Group of pictures layer	Random access unit: video
Picture layer	Primary coding unit
Slice layer	Resynchronization unit
Macroblock layer	Motion compensation unit
Block layer	DCT unit

0.6 Features supported by the algorithm

Applications using compressed video on digital storage media need to be able to perform a number of operations in addition to normal forward playback of the sequence. The coded bitstream has been designed to support a number of these operations.

0.6.1 Random access

Random access is an essential feature for video on a storage medium. Random access requires that any picture can be decoded in a limited amount of time. It implies the existence of access points in the bitstream - that is segments of information that are identifiable and can be decoded without reference to other segments of data. A spacing of two random access points (Intra-Pictures) per second can be achieved without significant loss of picture quality.

0.6.2 Fast search

Depending on the storage medium, it is possible to scan the access points in a coded bitstream (with the help of an application-specific directory or other knowledge beyond the scope of this part of ISO/IEC 11172) to obtain a fast-forward and fast-reverse playback effect.

0.6.3 Reverse playback

Some applications may require the video signal to be played in reverse order. This can be achieved in a decoder by using memory to store entire groups of pictures after they have been decoded before being displayed in reverse order. An encoder can make this feature easier by reducing the length of groups of pictures.

0.6.4 Error robustness

Most digital storage media and communication channels are not error-free. Appropriate channel coding schemes should be used and are beyond the scope of this part of ISO/IEC 11172. Nevertheless the compression scheme defined in this part of ISO/IEC 11172 is robust to residual errors. The slice structure allows a decoder to recover after a data error and to resynchronize its decoding. Therefore, bit errors in the compressed data will cause errors in the decoded pictures to be limited in area. Decoders may be able to use concealment strategies to disguise these errors.

0.6.5 Editing

There is a conflict between the requirement for high coding efficiency and easy editing. The coding structure and syntax have not been designed with the primary aim of simplifying editing at any picture. Nevertheless a number of features have been included that enable editing of coded data.

This page intentionally left blank

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

Part 2: Video

Section 1: General

1.1 Scope

This part of ISO/IEC 11172 specifies the coded representation of video for digital storage media and specifies the decoding process. The representation supports normal speed forward playback, as well as special functions such as random access, fast forward playback, fast reverse playback, normal speed reverse playback, pause and still pictures. This part of ISO/IEC 11172 is compatible with standard 525- and 625-line television formats, and it provides flexibility for use with personal computer and workstation displays.

ISO/IEC 11172 is primarily applicable to digital storage media supporting a continuous transfer rate up to about 1,5 Mbit/s, such as Compact Disc, Digital Audio Tape, and magnetic hard disks. Nevertheless it can be used more widely than this because of the generic approach taken. The storage media may be directly connected to the decoder, or via communications means such as busses, LANs, or telecommunications links. This part of ISO/IEC 11172 is intended for non-interlaced video formats having approximately 288 lines of 352 pels and picture rates around 24 Hz to 30 Hz.

1.2 Normative references

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

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

ISO/IEC 11172-3:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 3 Audio.*

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

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

CCIR Recommendation 648 *Recording of audio signals.*

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

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

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

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

Section 2: Technical elements

2.1 Definitions

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

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

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

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

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

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

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

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

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

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

2.1.10 audio sequence [audio]: A non-interrupted series of audio frames in which the following parameters are not changed:
- ID
- Layer
- Sampling Frequency
- For Layer I and II: Bitrate index

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

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

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

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

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

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

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

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

2.1.19 byte: Sequence of 8-bits.

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

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

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

2.1.23 coded audio bitstream [audio]: A coded representation of an audio signal as specified in ISO/IEC 11172-3.

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

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

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

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

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

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

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

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

2.1.32 constrained parameters [video]: The values of the set of coding parameters defined in 2.4.3.2.

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

2.1.34 CRC: Cyclic redundancy code.

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

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

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

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

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

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

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

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

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

2.1.44 decoder: An embodiment of a decoding process.

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

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

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

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

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

2.1.50 discrete cosine transform; DCT [video]: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in annex A.

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

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

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

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

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

2.1.56 encoder: An embodiment of an encoding process.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.73 Huffman coding: A specific method for entropy coding.

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

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

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

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

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

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

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

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

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

2.1.83 layer [audio]: One of the levels in the coding hierarchy of the audio system defined in ISO/IEC 11172-3.

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

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

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

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

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

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

2.1.90 MDCT [audio]: Modified Discrete Cosine Transform.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.105 pel [video]: Picture element.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.141 subband filterbank [audio]: A set of band filters covering the entire audio frequency range. In ISO/IEC 11172-3 the subband filterbank is a polyphase filterbank.

2.1.142 subband samples [audio]: The subband filterbank within the audio encoder creates a filtered and subsampled representation of the input audio stream. The filtered samples are called subband samples. From 384 time-consecutive input audio samples, 12 time-consecutive subband samples are generated within each of the 32 subbands.

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

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

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

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

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

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

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

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

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

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

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

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

2.2 Symbols and abbreviations

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

2.2.1 Arithmetic operators

+	Addition.
-	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
--	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, 7/4 and -7/-4 are truncated to 1 and -7/4 and 7/-4 are truncated to -1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and -3//2 is rounded to -2.
DIV	Integer division with truncation of the result towards $-\infty$.
| |	Absolute value. $\|x\| = x$ when $x > 0$; $\|x\| = 0$ when $x == 0$; $\|x\| = -x$ when $x < 0$
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign(x) = 1 when $x > 0$; 0 when $x == 0$; -1 when $x < 0$
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.
$\log_{10}$	Logarithm to base ten.
$\log_e$	Logarithm to base e.
$\log_2$	Logarithm to base 2.

2.2.2 Logical operators

||	Logical OR.
&&	Logical AND.

!	Logical NOT.

2.2.3 Relational operators

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
==	Equal to.
!=	Not equal to.
max [,...,]	the maximum value in the argument list.
min [,...,]	the minimum value in the argument list.

2.2.4 Bitwise operators

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

&	AND.
|	OR.
>>	Shift right with sign extension.
<<	Shift left with zero fill.

2.2.5 Assignment

=	Assignment operator.

2.2.6 Mnemonics

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

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

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

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

2.2.7 Constants

π	3,14159265358...
e	2,71828182845...

2.3 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in 2.4.2. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in 2.4.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:

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

while (condition) {
  data_element
    . . .
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do {
  data_element
	The data element always occurs at least once.

. . . } while (condition)	The data element is repeated until the condition is not true.

if (condition) {
  data_element
    . . .
}
	If the condition is true, then the first group of data elements occurs next in the data stream.

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

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

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

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

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

data_element []	data_element [] is an array of data. The number of data elements is indicated by the context.
data_element [n]	data_element [n] is the n+1th element of an array of data.
data_element [m][n]	data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.
data_element [l][m][n]	data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.
data_element [m..n]	is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that 2.4.3 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include a means to look for start codes in order to begin decoding correctly, 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 bitstream is the first bit in a byte. Otherwise it returns 0.

Definition of nextbits function

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

Definition of next_start_code function

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

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

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

2.4 Requirements

2.4.1 Coding structure and parameters

Video sequence

A coded video sequence commences with a sequence header and is followed by one or more groups of pictures and is ended by a sequence_end_code. Immediately before each of the groups of pictures there may be a sequence header. Within each sequence, pictures shall be decodable continuously.

In each of these repeated sequence headers all of the data elements with the permitted exception of those defining the quantization matrices (load_intra_quantizer_matrix, load_non_intra_quantizer_matrix and optionally intra_quantizer_matrix and non_intra_quantizer_matrix) shall have the same values as in the first sequence header. The quantization matrices may be redefined each time that a sequence header occurs in the bitstream. Thus the data elements load_intra_quantizer_matrix, load_non_intra_quantizer_matrix and optionally intra_quantizer_matrix and non_intra_quantizer_matrix may have any (non-forbidden) values.

Repeating the sequence header allows the data elements of the initial sequence header to be repeated in order that random access into the video sequence is possible. In addition the quantization matrices may be changed inside the video sequence as required.

Sequence header

A video sequence header commences with a sequence_header_code and is followed by a series of data elements.

Group of pictures

A **group of pictures** is a series of one or more coded pictures intended to assist random access into the sequence. In the stored bitstream, the first coded picture in a group of pictures is an I-Picture. The order of the pictures in the coded steam is the order in which the decoder processes them in normal playback. In particular, adjacent B-Pictures in the coded stream are in display order. The last coded picture, in display order, of a group of pictures is either an I-Picture or a P-Picture.

The following is an example of groups of pictures taken from the beginning of a video sequence. In this example the first group of pictures contains seven pictures and subsequent groups of pictures contain nine pictures. There are two B-pictures between successive P-pictures and also two B-pictures between successive I- and P-pictures. Picture '1I' is used to form a prediction for picture '4P'. Pictures '4P' and '1I' are both used to form predictions for pictures '2B' and '3B'. Therefore the order of pictures in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder should display them in the order '1I', '2B', '3B', '4P'.

At the encoder input,

```
‖ 1  2  3  4  5  6  7 ‖ 8  9  10 11 12 13 14 15 16 ‖ 17 18 19 20 21 22 23 24 25
‖ I  B  B  P  B  B  P ‖ B  B  I  B  B  P  B  B  P  ‖ B  B  I  B  B  P  B  B  P
```

At the encoder output, in the stored bitstream, and at the decoder input,

```
‖ 1  4  2  3  7  5  6 ‖ 10 8  9  13 11 12 16 14 15 ‖ 19 17 18 22 20 21 25 23 24
‖ I  P  B  B  P  B  B ‖ I  B  B  P  B  B  P  B  B  ‖ I  B  B  P  B  B  P  B  B
```

where the double vertical bars mark the group of pictures boundaries. Note that in this example, the first group of pictures is two pictures shorter than subsequent groups of pictures, since at the beginning of video coding there are no B-pictures preceding the first I-Picture. However, in general, in display order, there may be B-Pictures preceding the first I-Picture in the group of pictures, even for the first group of pictures to be decoded.

At the decoder output,

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

A group of pictures may be of any length. A group of pictures shall contain one or more I-Pictures. Applications requiring random access, fast-forward playback, or fast and normal reverse playback may use relatively short groups of pictures. Groups of pictures may also be started at scene cuts or other cases where motion compensation is ineffective.

The number of consecutive B-Pictures is variable. Neither B- nor P-Pictures need be present.

A video sequence of groups of pictures that is read by the decoder may be different from the one at the encoder output due to editing.

Picture

A source or reconstructed picture consists of three rectangular matrices of eight-bit numbers; a luminance matrix (Y), and two chrominance matrices (Cb and Cr). The Y-matrix shall have an even number of rows and columns, and the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions.

The Y, Cb and Cr components are related to the primary (analogue) Red, Green and Blue Signals (E'_R, E'_G and E'_B) as described in CCIR Recommendation 601. These primary signals are gamma pre-corrected. The assumed value of gamma is not defined in this part of ISO/IEC 11172 but may typically be in the region approximately 2,2 to approximately 2,8. Applications which require accurate colour reproduction may choose to specify the value of gamma more accurately, but this is outside the scope of this part of ISO/IEC 11172.

The luminance and chrominance samples are positioned as shown in figure 5, where "x" marks the position of the luminance (Y) samples and "0" marks the position of the chrominance (Cb and Cr) samples:

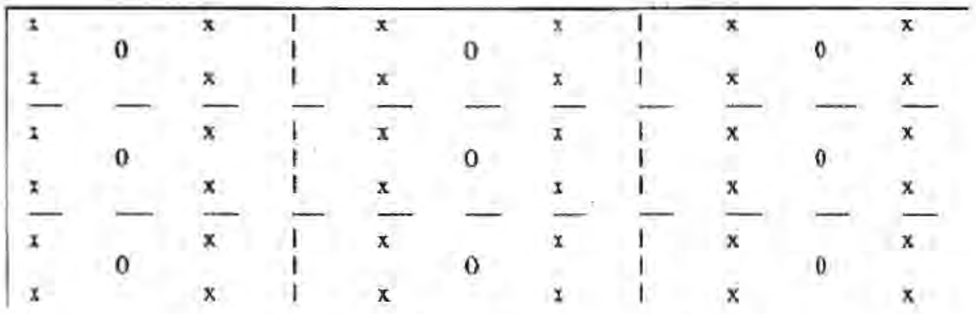


Figure 5 -- The position of luminance and chrominance samples.

There are four types of coded picture that use different coding methods.

An **Intra-coded picture (I-picture)** is coded using information only from itself.

A **Predictive-coded picture (P-picture)** is a picture which is coded using motion compensated prediction from a past I-Picture or P-Picture.

A **Bidirectionally predictive-coded picture (B-picture)** is a picture which is coded using motion compensated prediction from a past and/or future I-Picture or P-Picture.

A **dc coded (D) picture** is coded using information only from itself. Of the DCT coefficients only the dc ones are present. The D-Pictures shall not be in a sequence containing any other picture types.

Slice

A **slice** is a series of an arbitrary number of macroblocks with the order of macroblocks starting from the upper-left of the picture and proceeding by raster-scan order from left to right and top to bottom. The first and last macroblocks of a slice shall not be skipped macroblocks (see 2.4.4.4). Every slice shall contain at least one macroblock. Slices shall not overlap and there shall be no gaps between slices. The position of slices may change from picture to picture. The first slice shall start with the first macroblock in the picture and the last slice shall end with the last macroblock in the picture.

Macroblock

A **macroblock** contains a 16-pel by 16-line section of luminance component and the spatially corresponding 8-pel by 8-line section of each chrominance component. A macroblock has 4 luminance blocks and 2 chrominance blocks. The term "macroblock" can refer to source or reconstructed data or to scaled, quantized coefficients. The order of blocks in a macroblock is top-left, top-right, bottom-left, bottom-right blocks for Y, followed by Cb and Cr. Figure 6 shows the arrangement of these blocks. A skipped macroblock is one for which no information is stored (see 2.4.4.4).

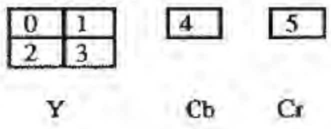


Figure 6 -- The arrangement of blocks in a macroblock.

Block

A **block** is an orthogonal 8-pel by 8-line section of a luminance or chrominance component.

The term "block" can refer either to source and reconstructed data or to the corresponding coded data elements.

Reserved, Forbidden and Marker bit

The terms "reserved" and "forbidden" are used in the description of some values of several fields in the coded bitstream.

The term "reserved" indicates that the value may be used in the future for ISO/IEC-defined extensions.

The term "forbidden" indicates a value that shall never be used (usually in order to avoid emulation of start codes).

The term "marker_bit" indicates a one bit field in which the value zero is forbidden. These marker bits are introduced at several points in the syntax to avoid start-code emulation.

2.4.2 Specification of the coded video bitstream syntax

2.4.2.1 Start codes

Start codes are reserved bit patterns that do not otherwise occur in the video stream. All start codes are bytealigned.

Name	Hexadecimal value
picture_start_code	00000100
slice_start_codes (including slice_vertical_positions)	00000101 through 000001AF
reserved	000001B0
reserved	000001B1
user_data_start_code	000001B2
sequence_header_code	000001B3
sequence_error_code	000001B4
extension_start_code	000001B5
reserved	000001B6
sequence_end_code	000001B7
group_start_code	000001B8
system start codes (see note)	000001B9 through 000001FF

NOTE - System start codes are defined in ISO/IEC 11172-1.

The use of the start codes is defined in the following syntax description with the exception of the sequence_error_code. The sequence_error_code has been allocated for use by the digital storage media interface to indicate where uncorrectable errors have been detected.

2.4.2.2 Video sequence layer

Syntax	No. of bits	Mnemonic
video_sequence() {		
next_start_code()		
do {		
sequence_header()		
do {		
group_of_pictures()		
} while (nextbits() == group_start_code)		
} while (nextbits() == sequence_header_code)		
sequence_end_code	**32**	**bslbf**
}		

2.4.2.3 Sequence header

Syntax	No.of bits	Mnemonic
sequence_header() {		
sequence_header_code	32	bslbf
horizontal_size	12	uimsbf
vertical_size	12	uimsbf
pel_aspect_ratio	4	uimsbf
picture_rate	4	uimsbf
bit_rate	18	uimsbf
marker_bit	1	"1"
vbv_buffer_size	10	uimsbf
constrained_parameters_flag	1	
load_intra_quantizer_matrix	1	
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix []	8*64	uimsbf
load_non_intra_quantizer_matrix	1	
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix []	8*64	uimsbf
next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
while (nextbits () != '0000 0000 0000 0000 0000 0001') {		
sequence_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
}		

2.4.2.4 Group of pictures layer

Syntax	No. of bits	Mnemonic
group_of_pictures() {		
group_start_code	32	bslbf
time_code	25	
closed_gop	1	
broken_link	1	
next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
group_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
do {		
picture()		
} while (nextbits() == picture_start_code)		
}		

2.4.2.5 Picture layer

Syntax	No. of bits	Mnemonic
picture() {		
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if ((picture_coding_type == 2) || (picture_coding_type == 3)) {		
full_pel_forward_vector	1	
forward_f_code	3	uimsbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	
backward_f_code	3	uimsbf
}		
while (nextbits() == '1') {		
extra_bit_picture	1	"1"
extra_information_picture	8	
}		
extra_bit_picture	1	"0"
next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
picture_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
do {		
slice()		
} while (nextbits() == slice_start_code)		
}		

2.4.2.6 Slice layer

Syntax	No. of bits	Mnemonic
slice() {		
slice_start_code	**32**	**bslbf**
quantizer_scale	**5**	**uimsbf**
while (nextbits() == '1') {		
extra_bit_slice	**1**	**"1"**
extra_information_slice	**8**	
}		
extra_bit_slice	**1**	**"0"**
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

2.4.2.7 Macroblock layer

Syntax	No. of bits	Mnemonic
macroblock() {		
while (nextbits() == '0000 0001 111')		
macroblock_stuffing	**11**	**vlclbf**
while (nextbits() == '0000 0001 000')		
macroblock_escape	**11**	**vlclbf**
macroblock_address_increment	**1-11**	**vlclbf**
macroblock_type	**1-6**	**vlclbf**
if (macroblock_quant)		
quantizer_scale	**5**	**uimsbf**
if (macroblock_motion_forward) {		
motion_horizontal_forward_code	**1-11**	**vlclbf**
if ((forward_f != 1) &&		
(motion_horizontal_forward_code != 0))		
motion_horizontal_forward_r	**1-6**	**uimsbf**
motion_vertical_forward_code	**1-11**	**vlclbf**
if ((forward_f != 1) &&		
(motion_vertical_forward_code != 0))		
motion_vertical_forward_r	**1-6**	**uimsbf**
}		
if (macroblock_motion_backward) {		
motion_horizontal_backward_code	**1-11**	**vlclbf**
if ((backward_f != 1) &&		
(motion_horizontal_backward_code != 0))		
motion_horizontal_backward_r	**1-6**	**uimsbf**
motion_vertical_backward_code	**1-11**	**vlclbf**
if ((backward_f != 1) &&		
(motion_vertical_backward_code != 0))		
motion_vertical_backward_r	**1-6**	**uimsbf**
}		
if (macroblock_pattern)		
coded_block_pattern	**3-9**	**vlclbf**
for (i=0; i<6; i++)		
block(i)		
if (picture_coding_type == 4)		
end_of_macroblock	**1**	**"1"**
}		

2.4.2.8 Block layer

```
Syntax                                                  No. of bits   Mnemonic
block( i ) {

    if ( pattern_code[i] ) {
        if ( macroblock_intra ) {
            if ( i<4 ) {
                dct_dc_size_luminance                       2-7        vlclbf
                if(dc_size_luminance != 0)
                    dct_dc_differential                     1-8        uimsbf
            }
            else {
                dct_dc_size_chrominance                     2-8        vlclbf
                if(dc_size_chrominance !=0)
                    dct_dc_differential                     1-8        uimsbf
            }
        }
        else {
            dct_coeff_first                                 2-28       vlclbf
        }
        if ( picture_coding_type != 4 ) {
            while ( nextbits() != '10')
                dct_coeff_next                              3-28       vlclbf
            end_of_block                                      2        vlclbf
        }
    }
}
```

2.4.3 Semantics for the video bitstream syntax

2.4.3.1 Video sequence layer

sequence_end_code -- The sequence_end_code is the bit string 000001B7 in hexadecimal. It terminates a video sequence.

2.4.3.2 Sequence header

sequence_header_code -- The sequence_header_code is the bit string 000001B3 in hexadecimal. It identifies the beginning of a sequence header.

horizontal_size -- The horizontal_size is the width of the displayable part of the luminance component in pels. The width of the encoded luminance component in macroblocks, mb_width, is (horizontal_size+15)/16. The displayable part of the picture is left-aligned in the encoded picture.

vertical_size -- The vertical_size is the height of the displayable part of the luminance component in pels. The height of the encoded luminance component in macroblocks, mb_height, is (vertical_size+15)/16. The displayable part of the picture is top-aligned in the encoded picture.

pel_aspect_ratio -- This is a four-bit integer defined in the following table.

pel_aspect_ratio	height/width	example
0000	forbidden	
0001	1,0000	VGA etc.
0010	0,6735	
0011	0,7031	16:9, 625line
0100	0,7615	
0101	0,8055	
0110	0,8437	16:9, 525line
0111	0,8935	
1000	0,9157	CCIR601, 625line
1001	0,9815	
1010	1,0255	
1011	1,0695	
1100	1,0950	CCIR601, 525line
1101	1,1575	
1110	1,2015	
1111	reserved	

picture_rate -- This is a four-bit integer defined in the following table.

picture_rate	pictures per second
0000	forbidden
0001	23,976
0010	24
0011	25
0100	29,97
0101	30
0110	50
0111	59,94
1000	60
. . .	reserved
1111	reserved

Applications and encoders should take into account the fact that 23,976, 29,97 and 59,94 are not exact representations of the nominal picture rate. The exact values are found from 24 000/1 001, 30 000/1 001, and 60 000/1 001 and can be derived from CCIR Report 624-4.

bit_rate -- This is an integer specifying the bitrate of the bitstream measured in units of 400 bits/s, rounded upwards. The value zero is forbidden. The value 3FFFF (11 1111 1111 1111 1111) identifies variable bit rate operation.

marker_bit -- This is one bit that shall be set to "1".

vbv_buffer_size -- This is a 10-bit integer defining the size of the VBV (Video Buffering Verifier, see annex C) buffer needed to decode the sequence. It is defined as:

$$B = 16 * 1\,024 * vbv\_buffer\_size$$

where B is the minimum VBV buffer size in bits required to decode the sequence (see annex C).

constrained_parameters_flag -- This is a one-bit flag which may be set to "1" if the following data elements meet the following constraints:

horizontal_size <= 768 pels,
vertical_size <= 576 pels,
((horizontal_size+15)/16) *((vertical_size+15)/16) <= 396,
((horizontal_size+15)/16) *((vertical_size+15)/16))*picture_rate <= 396*25,
picture_rate <= 30 pictures/s,
forward_f_code <= 4 (see 2.4.3.4)
backward_f_code <= 4 (see 2.4.3.4)

If the constrained_parameters_flag is set, then the vbv_buffer_size field shall indicate a VBV buffer size less than or equal to 327 680 bits (20*1024*16; i.e. 40 kbytes).

If the constrained_parameters_flag is set, then the bit_rate field shall indicate a coded data rate less than or equal to 1 856 000 bits/s.

load_intra_quantizer_matrix -- This is a one-bit flag which is set to "1" if an intra_quantizer_matrix follows. If it is set to "0" then the default values defined below in raster-scan order, are used until the next occurence of the sequence header.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

intra_quantizer_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, stored in the zigzag scanning order shown in 2.4.4.1, replace the default values shown above. The value zero is forbidden. The value for intra_quant[0][0] shall always be 8. The new values shall be in effect until the next occurence of a sequence header.

load_non_intra_quantizer_matrix -- This is a one-bit flag which is set to "1" if a non_intra_quantizer_matrix follows. If it is set to "0" then the default values defined below are used until the next occurence of the sequence header.

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

non_intra_quantizer_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, stored in the zigzag scanning order shown in 2.4.4.1, replace the default values shown above. The value zero is forbidden. The new values shall be in effect until the next occurence of a sequence header.

extension_start_code -- The extension_start_code is the bit string 000001B5 in hexadecimal. It identifies the beginning of extension data. The extension data continue until receipt of another start code. It is a requirement to parse extension data correctly.

sequence_extension_data -- Reserved.

user_data_start_code -- The user_data_start_code is the bit string 000001B2 in hexadecimal. It identifies the beginning of user data. The user data continues until receipt of another start code.

user_data -- The user_data is defined by the users for their specific applications. The user data shall not contain a string of 23 or more zero bits.

2.4.3.3 Group of pictures layer

group_start_code -- The group_start_code is the bit string 000001B8 in hexadecimal. It identifies the beginning of a group of pictures.

time_code -- This is a 25-bit field containing the following: drop_frame_flag, time_code_hours, time_code_minutes, marker_bit, time_code_seconds and time_code_pictures. The fields correspond to the fields defined in the IEC standard (Publication 461) for "time and control codes for video tape recorders" (see annex E). The code refers to the first picture in the group of pictures that has a temporal_reference of zero. The drop_frame_flag can be set to either "0" or "1". It may be set to "1" only if the picture rate is 29,97Hz. If it is "0" then pictures are counted assuming rounding to the nearest integral number of pictures per second, for example 29,97 Hz would be rounded to and counted as 30 Hz. If it is "1" then picture numbers 0 and 1 at the start of each minute, except minutes 0, 10, 20, 30, 40, 50 are omitted from the count.

time_code	range of value	bits	
drop_frame_flag		1	
time_code_hours	0 - 23	5	**uimsbf**
time_code_minutes	0 - 59	6	**uimsbf**
marker_bit	1	1	**"1"**
time_code_seconds	0 - 59	6	**uimsbf**
time_code_pictures	0 - 59	6	**uimsbf**

closed_gop -- This is a one-bit flag which may be set to "1" if the group of pictures has been encoded without motion vectors pointing to the previous group of pictures.

This bit is provided for use during any editing which occurs after encoding. If the previous group of pictures is removed by editing, **broken_link** may be set to "1" so that a decoder may avoid displaying the B-Pictures immediately following the first I-Picture of the group of pictures. However if the **closed_gop** bit indicates that there are no prediction references to the previous group of pictures then the editor may choose not to set the **broken_link** bit as these B-Picures can be correctly decoded in this case.

broken_link -- This is a one-bit flag which shall be set to "0" during encoding. It is set to "1" to indicate that the B-Pictures immediately following the first I-Picture of a group of pictures cannot be correctly decoded because the other I-Picture or P-Picture which is used for prediction is not available (because of the action of editing).

A decoder may use this flag to avoid displaying pictures that cannot be correctly decoded.

extension_start_code -- See 2.4.3.2.

group_extension_data -- Reserved.

user_data_start_code -- See 2.4.3.2.

user_data -- See 2.4.3.2.

2.4.3.4 Picture layer

picture_start_code -- The picture_start_code is a string of 32-bits having the value 00000100 in hexadecimal.

temporal_reference -- The temporal_reference is a 10-bit unsigned integer associated with each input picture. It is incremented by one, modulo 1024, for each input picture. For the earliest picture (in display order) in each group of pictures, the temporal_reference is reset to zero.

The temporal_reference is assigned (in sequence) to the pictures in display order, no temporal_reference shall be omitted from the sequence.

picture_coding_type -- The picture_coding_type identifies whether a picture is an intra-coded picture(I), predictive-coded picture(P), bidirectionally predictive-coded picture(B), or intra-coded with only dc coefficients picture(D) according to the following table. D-pictures shall never be included in the same video sequence as the other picture coding types.

picture_coding_type	coding method
000	forbidden
001	intra-coded (I)
010	predictive-coded (P)
011	bidirectionally-predictive-coded (B)
100	dc intra-coded (D)
101	reserved
...	...
111	reserved

vbv_delay -- The vbv_delay is a 16-bit unsigned integer. For constant bitrate operation, the vbv_delay is used to set the initial occupancy of the decoder's buffer at the start of decoding the picture so that the decoder's buffer does not overflow or underflow. The vbv_delay measures the time needed to fill the VBV buffer from an initially empty state at the target bit rate, R, to the correct level immediately before the current picture is removed from the buffer.

The value of vbv_delay is the number of periods of the 90kHz system clock that the VBV should wait after receiving the final byte of the picture start code. It may be calculated from the state of the VBV as follows:

$$\text{vbv\_delay}_n = 90\,000 * B_n^* / R$$

where:

$n > 0$

B_n^* = VBV occupancy, measured in bits, immediately before removing picture n from the buffer but after removing any group of picture layer data, sequence header data and the **picture_start_code** that immediately precedes the data elements of picture n.

R = bitrate measured in bits/s. The full precision of the bitrate rather than the rounded value encoded by the **bit_rate** field in the sequence header shall be used by the encoder in the VBV model.

For non-constant bitrate operation vbv_delay shall have the value FFFF in hexadecimal.

full_pel_forward_vector -- If set to "1", then the motion vector values decoded represent integer pel offsets (rather than half-pel units) as reflected in the equations of 2.4.4.2.

forward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. The variables forward_r_size and forward_f used in the process of decoding the forward motion vectors are derived from **forward_f_code** as described in 2.4.4.2

full_pel_backward_vector -- If set to "1", then the motion vector values decoded represent integer pel offsets (rather than half pel units) as reflected in the equations of 2.4.4.3.

backward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. The variables backward_r_size and backward_f used in the process of decoding the backward motion vectors are derived from **backward_f_code** as described in 2.4.4.3.

extra_bit_picture -- A bit indicates the presence of the following extra information. If extra_bit_picture is set to "1", extra_information_picture will follow it. If it is set to "0", there are no data following it.

extra_information_picture -- Reserved.

extension_start_code -- See 2.4.3.2.

picture_extension_data -- Reserved.

user_data_start_code -- See 2.4.3.2.

user_data -- See 2.4.3.2.

2.4.3.5 Slice layer

slice_start_code -- The slice_start_code is a string of 32-bits. The first 24-bits have the value 000001 in hexadecimal and the last 8-bits are the slice_vertical_position having a value in the range 01 through AF hexadecimal inclusive.

slice_vertical_position -- This is given by the last eight bits of the slice_start_code. It is an unsigned integer giving the vertical position in macroblock units of the first macroblock in the slice. The slice_vertical_position of the first row of macroblocks is one. Some slices may have the same slice_vertical_position, since slices may start and finish anywhere. Note that the slice_vertical_position is constrained by 2.4.1 to define non-overlapping slices with no gaps between them. The maximum value of slice_vertical_position is 175.

quantizer_scale -- An unsigned integer in the range 1 to 31 used to scale the reconstruction level of the retrieved DCT coefficient levels. The decoder shall use this value until another quantizer_scale is encountered either at the slice layer or the macroblock layer. The value zero is forbidden.

extra_bit_slice -- A bit indicates the presence of the following extra information. If extra_bit_slice is set to "1", extra_information_slice will follow it. If it is set to "0", there are no data following it.

extra_information_slice -- Reserved.

2.4.3.6 Macroblock layer

macroblock_stuffing -- This is a fixed bit string "0000 0001 111" which can be inserted by the encoder to increase the bit rate to that required of the storage or transmission medium. It is discarded by the decoder.

macroblock_escape -- The macroblock_escape is a fixed bit-string "0000 0001 000" which is used when the difference between macroblock_address and previous_macroblock_address is greater than 33. It causes the value of macroblock_address_increment to be 33 greater than the value that will be decoded by subsequent macroblock_escapes and the macroblock_address_increment codewords.

For example, if there are two macroblock_escape codewords preceding the macroblock_address_increment, then 66 is added to the value indicated by macroblock_address_increment.

macroblock_address_increment -- This is a variable length coded integer coded as per table B.1 which indicates the difference between macroblock_address and previous_macroblock_address. The maximum value of macroblock_address_increment is 33. Values greater than this can be encoded using the macroblock_escape codeword.

The macroblock_address is a variable defining the absolute position of the current macroblock. The macroblock_address of the top-left macroblock is zero.

The previous_macroblock_address is a variable defining the absolute position of the last non-skipped macroblock (see 2.4.4.4 for the definition of skipped macroblocks) except at the start of a slice. At the start of a slice, previous_macroblock_address is reset as follows:

previous_macroblock_address=(slice_vertical_position-1)*mb_width-1;

The spatial position in macroblock units of a macroblock in the picture (mb_row, mb_column) can be computed from the macroblock_address as follows:

mb_row = macroblock_address / mb_width
mb_column = macroblock_address % mb_width

where mb_width is the number of macroblocks in one row of the picture.

NOTE - The slice_vertical_position differs from mb_row by one.

macroblock_type -- Variable length coded indicator which indicates the method of coding and content of the macroblock according to the tables B.2a through B.2d.

macroblock_quant -- Derived from macroblock_type.

macroblock_motion_forward -- Derived from macroblock_type.

macroblock_motion_backward -- Derived from macroblock_type.

macroblock_pattern -- Derived from macroblock_type.

macroblock_intra -- Derived from macroblock_type.

quantizer_scale -- An unsigned integer in the range 1 to 31 used to scale the reconstruction level of the retrieved DCT coefficient levels. The value zero is forbidden. The decoder shall use this value until another quantizer_scale is encountered either at the slice layer or the macroblock layer. The presence of quantizer_scale is determined from macroblock_type.

motion_horizontal_forward_code -- motion_horizontal_forward_code is decoded according to table B.4. The decoded value is required (along with forward_f - see 2.4.4.2) to decide whether or not motion_horizontal_forward_r appears in the bitstream.

motion_horizontal_forward_r -- An unsigned integer (of forward_r_size bits - see 2.4.4.2) used in the process of decoding forward motion vectors as described in 2.4.4.2.

motion_vertical_forward_code -- motion_vertical_forward_code is decoded according to table B.4. The decoded value is required (along with forward_f - see 2.4.4.2) to decide whether or not motion_vertical_forward_r appears in the bitstream.

motion_vertical_forward_r -- An unsigned integer (of forward_r_size bits - see 2.4.4.2) used in the process of decoding forward motion vectors as described in 2.4.4.2.

motion_horizontal_backward_code -- motion_horizontal_backward_code is decoded according to table B.4. The decoded value is required (along with backward_f - see 2.4.4.2) to decide whether or not motion_horizontal_backward_r appears in the bitstream.

motion_horizontal_backward_r -- An unsigned integer (of backward_r_size bits - see 2.4.4.2) used in the process of decoding backward motion vectors as described in 2.4.4.2.

motion_vertical_backward_code -- motion_vertical_backward_code is decoded according to table B.4. The decoded value is required (along with backward_f) to decide whether or not motion_vertical_backward_r appears in the bitstream.

motion_vertical_backward_r -- An unsigned integer (of backward_r_size bits) used in the process of decoding backward motion vectors as described in 2.4.4.3.

coded_block_pattern -- coded_block_pattern is a variable length code that is used to derive the variable cbp according to table B.3. If macroblock_intra is zero, cbp=0. Then the pattern_code[i] for i=0 to 5 is derived from cbp using the following:

```
pattern_code[i] = 0;
if ( cbp & (1<<(5-i)) ) pattern_code[i] = 1;
if ( macroblock_intra ) pattern_code[i] = 1 ;
```

pattern_code[0] -- If 1, then the upper left luminance block is to be received in this macroblock.

pattern_code[1] -- If 1, then the upper right luminance block is to be received in this macroblock.

pattern_code[2] -- If 1, then the lower left luminance block is to be received in this macroblock.

pattern_code[3] -- If 1, then the lower right luminance block is to be received in this macroblock.

pattern_code[4] -- If 1, then the chrominance block Cb is to be received in this macroblock.

pattern_code[5] -- If 1, then the chrominance block Cr is to be received in this macroblock.

end_of_macroblock -- This is a bit which is set to "1" and exists only in D-Pictures.

2.4.3.7 Block layer

dct_dc_size_luminance -- The number of bits in the following dct_dc_differential code, dc_size_luminance, is derived according to the VLC table B.5a. Note that this data element is used in intra coded blocks.

dct_dc_size_chrominance -- The number of bits in the following dct_dc_differential code, dc_size_chrominance, is derived according to the VLC table B.5b. Note that this data element is used in intra coded blocks.

dct_dc_differential -- A variable length unsigned integer. If dc_size_luminance or dc_size_chrominance (as appropriate) is zero, then dct_dc_differential is not present in the bitstream. dct_zz [] is the array of quantized DCT coefficients in zig-zag scanning order. dct_zz[i] for i=0..63 shall be set to zero initially. If dc_size_luminance or dc_size_chrominance (as appropriate) is greater than zero, then dct_zz[0] is computed as follows from dct_dc_differential:

For luminance blocks:

```
    if ( dct_dc_differential & ( 1 << (dc_size_luminance-1)) ) dct_zz[0] = dct_dc_differential ;
    else dct_zz[0] = ( (-1) << (dc_size_luminance) ) | (dct_dc_differential+1) ;
```

For chrominance blocks:

```
    if ( dct_dc_differential & ( 1 << (dc_size_chrominance-1)) ) dct_zz[0] = dct_dc_differential ;
    else dct_zz[0] = ( (-1) << (dc_size_chrominance) ) | (dct_dc_differential+1) ;
```

Note that this data element is used in intra coded blocks.

example for dc_size_luminance = 3	
dct_dc_differential	dct_zz[0]
000	-7
001	-6
010	-5
011	-4
100	4
101	5
110	6
111	7

dct_coeff_first -- A variable length code according to tables B.5c through B.5f for the first coefficient. The variables run and level are derived according to these tables. The zigzag-scanned quantized DCT coefficient list is updated as follows.

```
i = run ;
if ( s == 0 ) dct_zz[i] = level ;
if ( s == 1 ) dct_zz[i] = - level ;
```

The terms dct_coeff_first and dct_coeff_next are run-length encoded and dct_zz[i], i>=0 shall be set to zero initially. A variable length code according to tables B.5c through B.5f is used to represent the run-length and level of the DCT coefficients. Note that this data element is used in non-intra coded blocks.

dct_coeff_next -- A variable length code according to tables B.5c through B.5f for coefficients following the first retrieved. The variables run and level are derived according to these tables. The zigzag-scanned quantized DCT coefficient list is updated as follows.

```
i = i + run +1 ;
if ( s == 0 ) dct_zz[i] = level ;
if ( s == 1 ) dct_zz[i] = - level ;
```

If macroblock_intra == 1 then the term i shall be set to zero before the first dct_coeff_next of the block. The decoding of dct_coeff_next shall not cause i to exceed 63.

end_of_block -- This symbol is always used to indicate that no additional non-zero coefficients are present. It is used even if dct_zz[63] is non-zero. Its value is the bit-string "10" as defined in table B.5c.

2.4.4 The video decoding process

Compliance requirements for decoders are contained in ISO/IEC 11172-4.

2.4.4.1 Intra-coded macroblocks

In I-pictures all macroblocks are intra-coded and stored. In P-pictures and B-pictures, some macroblocks may be intra-coded as identified by macroblock_type. Thus, macroblock_intra identifies the intra-coded macroblocks.

The variables mb_row and mb_column locate the macroblock in the picture. They are defined in 2.4.3.6. The definitions of dct_dc_differential, and dct_coeff_next also have defined the zigzag-scanned quantized DCT coefficient list, dct_zz[]. Each dct_zz[] is located in the macroblock as defined by pattern_code[].

Define dct_recon[m][n] to be the matrix of reconstructed DCT coefficients of the block, where the first index identifies the row and the second the column of the matrix. Define dct_dc_y_past, dct_dc_cb_past and dct_dc_cr_past to be the dct_recon[0][0] of the most recently decoded intra-coded Y, Cb and Cr blocks respectively. The predictors dct_dc_y_past, dct_dc_cb_past and dct_dc_cr_past shall all be reset at the start of a slice and at non-intra-coded macroblocks (including skipped macroblocks) to the value 1 024 (128*8).

Define intra_quant[m][n] to be the intra quantizer matrix that is specified in the sequence header.

Note that intra_quant[0][0] is used in the dequantizer calculations for simplicity of description, but the result is overwritten by the subsequent calculation for the dc coefficient.

Define scan[m][n] to be the matrix defining the zigzag scanning sequence as follows:

$$\left|\begin{array}{cccccccc}
0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\
2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\
3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\
9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\
10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\
20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\
21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\
35 & 36 & 48 & 49 & 57 & 58 & 62 & 63
\end{array}\right|$$

Where n is the horizontal index and m is the vertical index.

Define past_intra_address as the macroblock_address of the most recently retrieved intra-coded macroblock within the slice. It shall be reset to -2 at the beginning of each slice.

Then dct_recon[m][n] shall be computed by any means equivalent to the following procedure for the first luminance block:

```
for (m=0; m<8; m++) {
        for (n=0; n<8; n++) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( 2 * dct_zz[i] * quantizer_scale * intra_quant[m][n] ) /16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2 047) dct_recon[m][n] = 2 047 ;
                if (dct_recon[m][n] < -2 048) dct_recon[m][n] = -2 048 ;
        }
}
dct_recon[0][0] = dct_zz[0] * 8 ;
if ( (macroblock_address - past_intra_address > 1) )
        dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
else
        dct_recon[0][0] = dct_dc_y_past + dct_recon[0][0] ;
dct_dc_y_past = dct_recon[0][0] ;
```

Note that this process disallows even valued numbers. This has been found to prevent accumulation of mismatch errors.

For the subsequent luminance blocks in the macroblock, in the order of the list defined by the array pattern_code[]:

```
for (m=0; m<8; m++) {
        for (n=0; n<8; n++) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( 2 * dct_zz[i] * quantizer_scale * intra_quant[m][n] ) /16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2 047)  dct_recon[m][n] = 2 047 ;
                if (dct_recon[m][n] < -2 048)  dct_recon[m][n] = -2 048 ;
        }
}
dct_recon[0][0] = dct_dc_y_past + (dct_zz[0] * 8) ;
dct_dc_y_past = dct_recon[0][0] ;
```

For the chrominance Cb block,:

```
for (m=0; m<8; m++) {
        for (n=0; n<8; n++) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( 2 * dct_zz[i] * quantizer_scale * intra_quant[m][n] ) /16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2 047)  dct_recon[m][n] = 2 047 ;
                if (dct_recon[m][n] < -2 048)  dct_recon[m][n] = -2 048 ;
        }
}
dct_recon[0][0] = dct_zz[0] * 8 ;
if ( ( macroblock_address - past_intra_address ) > 1 )
        dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
else
        dct_recon[0][0] = dct_dc_cb_past + dct_recon[0][0] ;
dct_dc_cb_past = dct_recon[0][0] ;
```

For the chrominance Cr block, :

```
for (m=0; m<8; m++) {
        for (n=0; n<8; n++) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( 2 * dct_zz[i] * quantizer_scale * intra_quant[m][n] ) /16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2 047)  dct_recon[m][n] = 2 047 ;
                if (dct_recon[m][n] < -2 048)  dct_recon[m][n] = -2 048 ;
        }
}
dct_recon[0][0] = dct_zz[0] * 8 ;
if ( ( macroblock_address - past_intra_address ) > 1 )
        dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
else
        dct_recon[0][0] = dct_dc_cr_past+ dct_recon[0][0] ;
dct_dc_cr_past = dct_recon[0][0] ;
```

After all the blocks in the macroblock are processed:

```
past_intra_address = macroblock_address ;
```

Values in the coded data elements leading to dct_recon[0][0] < 0 or dct_recon[0][0] > 2 047 are not permitted.

Once the DCT coefficients are reconstructed, the inverse DCT transform defined in annex A shall be applied to obtain the inverse transformed pel values in the range [-256, 255]. These pel values shall be limited to the range [0, 255] and placed in the luminance and chrominance matrices in the positions defined by mb_row , mb_column, and the list defined by the array pattern_code[].

2.4.4.2 Predictive-coded macroblocks in P-pictures

Predictive-coded macroblocks in P-Pictures are decoded in two steps.

First, the value of the forward motion vector for the macroblock is reconstructed and a prediction macroblock is formed, as detailed below.

Second, the DCT coefficient information stored for some or all of the blocks is decoded, dequantized, inverse DCT transformed, and added to the prediction macroblock.

Let **recon_right_for and recon_down_for** be the reconstructed horizontal and vertical components of the motion vector for the current macroblock, and recon_right_for_prev and recon_down_for_prev be the reconstructed motion vector for the previous predictive-coded macroblock. If the current macroblock is the first macroblock in the slice, or if the last macroblock that was decoded contained no motion vector information (either because it was skipped or macroblock_motion_forward was zero), then recon_right_for_prev and recon_down_for_prev shall be set to zero.

If no forward motion vector data exists for the current macroblock (either because it was skipped or macroblock_motion_forward == 0), the motion vectors shall be set to zero.

If forward motion vector data exists for the current macroblock, then any means equivalent to the following procedure shall be used to reconstruct the motion vector horizontal and vertical components.

forward_r_size and forward_f are derived from forward_f_code as follows:

```
        forward_r_size = forward_f_code - 1
        forward_f = 1 << forward_r_size

if ( (forward_f == 1) || (motion_horizontal_forward_code == 0) ) {
        complement_horizontal_forward_r = 0;
} else {
        complement_horizontal_forward_r = forward_f - 1 - motion_horizontal_forward_r;
}
if ( (forward_f == 1) || (motion_vertical_forward_code == 0) ) {
        complement_vertical_forward_r = 0;
} else {
        complement_vertical_forward_r = forward_f - 1 - motion_vertical_forward_r;
}

right_little = motion_horizontal_forward_code * forward_f;
if (right_little == 0) {
        right_big = 0;
} else {
        if (right_little > 0) {
                right_little = right_little - complement_horizontal_forward_r ;
                right_big = right_little - (32 * forward_f);
        } else {
                right_little = right_little + complement_horizontal_forward_r ;
                right_big = right_little + (32 * forward_f);
        }
}
```

```
down_little = motion_vertical_forward_code * forward_f;
if (down_little == 0) {
        down_big = 0;
} else {
        if (down_little > 0) {
                down_little = down_little - complement_vertical_forward_r ;
                down_big = down_little - (32 * forward_f);
        } else {
                down_little = down_little + complement_vertical_forward_r ;
                down_big = down_little + (32 * forward_f);
        }
}
```

Values of forward_f, motion_horizontal_forward_code and if present motion_horizontal_forward_r shall be such that right_little is not equal to forward_f * 16.

Values of forward_f, motion_vertical_forward_code and if present motion_vertical_forward_r shall be such that down_little is not equal to forward_f * 16.

```
        max = ( 16 * forward_f ) - 1 ;
        min = ( -16 * forward_f ) ;

        new_vector = recon_right_for_prev + right_little ;
        if ( (new_vector <= max) && (new_vector >= min) )
                recon_right_for = recon_right_for_prev + right_little ;
        else
                recon_right_for = recon_right_for_prev + right_big ;
        recon_right_for_prev = recon_right_for ;

        if ( full_pel_forward_vector ) recon_right_for = recon_right_for << 1 ;
        new_vector = recon_down_for_prev + down_little ;
        if ( (new_vector <= max) && (new_vector >= min) )
                recon_down_for = recon_down_for_prev + down_little ;
        else
                recon_down_for = recon_down_for_prev + down_big ;
        recon_down_for_prev = recon_down_for ;
        if ( full_pel_forward_vector ) recon_down_for = recon_down_for << 1 ;
```

The motion vectors in whole pel units for the macroblock, right_for and down_for, and the half pel unit flags, right_half_for and down_half_for, are computed as follows:

for luminance	for chrominance

right_for = recon_right_for >> 1 ;
down_for = recon_down_for >> 1 ;
right_half_for = recon_right_for - (2*right_for) ;
down_half_for = recon_down_for - (2*down_for) ;
	right_for = (recon_right_for / 2) >> 1 ;
	down_for = (recon_down_for / 2) >> 1 ;
	right_half_for = recon_right_for/2 - (2*right_for) ;
	down_half_for = recon_down_for/2 - (2*down_for);

Motion vectors leading to references outside a reference picture's boundaries are not allowed.

A positive value of the reconstructed horizontal motion vector (right_for) indicates that the referenced area of the past reference picture is to the right of the macroblock in the coded picture.

A positive value of the reconstructed vertical motion vector (down_for) indicates that the referenced area of the past reference picture is below the macroblock in the coded picture.

Defining pel_past[][] as the pel values of the past picture referenced by the forward motion vector, and pel[][] as the predictors for the pel values of the block being decoded, then:

```
        if ( (! right_half_for) && (! down_half_for) )
                pel[i][j] = pel_past[i+down_for][j+right_for] ;
```

```
if ( (! right_half_for) &&  down_half_for )
        pel[i][j] = ( pel_past[i+down_for][j+right_for] +
                                        pel_past[i+down_for+1][j+right_for] ) // 2 ;

if (  right_half_for && (! down_half_for) )
        pel[i][j] = ( pel_past[i+down_for][j+right_for] +
                                        pel_past[i+down_for][j+right_for+1] ) // 2 ;

if (  right_half_for &&  down_half_for )
        pel[i][j] = ( pel_past[i+down_for][j+right_for] + pel_past[i+down_for+1][j+right_for] +
        pel_past[i+down_for][j+right_for+1] + pel_past[i+down_for+1][j+right_for+1] ) // 4 ;
```

Define non_intra_quant[m][n] to be the non-intra quantizer matrix that is specified in the sequence header.

The DCT coefficients for each block present in the macroblock shall be reconstructed by any means equivalent to the following procedure:

```
for ( m=0; m<8; m++ ) {
        for ( n=0; n<8; n++ ) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( ( (2 * dct_zz[i]) + Sign(dct_zz[i]) ) *
                                quantizer_scale * non_intra_quant[m][n] ) / 16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2047)  dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
                if ( dct_zz[i] == 0 )
                        dct_recon[m][n] = 0 ;
        }
}
```

dct_recon[m][n] = 0 for all m, n in skipped macroblocks and when pattern[i] == 0.

Once the DCT coefficients are reconstructed, the inverse DCT transform defined in annex A shall be applied to obtain the inverse transformed pel values in the interval [-256, 255]. The inverse DCT pel values shall be added to the pel[i][j] which were computed above using the motion vectors. The result of the addition shall be limited to the interval [0,255]. The location of the pels is determined from mb_row, mb_column and the pattern_code list.

2.4.4.3 Predictive-coded macroblocks in B-pictures

Predictive-coded macroblocks in B-Pictures are decoded in four steps.

First, the value of the forward motion vector for the macroblock is reconstructed from the retrieved forward motion vector information, and the forward motion vector reconstructed for the previous macroblock, using the same procedure as for calculating the forward motion vector in P-pictures. However, for B-pictures the previous reconstructed motion vectors shall be reset only for the first macroblock in a slice, or when the last macroblock that was decoded was an intra-coded macroblock. If no forward motion vector data exists for the current macroblock, the motion vectors shall be obtained by:

```
recon_right_for = recon_right_for_prev,
recon_down_for = recon_down_for_prev.
```

Second, the value of the backward motion vector for the macroblock shall be reconstructed from the retrieved backward motion vector information, and the backward motion vector reconstructed for the previous macroblock using the same procedure as for calculating the forward motion vector in B-pictures. In this procedure, the variables needed to find the backward motion vector are substituted for the variables needed to find the forward motion vector. The variables and coded data elements used to calculate the backward motion vector are:

recon_right_back_prev, recon_down_back_prev, backward_f_code, full_pel_backward_vector
motion_horizontal_backward_code, motion_horizontal_backward_r,
motion_vertical_backward_code, motion_vertical_backward_r,

backward_r_size and backward_f are derived from backward_f_code as follows:

```
backward_r_size = backward_f_code - 1
backward_f = 1 << backward_r_size
```

The following variables result from applying the algorithm in 2.4.4.2, modified as described in the previous paragraphs in this clause:

```
right_for    right_half_for     down_for     down_half_for
right_back   right_half_back    down_back    down_half_back
```

They define the integral and half pel value of the rightward and downward components of the forward motion vector (which references the past picture in display order) and the backward motion vector (which references the future picture in display order).

Third, the predictors of the pel values of the block being decoded, pel [][], are calculated. If only forward motion vector information was retrieved for the macroblock, then pel[][] of the decoded picture shall be calculated according to the formulas in 2.4.4.2. If only backward motion vector information was retrieved for the macroblock, then pel[][] of the decoded picture shall be calculated according to the formulas in the predictive-coded macroblock clause, with "back" replacing "for", and pel_future[][] replacing pel_past[][]. If both forward and backward motion vectors information are retrieved, then let pel_for[][] be the value calculated from the past picture by use of the reconstructed forward motion vector, and let pel_back[][] be the value calculated from the future picture by use of the reconstructed backward motion vector. Then the value of pel[][] shall be calculated by:

```
pel[][] = ( pel_for[][] + pel_back[][] ) // 2 ;
```

Define non_intra_quant[m][n] to be the non-intra quantizer matrix that is specified in the sequence header.

Fourth, the DCT coefficients for each block present in the macroblock shall be reconstructed by any means equivalent to the following procedure:

```
for ( m=0; m<8; m++ ) {
        for ( n=0; n<8; n++ ) {
                i = scan[m][n] ;
                dct_recon[m][n] = ( ( (2 * dct_zz[i]) + Sign(dct_zz[i]) ) *
                                quantizer_scale * non_intra_quant[m][n] ) / 16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n]) ;
                if (dct_recon[m][n] > 2 047)  dct_recon[m][n] = 2 047 ;
                if (dct_recon[m][n] < -2 048)  dct_recon[m][n] = -2 048 ;
                if ( dct_zz[i] == 0 )
                        dct_recon[m][n] = 0 ;
        }
}
```

dct_recon[m][n] = 0 for all m, n in skipped macroblocks and when pattern[i] == 0.

Once the DCT coefficients are reconstructed, the inverse DCT transform defined in annex A shall be applied to obtain the inverse transformed pel values in the range [-256, 255]. The inverse DCT pel values shall be added to pel[][], which were computed above from the motion vectors. The result of the addition shall be limited to the interval [0,255]. The location of the pels is determined from mb_row, mb_column and the pattern_code list.

2.4.4.4 Skipped macroblocks

For some macroblocks there are no coded data, that is neither motion vector information nor DCT information is available to the decoder. These macroblocks are called skipped macroblocks and are indicated when the macroblock_address_increment is greater than 1.

In I-pictures, all macroblocks shall be coded and there shall be no skipped macroblocks.

In P-pictures, the skipped macroblock is defined to be a macroblock with a reconstructed motion vector equal to zero and no DCT coefficients.

In B-pictures, the skipped macroblock is defined to have the same macroblock_type (forward, backward, or both motion vectors) as the prior macroblock, differential motion vectors equal to zero, and no DCT coefficients. In a B-picture, a skipped macroblock shall not follow an intra-coded macroblock.

2.4.4.5 Forced updating

This function is achieved by forcing the use of an intra-coded macroblock. The update pattern is not defined. For control of accumulation of IDCT mismatch error, each macroblock shall be intra-coded at least once per every 132 times it is coded in a P-picture without an intervening I-picture.

Annex A

(normative)

8 by 8 Inverse discrete cosine transform

The 8 by 8 inverse discrete cosine transform for I-pictures and P-pictures shall conform to IEEE Draft Standard, P1180/D2, July 18, 1990. For B-pictures this specification may also be applied but may be unnecessarily stringent. Note that clause 2.3 of P1180/D2 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-coding in order to control the accumulation of mismatch errors. The maximum refresh period requirement for this part of ISO/IEC 11172 shall be 132 intra-coded pictures or predictive-coded pictures as stated in 2.4.4.5, which is the same as indicated in P1180/D2 for visual telephony according to CCITT Recommendation H.261 [5].

Annex B

(normative)

Variable length code tables

Introduction

This annex contains the variable length code tables for macroblock addressing, macroblock type, macroblock pattern, motion vectors, and DCT coefficients.

B.1 Macroblock addressing

Table B.1. -- Variable length codes for macroblock_address_increment.

macroblock_address_ increment VLC code	increment value
1	1
011	2
010	3
0011	4
0010	5
0001 1	6
0001 0	7
0000 111	8
0000 110	9
0000 1011	10
0000 1010	11
0000 1001	12
0000 1000	13
0000 0111	14
0000 0110	15
0000 0101 11	16
0000 0101 10	17
0000 0101 01	18
0000 0101 00	19
0000 0100 11	20
0000 0100 10	21
0000 0100 011	22
0000 0100 010	23
0000 0100 001	24
0000 0100 000	25
0000 0011 111	26
0000 0011 110	27
0000 0011 101	28
0000 0011 100	29
0000 0011 011	30
0000 0011 010	31
0000 0011 001	32
0000 0011 000	33
0000 0001 111	macroblock_stuffing
0000 0001 000	macroblock_escape

B.2 Macroblock type

The properties of the macroblock are determined by the macroblock type VLC according to these tables.

Table B.2a. -- Variable length codes for macroblock_type in intra-coded pictures (I-pictures).

macroblock_ typeVLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock_ motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	0	0	0	1
01	1	0	0	0	1

Table B.2b. -- Variable length codes for macroblock_type in predictive-coded pictures (P-pictures).

macroblock_ typeVLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock_ motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	1	0	1	0
01	0	0	0	1	0
001	0	1	0	0	0
00011	0	0	0	0	1
00010	1	1	0	1	0
00001	1	0	0	1	0
000001	1	0	0	0	1

Table B.2c. -- Variable length codes for macroblock_type in bidirectionally predictive-coded pictures (B-pictures).

macroblock_ typeVLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock_ motion_ backward	macroblock_ pattern	macroblock_ intra
10	0	1	1	0	0
11	0	1	1	1	0
010	0	0	1	0	0
011	0	0	1	1	0
0010	0	1	0	0	0
0011	0	1	0	1	0
00011	0	0	0	0	1
00010	1	1	1	1	0
000011	1	1	0	1	0
000010	1	0	1	1	0
000001	1	0	0	0	1

Table B.2d. -- Variable length codes for macroblock_type in dc intra-coded pictures (D-pictures).

macroblock_ typeVLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock_ motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	0	0	0	1

B.3 Macroblock pattern

Table B.3. -- Variable length codes for coded_block_pattern.

coded_block_pattern VLC code	cbp
111	60
1101	4
1100	8
1011	16
1010	32
1001 1	12
1001 0	48
1000 1	20
1000 0	40
0111 1	28
0111 0	44
0110 1	52
0110 0	56
0101 1	1
0101 0	61
0100 1	2
0100 0	62
0011 11	24
0011 10	36
0011 01	3
0011 00	63
0010 111	5
0010 110	9
0010 101	17
0010 100	33
0010 011	6
0010 010	10
0010 001	18
0010 000	34
0001 1111	7
0001 1110	11
0001 1101	19
0001 1100	35
0001 1011	13
0001 1010	49
0001 1001	21
0001 1000	41
0001 0111	14
0001 0110	50
0001 0101	22
0001 0100	42
0001 0011	15
0001 0010	51
0001 0001	23
0001 0000	43
0000 1111	25
0000 1110	37
0000 1101	26
0000 1100	38
0000 1011	29
0000 1010	45
0000 1001	53
0000 1000	57
0000 0111	30
0000 0110	46
0000 0101	54
0000 0100	58
0000 0011 1	31
0000 0011 0	47
0000 0010 1	55
0000 0010 0	59
0000 0001 1	27
0000 0001 0	39

B.4 Motion vectors

Table B.4. -- Variable length codes for motion_horizontal_forward_code, motion_vertical_forward_code, motion_horizontal_backward_code, and motion_vertical_backward_code.

motion VLC code	code
0000 0011 001	-16
0000 0011 011	-15
0000 0011 101	-14
0000 0011 111	-13
0000 0100 001	-12
0000 0100 011	-11
0000 0100 11	-10
0000 0101 01	-9
0000 0101 11	-8
0000 0111	-7
0000 1001	-6
0000 1011	-5
0000 111	-4
0001 1	-3
0011	-2
011	-1
1	0
010	1
0010	2
0001 0	3
0000 110	4
0000 1010	5
0000 1000	6
0000 0110	7
0000 0101 10	8
0000 0101 00	9
0000 0100 10	10
0000 0100 010	11
0000 0100 000	12
0000 0011 110	13
0000 0011 100	14
0000 0011 010	15
0000 0011 000	16

B.5 DCT coefficients

Table B.5a -- Variable length codes for dct_dc_size_luminance.

VLC code	dct_dc_size_luminance
100	0
00	1
01	2
101	3
110	4
1110	5
11110	6
111110	7
1111110	8

Table B.5b. -- Variable length codes for dct_dc_size_chrominance.

VLC code	dct_dc_size_chrominance
00	0
01	1
10	2
110	3
1110	4
11110	5
111110	6
1111110	7
11111110	8

Table B.5c. -- Variable length codes for dct_coeff_first and dct_coeff_next.

dct_coeff_first and dct_coeff_next variable length code (NOTE1)	run	level
10	end_of_block	
1 s (NOTE2)	0	1
11 s (NOTE3)	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1 s	0	3
0011 1 s	3	1
0011 0 s	4	1
0001 10 s	1	2
0001 11 s	5	1
0001 01 s	6	1
0001 00 s	7	1
0000 110 s	0	4
0000 100 s	2	2
0000 111 s	8	1
0000 101 s	9	1
0000 01	escape	
0010 0110 s	0	5
0010 0001 s	0	6
0010 0101 s	1	3
0010 0100 s	3	2
0010 0111 s	10	1
0010 0011 s	11	1
0010 0010 s	12	1
0010 0000 s	13	1
0000 0010 10 s	0	7
0000 0011 00 s	1	4
0000 0010 11 s	2	3
0000 0011 11 s	4	2
0000 0010 01 s	5	2
0000 0011 10 s	14	1
0000 0011 01 s	15	1
0000 0010 00 s	16	1

NOTES
1 - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.
2 - This code shall be used for dct_coeff_first.
3 - This code shall be used for dct_coeff_next.

Table B.5d. -- Variable length codes for dct_coeff_first and dct_coeff_next.

dct_coeff_first and dct_coeff_next variable length code (NOTE)	run	level
0000 0001 1101 s	0	8
0000 0001 1000 s	0	9
0000 0001 0011 s	0	10
0000 0001 0000 s	0	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1

NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

Table B.5e. -- Variable length codes for dct_coeff_first and dct_coeff_next (concluded).

dct_coeff_first and dct_coeff_next variable length code (NOTE)	run	level
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1

NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

Table B.5f. -- Encoding of run and level following an escape code either as a 14-bit fixed length code (-127 <= level <= 127) or as a 22-bit fixed length code (-255 <= level <= -128, 128 <= level <= 255). (Note - This yields total escape code lengths of 20-bits and 28-bits respectively).

fixed length code	run
0000 00	0
0000 01	1
0000 10	2
...	...
...	...
...	...
...	...
...	...
...	...
1111 11	63

fixed length code	level
forbidden	-256
1000 0000 0000 0001	-255
1000 0000 0000 0010	-254
...	
1000 0000 0111 1111	-129
1000 0000 1000 0000	-128
1000 0001	-127
1000 0010	-126
...	...
1111 1110	-2
1111 1111	-1
forbidden	0
0000 0001	1
...	...
0111 1111	127
0000 0000 1000 0000	128
0000 0000 1000 0001	129
...	
0000 0000 1111 1111	255

Annex C

(normative)

Video buffering verifier

Constant rate coded video bitstreams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in clause C.1.

The VBV is a hypothetical decoder which is conceptually connected to the output of an encoder. Coded data are placed in the input buffer of the model decoder at the constant bitrate that is being used. Coded data is removed from the buffer as defined in C.1.4, below. It is a requirement of the encoder (or editor) that the bitstream it produces will not cause the VBV input buffer to either overflow or underflow.

C.1 Video buffering verifier

C.1.1 The VBV and the video encoder have the same clock frequency as well as the same picture rate, and are operated synchronously.

C.1.2 The VBV has an input buffer of size B, where B is given in the vbv_buffer_size field in the sequence header.

C.1.3 The VBV input buffer is initially empty. After filling the input buffer with all the data that precedes the first picture start code and the picture start code itself, the input buffer is filled from the bitstream for the time specified by the vbv_delay field in the video bitstream.

C.1.4 All of the picture data for the picture that has been in the buffer longest is instantaneously removed. Then after each subsequent picture interval all of the picture data for the picture which at that time has been in the buffer longest is instantaneously removed.

For the purposes of this annex picture data includes any sequence header and group of picture layer data that immediately precede the picture start code as well as all the picture data elements and any trailing stuffing bits or bytes. For the first coded picture in the video sequence, any zero bit or byte stuffing immediately preceding the sequence header is also included in the picture data.

The VBV buffer is examined immediately before removing any picture data and immediately after this picture data is removed. Each time the VBV is examined its occupancy shall lie between zero bits and B bits where, B is the size of the VBV buffer indicated by vbv_buffer_size in the sequence header.

This is a requirement for the entire video bitstream.

To meet these requirements the number of bits for the (n+1)'th coded picture d_{n+1} shall satisfy

$$d_{n+1} > B_n + (2R/P) - B$$

$$d_{n+1} <= B_n + (R/P)$$ Real-valued arithmetic is used in these inequalities.

where

$n >= 0$

B = VBV receiving buffer size given by **vbv_buffer_size** * 16 384 bits.

B_n = the buffer occupancy (measured in bits) just after time t_n

R = bitrate measured in bits/s. The full precision of the bitrate rather than the rounded value encoded by the **bit_rate** field in the sequence header shall be used by the encoder in the VBV model.

P = nominal number of pictures per second

t_n = the time when the n'th coded picture is removed from the VBV buffer

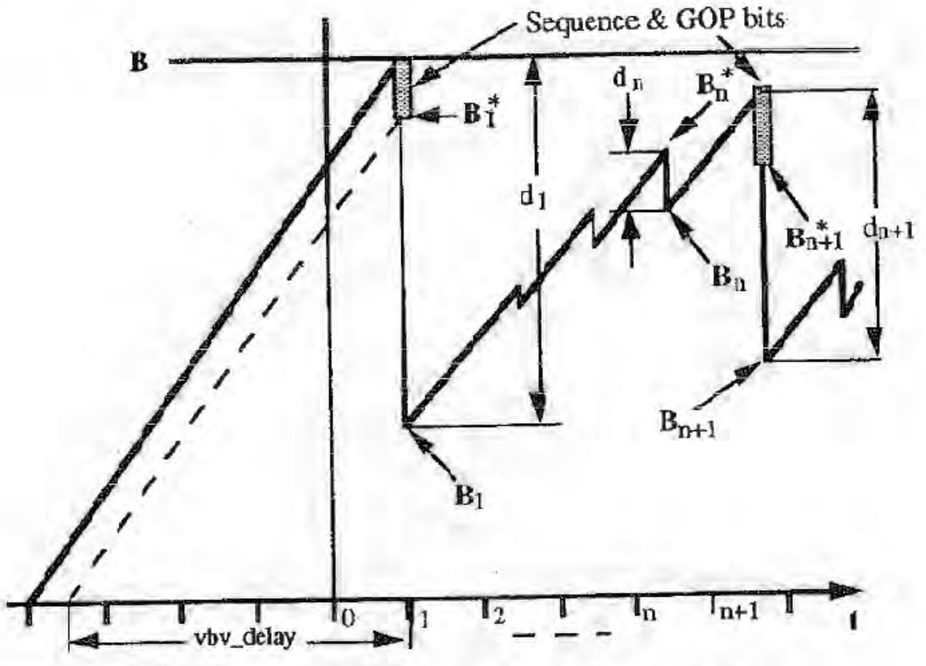


Figure C.1 -- VBV buffer occupancy

Annex D

(informative)

Guide to encoding video

D.1 Introduction

This annex provides background material to help readers understand and implement this part of ISO/IEC 11172. The normative clauses of this part of ISO/IEC 11172 do not specify the design of a decoder. They provide even less information about encoders; they do not specify what algorithms encoders should employ in order to produce a valid bitstream. The normative material is written in a concise form and contains few examples; consequently is not easy to understand. This annex attempts to address this problem by explaining coding methods, giving examples, and discussing encoding and decoding algorithms which are not directly covered by this part of ISO/IEC 11172.

The normative clauses specify the bitstream in such a way that it is fairly straightforward to design a compliant decoder. Decoders may differ considerably in architecture and implementation details, but have very few choices during the decoding process: the methods and the results of the decoding process are closely specified. Decoders do have some freedom in methods of post processing and display, but the results of such post processing cannot be used in subsequent decoding steps.

The situation is quite different for encoders. This part of ISO/IEC 11172 does not specify how to design or implement an encoder which produces good quality video. This annex devotes a major part to discussing encoder algorithms.

This part of ISO/IEC 11172 was developed by ISO/IEC JTC1/SC29/WG11 which is widely known as MPEG (Moving Pictures Expert Group). This part of ISO/IEC 11172 was developed in response to industry needs for an efficient way of storing and retrieving audio and video information on digital storage media (DSM). CD-ROM is an inexpensive medium which can deliver data at approximately 1,2 Mbits/s, and this part of ISO/IEC 11172 was aimed at approximately this data rate. The "constrained parameters bitstream", a subset of all permissible bitstreams that is expected to be widely used, is limited to data rates up to 1 856 000 bits/s. However, it should be noted that this part of ISO/IEC 11172 is not limited to this value and may be used at higher data rates.

Two other relevant International Standards were being developed during the work of the MPEG video committee: H.261 by CCITT aimed at telecommunications applications [5], and ISO/IEC 10918 by the ISO/IEC JTC1/SC29 (JPEG) committee aimed at the coding of still pictures [6]. Elements of both of these standards were incorporated into this part of ISO/IEC 11172, but subsequent development work by the committee resulted in coding elements that are new to this part of ISO/IEC 11172. Le Gall [2] gives an account of the method by which ISO/IEC JTC1/SC29/WG11 (MPEG) developed this part of ISO/IEC 11172, and a summary of this part of ISO/IEC 11172 itself.

D.2 Overview

D.2.1 Video concepts

This part of ISO/IEC 11172 defines a format for compressed digital video. This annex describes some ways in which practical encoders and decoders might be implemented.

Although this part of ISO/IEC 11172 is quite flexible, the basic algorithms have been tuned to work well at data rates of about 1 to 1,5 M bits/s, at spatial resolutions of about 350 pels horizontally by about 250 pels vertically, and picture rates of about 24 to 30 pictures/s. The use of the word "picture" as opposed to "frame" is deliberate. This part of ISO/IEC 11172 codes progressively-scanned images and does not recognize the concept of interlace. Interlaced source video must be converted to a non-interlaced format before coding. After decoding, the decoder may optionally produce an interlaced format for display.

This part of ISO/IEC 11172 is designed to permit several methods of viewing coded video which are normally associated with VCRs such as forward playback, freeze picture, fast forward, fast reverse, and slow forward. In addition, random access may be possible. The ability of the decoder to implement these modes depends to some extent on the nature of the digital storage medium on which the coded video is stored.

The overall process of encoding and decoding is illustrated below:

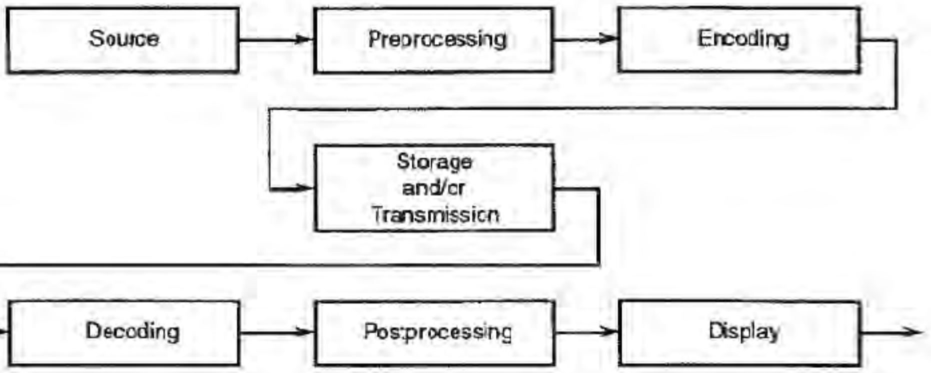


Figure D.1 -- Coding and decoding process

Figure D.1 shows a typical sequence of operations that must be performed before moving pictures can be seen by a viewer. The unencoded source may exist in many forms, such as the CCIR 601 format. Clause D.3 describes how such a source may be converted into the appropriate resolution for subsequent encoding. In the encoding step, the encoder must be aware of the decoder buffer capacity, and the need of the decoder to match the rate of the media to the rate of filling the picture buffer with each successive picture. To this end, a model of the decoder buffer and its overflow and underflow problem is introduced in D.4, and rate control is described in D.6.1 The structure of an ISO/IEC 11172-2 bitstream is covered in D.5, as are the coding operations that compress the video. Following the encoding process, the bitstream may be copied to a storage medium. To view the moving pictures, the decoder accesses the ISO/IEC 11172-2 bitstream, and decodes it as described in D.7. Postprocessing for display is described in D.8.

D.2.2 MPEG video compression techniques

Video is represented as a succession of individual pictures, and each picture is treated as a two-dimensional array of picture elements (pels). The colour representation for each pel consists of three components: Y (luminance), and two chrominance components, Cb and Cr.

Compression of digitized video comes from the use of several techniques: subsampling of the chrominance information to match the sensitivity of the human visual system (HVS), quantization, motion compensation (MC) to exploit temporal redundancy, frequency transformation by discrete cosine transform (DCT) to exploit spatial redundancy, variable length coding (VLC), and picture interpolation.

D.2.2.1 Subsampling of chrominance information

The HVS is most sensitive to the resolution of an image's luminance component, so the Y pel values are encoded at full resolution. The HVS is less sensitive to the chrominance information. Subsampling reduces the number of pel values by systematically combining them with a type of averaging process. This reduces the amount of information to be compressed by other techniques.The International Standard retains one set of chrominance pels for each 2x2 neighbourhood of luminance pels.

D.2.2.2 Quantization

Quantization represents a range of values by a single value in the range. For example, converting a real number to the nearest integer is a form of quantization. The quantized range can be concisely represented as an integer code, which can be used to recover the quantized value during decoding. The difference between the actual value and the quantized value is called the quantization noise. Under some circumstances, the HVS is less sensitive to quantization noise so such noise can be allowed to be large, thus increasing coding efficiency.