TITLE PAGE PROVIDED BY ISO

CD 11172-2

CODING OF MOVING PICTURES AND ASSOCIATED AUDIO -- FOR DIGITAL STORAGE MEDIA AT UP TO ABOUT 1.5 Mbit/s --

Part 2 Video

CONTENTS

FOREWORD

This standard is a committee draft that was submitted for approval to ISO-IEC/JTC1 SC29 on 23 November 1991. It was prepared by SC29/WG11 also known as MPEG (Moving Pictures Expert Group). MPEG was formed in 1988 to establish a standard for the coded representation of moving pictures and associated audio stored on digital storage media.

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

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

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

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

INTRODUCTION - PART 2: VIDEO

I.1 Purpose

This Part of this standard 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. The standard specifies a coded representation that can be used for compressing video sequences to bitrates around 1.5 Mbit/s. The use of this standard 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 with both 625-line and 525-line television and provides flexibility for use with workstation and personal computer displays.

The standard was developed to operate principally from storage media offering a continual transfer rate of about 1.5 Mbit/s. Nevertheless it can be used more widely than this because the approach taken is generic.

I.1.1 Coding Parameters

The intention in developing this standard 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 standard, a sub-set of these coding parameters known as the "Constrained Parameters bitstream" 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 in not a requirement of this standard. 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
Pixel rate	Less than or equal to 396x25 macroblocks per second
Picture rate	Less than or equal to 30 Hz
Motion vector range	Less than +/- 64 pels (using half-pel vectors) [Vector scale code <= 4)
Input buffer size (in VBV model)	Less than or equal to 327 680 bits
Bitrate	Less than or equal to 1 856 000 bits/second (constant bitrate)

I.2 Overview of the Algorithm

The coded representation defined in this standard 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 very high compression, 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 high compression. The first, which is almost independent from this standard, 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 both for causal prediction of the current picture from a previous picture, and for non-causal, interpolative prediction from past and future pictures. Motion vectors are defined for each 16-pel by 16-line region of the image. 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 residual DCT information, and transmitted using variable length codes.

I.2.1 Temporal Processing

Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intra 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 moderate compression. Predicted pictures (P-Pictures) are coded more efficiently using motion compensated prediction from a past intra or predicted picture and are generally used as a reference for further prediction. Bidirectionally-predicted pictures (B-Pictures) provide the highest degree of compression but require both past and future reference pictures for motion compensation. Bidirectionally-predicted 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 I.1 illustrates the relationship between the three different picture types.

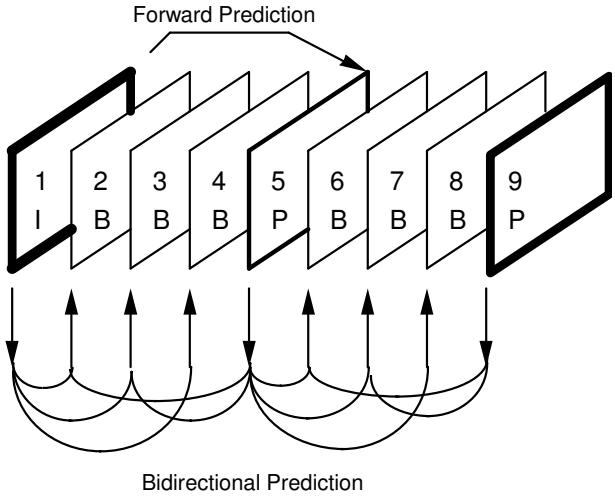


Figure I.1 Example of temporal picture structure, M=3

The fourth picture type defined in the standard, the DC-picture, is provided to allow a simple, but limited quality, fast-forward mode.

I.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 the coding gain 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, forwards- prediction-coded, backwards-prediction coded, and bidirectionally-prediction-coded macroblocks are permitted in bidirectionally-predicted pictures. Depending on the type of the macroblock, motion vector information and other side information is stored with the compressed prediction error signal in each macroblock. The motion vectors are encoded differentially with respect to the last transmitted 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. The standard does not specify how this should be done.

I.2.3 Spatial Redundancy Reduction

Both original pictures and prediction error signals have high spatial redundancy. This standard uses a block-based DCT method with visually weighted quantization and run-length coding. After motion compensated prediction or interpolation, the residual picture is split into 8 by 8 blocks. These are transformed into the DCT domain where they are weighted 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.

I.3 Encoding

The standard 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 I.2 shows the main functional blocks.

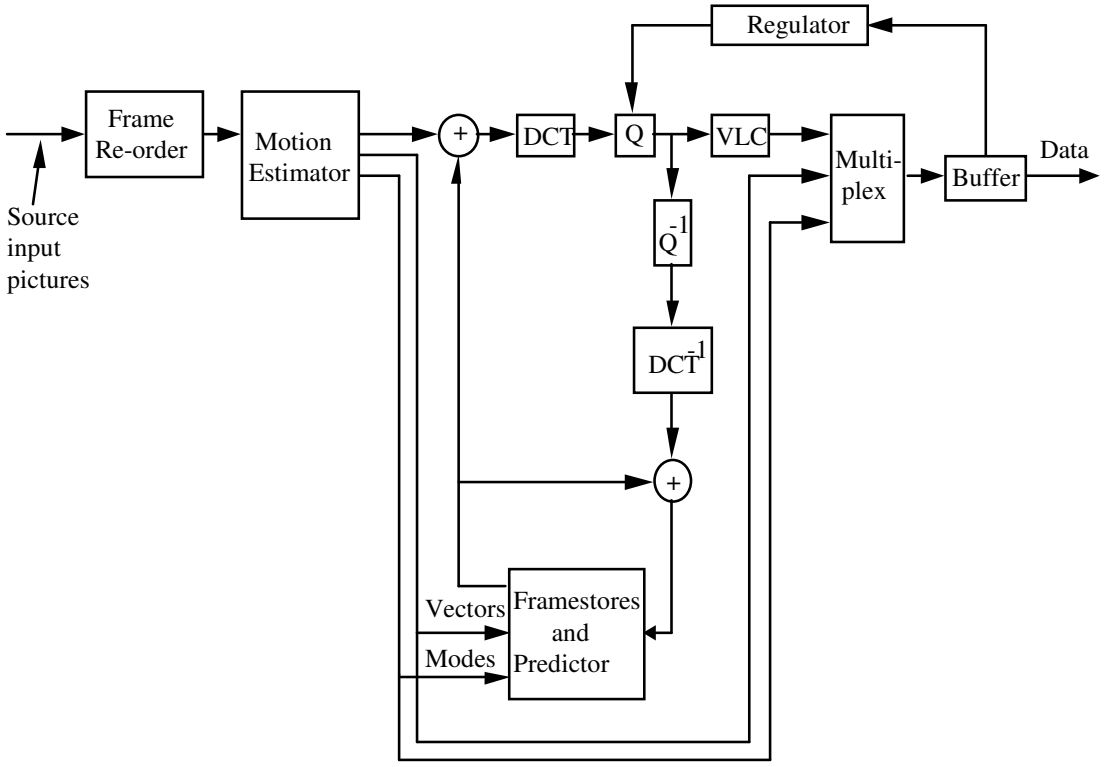


Figure I.2 Simplified Video Encoder Block Diagram

The input video signal must be digitized and represented as a luminance and two colour difference signals (Y, C_r, C_b). This may be followed by preprocessing and format conversion to select an appropriate window, resolution and input format. The standard requires that the colour difference signals (C_r and C_b) 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. Typically in P-Pictures one vector is needed for each 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 component 8 by 8 blocks: four blocks of luminance, one block of Cb chrominance, and one block of Cr chrominance. See Figure I.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 sensitivity of the human visual system.

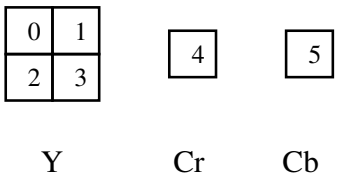


Figure I.3 Macroblock Structure

Firstly, for a given macroblock, the coding mode is chosen. It depends on the picture type, the effectiveness of motion compensated predictionin 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. The 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 (type, vectors etc) and the quantized coefficient data is 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. The standard specifies an abstract model of the buffering system (the Video Buffer 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 error signal produced is then added back to the prediction signal and clipped to the required range to give a decoded reference picture.

I.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 standard. 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 I.4 shows the main functional blocks.

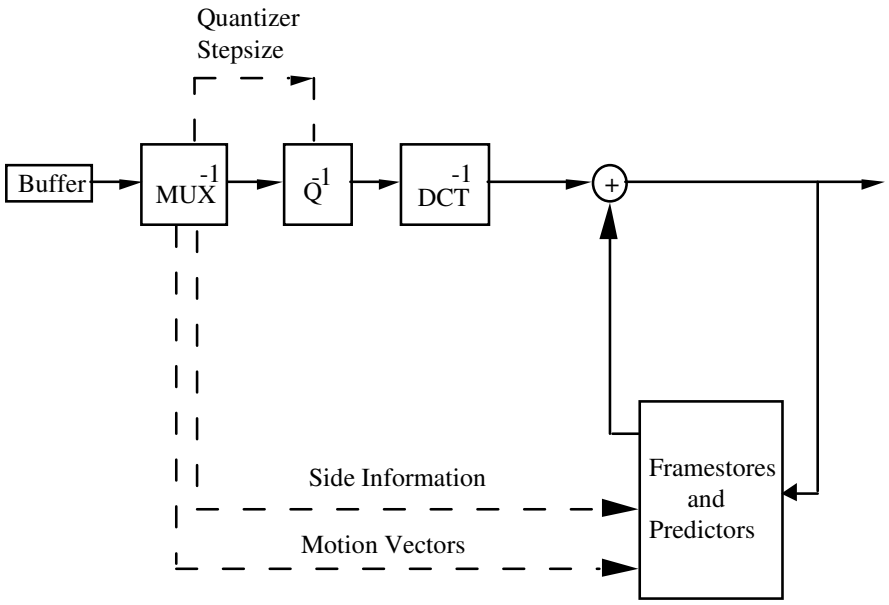


Figure I.4 Basic Video Decoder Block Diagram

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 stream, 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 is decoded and inverse quantized. Each 8 by 8 block of coefficient data is transformed by an inverse DCT (specified in 2-Annex A), and the result is added to the prediction signal and clipped 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 coding 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.

I.5 Structure of the Coded bitstream

This standard specifies a syntax for a compressed 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

I.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 forwards play of the sequence. The coded bitstream has been designed to support a number of these operations.

I.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.

I.6.2 Fast Search

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

I.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.

I.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 standard. Nevertheless the compression scheme defined in this standard 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.

I.6.5 Editing

There is a conflict between the requirement for high compression 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.

1 GENERAL NORMATIVE ELEMENTS

1.1 Scope

This international standard 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 play, fast reverse play, normal speed reverse playback, pause and still pictures. This international standard is compatible with standard 525- and 625-line television formats, and it provides flexibility for use with personal computer and workstation displays.

This international standard 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. The storage media may be directly connected to the decoder, or via communications means such as busses, LANs, or telecommunications links. This international standard is intended for non-interlaced video formats having approximately 288 lines of 352 pels and picture rates around 24Hz to 30Hz.

1.2 References

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

Recommendations and reports of the CCIR, 1990
XVIIth Plenary Assembly, Dusseldorf, 1990
Volume XI - Part 1
Broadcasting Service (Television)
Rec 601-2 "Encoding parameters of digital television for studios"

IEEE Draft Standard "Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform", P1180/D2, July 18, 1990.

Recommendations and reports of the CCIR
Volume X
Rec 953 "Encoding parameters of digital audio"

2 TECHNICAL NORMATIVE ELEMENTS

2.1 Definitions

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

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

access unit: in the case of compressed audio an access unit is an audio access unit. In the case of compressed video an access unit is the coded representation of a picture.

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

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

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

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

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

Audio Access Unit: An Audio Access Unit is defined as the smallest part of the encoded bitstream which can be decoded by itself, where decoded means "fully reconstructed sound".

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

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

Bark: Unit of critical band rate.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CRC: Cyclic redundancy code.

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

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

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

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

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

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

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

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

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

decoder: An embodiment of a decoding process.

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

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

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

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

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

discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in 2-Annex A of Part 2.

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

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

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

encoder: An embodiment of an encoding process.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Huffman coding: A specific method for entropy coding.

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

IMDCT: Inverse Modified Discrete Cosine Transform.

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

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

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

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

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

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

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

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

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

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

macroblock: The four 8 by 8 blocks of luminance data and the two corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the pel data and sometimes to the coded representation of the pel and other data elements defined in the macroblock layer of the syntax defined in Part 2 of this International Standard. The usage is clear from the context.

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

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

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

MDCT: Modified Discrete Cosine Transform.

motion compensation: The use of motion vectors to improve the efficiency of the prediction of pel values. The prediction uses motion vectors to provide offsets into the past and/or future reference frames containing previously decoded pels that are used to form the prediction.

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

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

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

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

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

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

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

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

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

packet: A packet consists of a header followed by a number of contiguous bytes from an elementary data stream. It is a layer in the system coding syntax described in Part 1 of this International Standard.

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

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

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

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

picture period: The reciprocal of the picture rate.

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

picture: Source or reconstructed image data. A picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. The Picture layer is one of the layers in the coding syntax defined in Part 2 of this International Standard. NOTE: the term "picture" is always used in this standard in preference to the terms field or frame.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Scalefactor index: A numerical code for a scalefactor.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2 Symbols and Abbreviations

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

2.2.1 Arithmetic Operators

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

2.2.2 Logical Operators

||	Logical OR.

&& Logical AND.

! Logical NOT

2.2.3 Relational Operators

> Greater than.

>= Greater than or equal to.

< Less than.

<= Less than or equal to.

== Equal to.

!= Not equal to.

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

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

2.2.4 Bitwise Operators

& AND

| OR

>> Shift right with sign extension.

<< Shift left with zero fill.

2.2.5 Assignment

= Assignment operator.

2.2.6 Mnemonics

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

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in the standard. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.

uimsbf Unsigned integer, most significant bit first.

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

ch channel.

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

rpchof remainder polynomial coefficients, highest order first.

sb sub-band.

scfsi scale-factor selector information.

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

2.2.7 Constants

pi 3.14159265359...
e 2.71828182846...

2.3 Method of Describing Bit Stream Syntax

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

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

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

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

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

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

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

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

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

While the syntax is expressed in procedural terms, it should not be assumed that clause 2.4.3 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include a means to look for start codes in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardized.

Definition of bytealigned function

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

Definition of nextbits function

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

Definition of next_start_code function

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

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

2.4 Requirements

2.4.1 Coding Structure and Parameters

Video Sequence

A 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.

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 pictures intended to assist random access into the sequence.
In the stored bit stream, the first coded picture in a group of pictures is an I-Picture. The order of the pictures in the coded stream is the order in which the decoder processes them in normal play. 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,

```
X  1I   2B   3B    4P   5B   6B   7P  X  8B   9B   10I  11B  12B  13P
   14B  15B  16P  X 17B  18B  19I  20B    21B  22P  23B  24B  25P  X 26B
   27B  28I  ...
```

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

```
              X  1I   4P   2B   3B   7P   5B   6B   X 10I  8B   9B   13P
   11B  12B      16P  14B  15B  X 19I  17B  18B  22P    20B  21B  25P  23B
   24B  X 28I    26B  27B  ...
```

where the Xs mark the group of pictures boundaries. Note that in this example, the first group of pictures is two pictures shorter than in subsequent groups of pictures, since at the beginning of video coding there are no B-pictures preceding the first I-Picture. However, in general 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	26	27	...								

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 play, or fast and normal reverse play 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 input to the decoder may be different from the one at the encoder output due to editing.

Picture

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

The Y, Cr and Cb 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 standard 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 standard.

The luminance and chrominance samples are positioned as shown in Figure 1, where "x" marks the position of the luminance (Y) pels and "0" marks the position of the chrominance (Cr and Cb) pels:

```
x       x   |   x       x   |   x       x
    0       |       0       |       0
x       x   |   x       x   |   x       x
-----------------------------------------------
x       x   |   x       x   |   x       x
    0       |       0       |       0
x       x   |   x       x   |   x       x
-----------------------------------------------
x       x   |   x       x   |   x       x
    0       |       0       |       0
x       x   |   x       x   |   x       x
```

Figure 1 The position of luminance and chrominance samples.

There are four types of pictures that use different coding methods.

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

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

A **Bidirectionally predictive-coded (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 Clause 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. Figure 2 shows the arrangement of these blocks. A skipped macroblock is one for which no information is stored (see Clause 2.4.4.4).

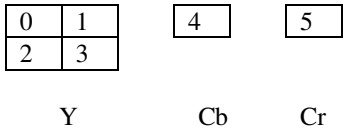


Figure 2 The arrangement of blocks in a macroblock.

Block

A **block** is an orthogonal 8-pel by 8-line section of a luminance or chrominance component. The order of blocks in a macroblock is top-left, top-right, bottom-left, bottom-right block for Y, followed by Cb and Cr.

Reserved, Forbidden and Marker_bit

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

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.

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

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 byte aligned.

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 Part 1 of th	

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

```
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

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_parameter_flag	**1**	
load_intra_quantizer_matrix	**1**	
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix[64]	**8*64**	**uimsbf**
load_non_intra_quantizer_matrix	**1**	
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix[64]	**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

```
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

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

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

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

block(i) {		
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	**2-7**	**vlclbf**
if(dct_dc_size_luminance != 0)		
dct_dc_differential	**1-8**	**uimsbf**
}		
else {		
dct_dc_size_chrominance	**2-8**	**vlclbf**
if(dct_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**	**"10"**
}		
}		
}		

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 0000 01B3 in hexadecimal. It identifies the beginning of a sequence header.

horizontal_size -- The horizontal_size is the width of the displayable part of each luminance picture in pels. The width of the encoded luminance picture 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 each luminance picture in pels.
The height of the encoded luminance picture 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	0.9375	CCIR601, 625line
1001	0.9815	
1010	1.0255	
1011	1.0695	
1100	1.1250	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

bit_rate -- This is an integer specifying the bit rate of the bit stream measured in units of 400 bits/second, 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 2-Annex C) buffer needed to decode the sequence. It is defined as:

B = 16 * 1024 * vbv_buffer_size

where B is the minimum VBV buffer size in bits required to decode the sequence (see 2-Annex C).

constrained_parameters_flag -- This is a one-bit flag which is 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 per second.
forward_f_code <= 4 (forward_f <= 8)
backward_f_code <= 4 (backward_f <= 8)

If the constrained_parameters_flag is set, then then the vbv_buffer_size field shall indicate a VBV buffer size less than 327680 bits (20*1024*16).

If the constrained_parameters_flag is set, then the bit_rate field shall indicate a coded data rate less than or equal to 1856000 bits per second.

load_intra_quantizer_matrix -- This is a one-bit flag which is set to "1" if 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.

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 the clause on macroblock decoding, replace the default values shown in the same clause for decoding of intra-coded macroblocks. The value for intra_quant[0][0] shall always be 8. For the 8-bit unsigned integers, the value zero is forbidden. 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 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 the clause on macroblock decoding, replace the default values shown in the same clause for decoding of non-intra-macroblocks. For the 8-bit unsigned integers, 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.

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 for "time and control codes for video tape recorders" (see Bibliography, 2-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 29.97Hz would be rounded to and counted as 30Hz. 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 is set to "1" if the group of pictures has been encoded without prediction vectors pointing to the 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-Pictures 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 sub-Clause 2.4.3.2.

group_extension_data -- Reserved.

user_data_start_code -- See sub-Clause 2.4.3.2.

user_data -- See sub-Clause 2.4.3.2.

2.4.3.4 Picture Layer

picture_start_code -- The picture_start_code is the bit string 00000100 in hexadecimal.

temporal_reference -- The temporal_reference is an 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 ommited 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 (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 -- For constant bitrate operation, the vbv_delay is used to set the initial occupancy of the decoder's buffer at the start of play 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:

$$vbv\_delay_n = 90000 * B_n^* / R$$

where:

n > 0

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

R = bit rate as defined by **bit_rate** in the sequence header.

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 Clause 2.4.4.2.

forward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. 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

forward_r_size and forward_f are used in the process of decoding the forward motion vectors as described in Clause 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 Clause 2.4.4.3.

backward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. 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

backward_r_size and backward_f are used in the process of decoding the backward motion vectors as described in Clause 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 is 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 sub-Clause 2.4.3.2.

user_data -- See sub-Clause 2.4.3.2.

2.4.3.5 Slice Layer

slice_start_code -- The slice_start_code is the bit string 000001 in hexadecimal followed by a slice_vertical_position in the range 01 through AF hexadecimal.

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. 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 is 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 2-Annex B Table 2-B1 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 Clause 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 2-B2a through 2-B2d.

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 2-B4 in 2-Annex B. The decoded value is required (along with forward_f) to decide whether or not motion_horizontal_forward_r appears in the bit stream.

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

motion_vertical_forward_code -- motion_vertical_forward_code is decoded according to table 2-B4 in 2-Annex B. The decoded value is required (along with forward_f) to decide whether or not motion_vertical_forward_r appears in the bit stream.

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

motion_horizontal_backward_code -- motion_horizontal_backward_code is decoded according to table 2-B4 in 2-Annex B. The decoded value is required (along with backward_f) to decide whether or not motion_horizontal_backward_r appears in the bit stream.

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

motion_vertical_backward_code -- motion_vertical_backward_code is decoded according to table 2-B4 in 2-Annex B. The decoded value is required (along with backward_f) to decide whether or not motion_vertical_backward_r appears in the bit stream.

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

coded_block_pattern -- The variable cbp is derived from the coded_block_pattern using the variable length code table 2-B3 in 2-Annex B. 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 difference block Cb is to be received in this macroblock.

pattern_code[5] -- If 1, then the chrominance difference 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, dct_dc_size, is derived according to the VLC table 2-B5a.

dct_dc_size_chrominance -- The number of bits in the following dct_dc_differential code, dct_dc_size, is derived according to the VLC table 2-B5b.

dct_dc_differential -- A variable length unsigned integer. If dct_dc_size is zero, then dct_dc_differential is not present in the bit stream, and the first element of the zigzag-scanned quantized DCT coefficient list, dct_zz[0], is equal to zero. If dct_dc_size is greater than zero, then dct_zz[0] is computed as follows from dct_dc_differential:

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

dct_zz[i], i>0 shall be set to zero initially.

example for dct_dc_size=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 2-B.5c through 2-B.5g in 2-Annex B for the first coefficient. 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 2-B5c through 2-B5g is used to represent the run-length and level of the DCT coefficients.

dct_coeff_next -- A variable length code according to tables 2-B...5c through 2-B.5g in 2-Annex B for coefficients following the first retrieved. 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.

2.4.4 The Video Decoding Process

2.4.4.1 Intra-coded Macroblocks

In I-pictures all blocks 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 discussion of semantics has defined mb_row and mb_column, which locate the macroblock in the picture. The definitions of dct_dc_differential, and dct_coeff_next have also 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[8][8] to be the matrix of reconstructed DCT coefficients, 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 1024 (128*8).

Define intra_quant[8][8] to be the intra quantizer matrix that is specified in the sequence header.

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

Define non_intra_quant[8][8] to be the non-intra quantizer matrix that is specified in the sequence header.

Define scan[8][8] to be the matrix defining the zigzag scanning sequence as follows:

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

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[8][8] 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] > 2047)  dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
        }
}
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] ;
```

For the following 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] > 2047)  dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
                }
        }
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] > 2047)  dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
        }
}
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] > 2047)  dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
        }
}
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 ;
```

Once the DCT coefficients are reconstructed, the inverse DCT transform defined in 2-Annex A shall be applied to obtain the inverse transformed pel values in the range [-256, 255]. These pel values shall be clipped 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 bu 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, 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.

```
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 pels of the past picture referenced by the forward motion vector, and pel[][] as the pels of the picture 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 ;
```

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 ;
        }
}
```

Note that 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 2-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. However, for B-coded pictures the previous reconstructed motion vectors shall be reset only for the first macroblock in a slice, and 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.

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

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

which 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 pels of the decoded picture 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 the predictive-coded macroblock clause. 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 ;
```

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] > 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 ;
        }
}
```

Note that 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 2-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

Some macroblocks are not stored, that is neither motion vector information nor DCT information is available to the decoder. These macroblocks occur when the macroblock_address_increment is greater than 1. The macroblocks for which no data is stored are called "skipped macroblocks".

In I-pictures, all macroblocks are coded and there are 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.

2-ANNEX A (normative)

8 by 8 INVERSE DISCRETE COSINE TRANSFORM

The 8 by 8 inverse discrete transform shall conform to IEEE Draft Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, P1180/D2, July 18, 1990. Note that Clause 2.3 P1180/D2 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. The maximum refresh period requirement for this standard shall be 132 pictures, the same as indicated in P1180/D2 for visual telephony according to CCITT Recommendation H.261 (see Bibliography).

2-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.

2-B.1 Macroblock Addressing

Table 2-B.1. Variable length codes for macroblock_address_increment.

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

2-B.2 Macroblock Type

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

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

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

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

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

2-B.3 Macroblock Pattern

Table 2-B.3. Variable length codes for coded_block_pattern.

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

2-B.4 Motion Vectors

Table 2-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

2-B.5 DCT Coefficients

Table 2-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 2-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 2-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

NOTE1 - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.
NOTE2 - This code shall be used for dct_coeff_first.
NOTE3 - This code shall be used for dct_coeff_next.

Table 2-B.5d. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

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 2-B.5e. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

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

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

Table 2-B.5f. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

dct_coeff_first and dct_coeff_next variable length code (NOTE)	run	level
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 2-B.5g. Encoding of run and level following escape code as a 20-bit fixed length code (-127 <= level <= 127) or as a 28-bit fixed length code (-255 <= level <= -128, 128 <= level <= 255).

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

2-ANNEX C (normative)

VIDEO BUFFERING VERIFIER

Constant rate coded video bit streams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in Clause 2-C.1.

The VBV is a hypothetical decoder which is conceptually connected to the output of an encoder. Coded data is placed in the buffer at the constant bit rate that is being used. Coded data is removed from the buffer as defined in Clause 2-C.1.4, below. It is a requirement of the encoder (or editor) that the bit stream it produces will not cause the VBV to either overflow or underflow.

2-C.1 Video Buffering Verifier

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

2. The VBV has a receiving buffer of size B, where B is given in the vbv_buffer_size field in the sequence header.

3. The VBV is initially empty. It is filled from the bitstream for the time specified by the vbv_delay field in the video bitstream.

4. All of the data for the picture which has been in the buffer longest is instantaneously removed. Then after each subsequent picture interval all of the data for the picture which (at that time) has been in the buffer longest is instantaneously removed. Sequence header and group of picture layer data elements which immediately precede a picture are removed at the same time as that picture. The VBV is examined immediately before removing any data (sequence header data, group of picture layer or picture) and immediately after each picture 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 on the video bit stream including coded picture data, user data and all stuffing.

To meet these requirements the number of bits for the (n+1)'th coded picture d_{n+1} (including any preceding sequence header and group of picture layer data elements) must satisfy:

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

$$d_{n+1} <= B_n + R/P$$

where:

$n >= 0$
B_n is the buffer occupancy just after time t_n
R = bitrate
P = number of pictures per second
t_n is the time when the n'th coded picture is removed from the VBV buffer

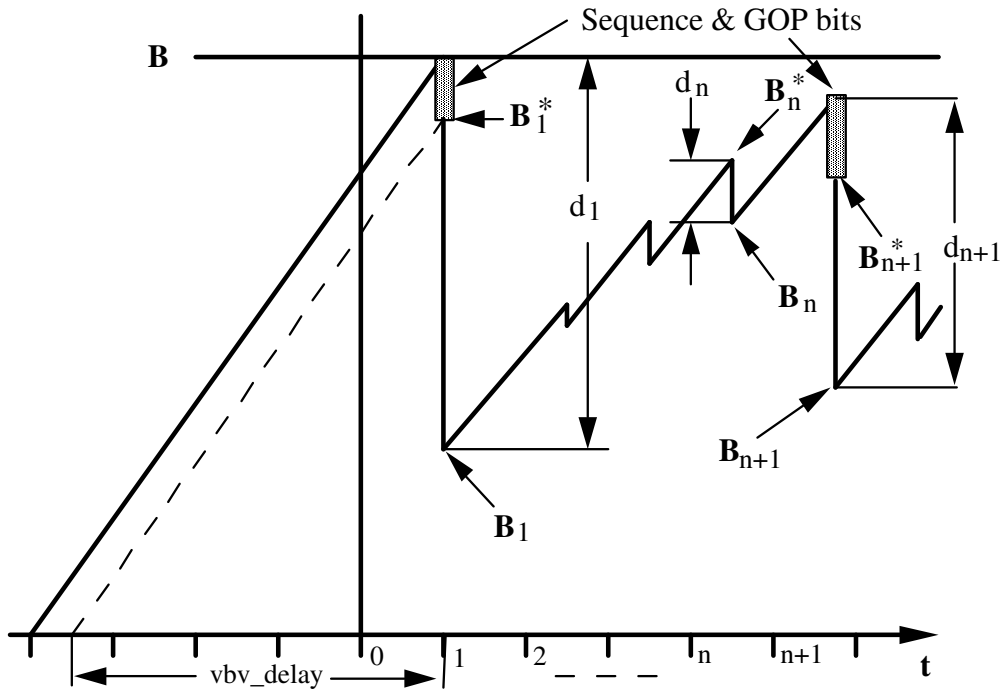


Figure 2-C.1 VBV Buffer Occupancy

2-ANNEX D (informative)

GUIDE TO ENCODING VIDEO

2-D.1 INTRODUCTION

This annex provides background material to help readers understand and implement Part 2 of this standard. The normative clauses of the standard 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 the coding methods, giving examples, and discussing encoding and decoding algorithms which are not directly covered by the standard.

The normative clauses specify the bitstream in such a way that it is fairly straightforward to design a conforming 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. The standard 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.

The MPEG standard 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 Mbps, and the MPEG standard 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 Mbps. However, it should be noted that the standard 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 [6], and ISO 10918 by the ISO JPEG committee aimed at the coding of still pictures [8]. Elements of both of these standards were incorporated into the MPEG video standard, but subsequent development work by the committee resulted in coding elements found in neither. Le Gall [2] gives an account of the method by which the MPEG video committee developed the standard, and a summary of the standard itself.

2-D.2 OVERVIEW

2-D.2.1 Video Concepts

The MPEG video standard defines a format for compressed digital video. This annex describes some ways in which practical encoders and decoders might be implemented.

Although the MPEG video standard is quite flexible, the basic algorithms have been tuned to work well at data rates of about 1 to 1.5 Mbps, at spatial resolutions of about 350 pel horizontally by about 250 pels vertically, and picture rates of about 24 to 30 pictures per second. The use of the word "picture" as opposed to "frame" is deliberate. MPEG video 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.

The MPEG video standard is designed to permit several methods of viewing coded video which are normally associated with VCRs: forward play, 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:

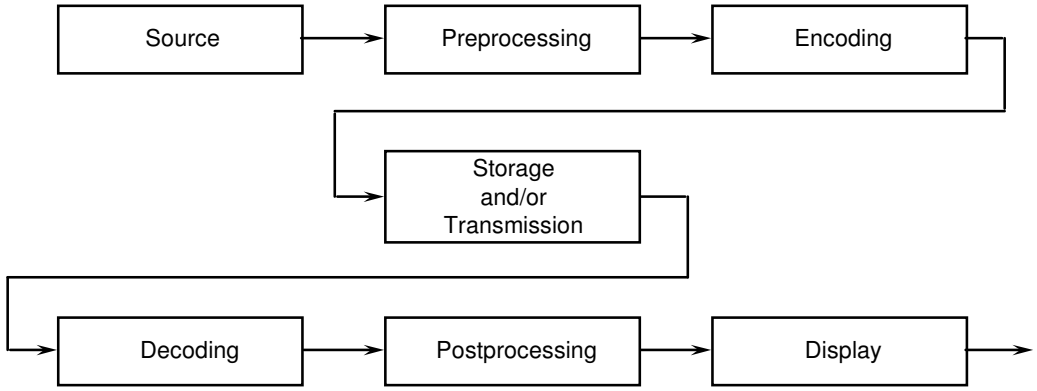


Figure 2-D.1 Coding and Decoding Process

Figure 2-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 2-D.3 of this annex 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 Clause 2-D.4, and rate control is described in Clause 2-D.6.1 The structure of an MPEG video bitstream is covered in Clause 2-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 picture, the decoder accesses the MPEG video bitstream, and decodes it as described in Clause 2-D.7. Postprocessing for display is described in Clause 2-D.8.

2-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 color representation for each pel consists of three components: value 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.

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 discards pel values systematically based on location, thus reducing the amount of information to be compressed by other techniques.The standard retains one set of chrominance pels for each 2x2 neighborhood of luminance pels.

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.

Predictive Coding

Predictive coding is a technique to improve the compression through statistical redundancy. Based on values of pels previously decoded, both the encoder and decoder can estimate or predict the value of a pel yet to be encoded or decoded. The difference between the predicted and actual values is encoded. This difference value is the prediction error which the decoder can use to correct the prediction. Most error values will be small and cluster around the value 0 since pel values typically do not have large changes within a small spatial neighborhood. The probability distribution of the prediction error is skewed and compresses better than the distribution of the pel values themselves. Additional information can be discarded by quantizing the prediction error.

Motion Compensation

Motion compensation (MC) predicts the value of a block of neighboring pels in a picture by relocating a block of neighboring pel values from a known picture. The motion is described in terms of the two-dimensional motion vector that translates the block to the new location. The simplest example is a scene where the camera is not moving, and no objects in the scene are moving. The pel values at each image location remain the same, and the motion vector for each block is 0. In general, the encoder must transmit a motion vector for each block. The translated block from the known picture becomes a prediction for encoding the block in the picture to be encoded. The technique relies on the fact that within a short sequence of pictures of the same general scene, many objects remain in the same location while others may move a short distance.

Frequency Transformation

The discrete cosine transform (DCT) converts an 8 by 8 block of pel values to an 8 by 8 matrix of horizontal and vertical spatial frequency coefficients. An 8 by 8 block of pel values can be reconstructed by performing the inverse discrete cosine transform (IDCT) on the spatial frequency coefficients. In general, most of the energy is concentrated in the low frequency coefficients, which are located in the upper left corner of the transformed matrix. Compression is achieved by a quantization step, where the quantization intervals are identified by an index. Since the encoder identifies the interval and not the exact value within the interval, the pel values of the block reconstructed by the IDCT have reduced accuracy.

The DCT coefficient in location (0,0) (upper left) of the block represents the zero horizontal and zero vertical frequency and is called the DC coefficient. The DC coefficient is proportional to the average pel value of the 8 by 8 block, and additional compression is provided through predictive coding since the difference in the average value of neighboring 8 by 8 blocks tends to be relatively small. The other coefficients represent one or more nonzero horizontal or nonzero vertical spatial frequencies, and are called AC coefficients. The quantization level of the coefficients corresponding to the higher spatial frequencies favors the creation of an AC coefficient of 0 by choosing a quantization step size such that the HVS is unlikely to perceive the loss of the particular spatial frequency unless the coefficient value lies above the particular quantization level. The statistical encoding of the expected runs of consecutive zero-valued coefficients of higher-order coefficients accounts for considerable compression gain. To cluster nonzero coefficients early in the series and encode as many zero coefficients as possible following the last nonzero coefficient in the ordering, the coefficient sequence is specified to be a zig-zag ordering. - See Figure 2-D.25 - The ordering concentrates the highest spatial frequencies at the end of the series.

Variable-Length Coding

Variable-length coding (VLC) is a statistical coding technique that assigns codewords to values to be encoded. Values of high frequency of occurrence are assigned short codewords, and those of infrequent occurrence are assigned long codewords. On average, the more frequent shorter codewords dominate such that the code string is shorter than the original data.

Picture Interpolation

If the decoder reconstructs a picture from the past and a picture from the future, then the intermediate pictures can be reconstructed by the technique of interpolation, or bidirectional prediction. Macroblocks in the intermediate pictures can be forward and backward predicted and translated by means of motion vectors. The decoder may reconstruct pel values belonging to a given macroblock as an average of values from the past and future pictures.

2-D.2.3 Bitstream Hierarchy

The MPEG video coding scheme is arranged in layers corresponding to a hierarchical structure. A **sequence** is the top layer of the coding hierarchy and consists of a header and some number of **groups-of-pictures** (**GOP**s). The sequence header initialises the state of the decoder. This allows decoders to decode any sequence without being affected by a past decoding history.

A **GOP** is a random access point, i.e. it is the smallest coding unit that can be independently decoded within a sequence, and consists of a header and some number of **pictures**. The GOP header contains time and editing information.

A **picture** corresponds to a single frame of motion video, or to a movie frame. There are four picture types: **I** pictures, or *intra* pictures, which are coded without reference to any other pictures; **P** pictures, or *predicted* pictures, which are coded using motion compensation from a previous I or P picture; **B** pictures, or *bidirectionally* predicted pictures, which are coded using motion compensation from a previous and a future I or P picture, and **D** pictures, or *DC* pictures, which are intended only for a fast forward search mode. A typical coding scheme contains a mix of I, P, and B pictures. Typically, an I picture may occur every half a second, to give reasonably fast random access, with two B pictures inserted between each pair of I or P pictures.

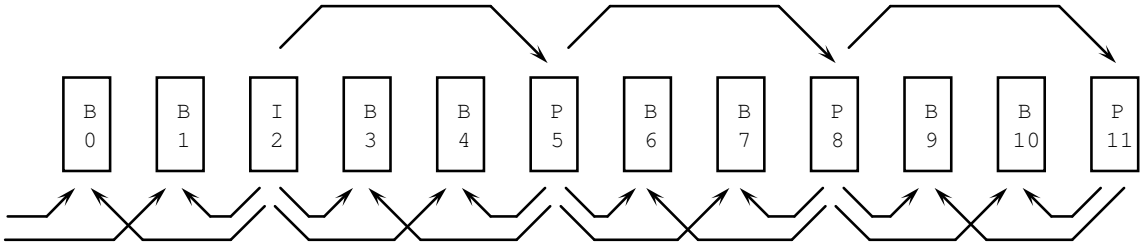


Figure 2-D.2 Dependency Relationship Between I, B, and P Pictures

Figure 2-D.2 illustrates a number of pictures in display order. The arrows show the dependency relationship of the predicted and bidirectionally predicted pictures.

Note that because of the picture dependencies, the bitstream order, i.e. the order in which pictures are transmitted, stored, or retrieved, is not the display order, but rather the order which the decoder requires them to decode the bitstream. For example, a typical sequence of pictures, in display order, might be:

Figure 2-D.3 Typical Sequence of Pictures in Display Order

whereas the bitstream order would be as shown below:

Figure 2-D.4 Typical Sequence of Pictures in Bitstream Order

Because the B pictures depend on the following (in display order) P picture, the P picture must be transmitted and decoded before the dependent B pictures.

Pictures consist of a header and one or more **slices.** The picture header contains time, picture type, and coding information.

A **slice** provides some immunity to data corruption. Should the bitstream becomes unreadable within a picture, the decoder should be able to recover by waiting for the next slice, without having to drop an entire picture.

Slices consist of a header and one or more **macroblocks.** The slice header contains position and quantizer scale information. This is sufficient for recovery from local corruption.

A **macroblock** is the basic unit for motion compensation and quantizer scale changes.

Each macroblock consists of a header and six component **blocks**: four blocks of luminance, one block of Cb chrominance, and one block of Cr chrominance. See Figure 2-D.5. The macroblock header contains quantizer scale and motion compensation information.

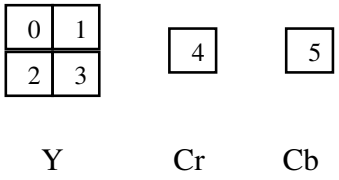


Figure 2-D.5 Macroblock Structure

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 model sensitivity of the human visual system.

Blocks are the basic coding unit, and the DCT is applied at this block level. Each block contains 64 component **pels** arranged in an 8 by 8 array as shown in Figure 2-D.6. Note that pel values are not individually coded, but are components of the coded block.

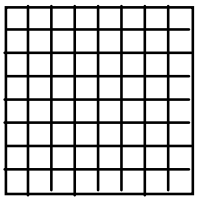


Figure 2-D.6 Block Structure

Each luminance pel corresponds to one picture pel, but since the chrominance information is subsampled with a 2:1 ratio both horizontally and vertically, each chrominance pel corresponds to 4 picture pels.

2-D.2.4 Decoder Overview

A simplified block diagram of a possible decoder implementation is shown below:

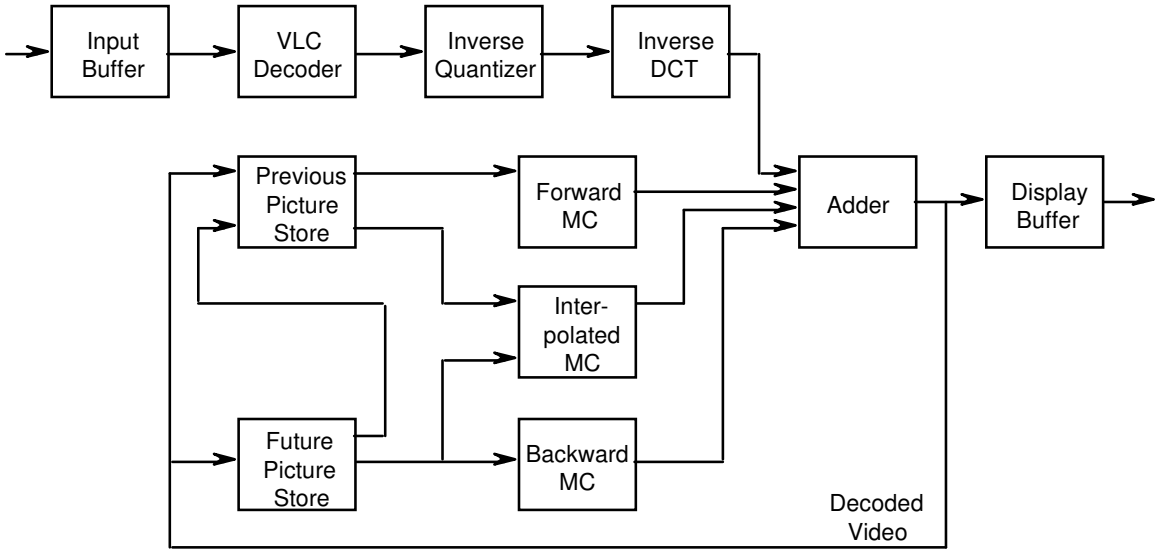


Figure 2-D.7 Simplified Decoder Block Diagram

It is instructive to follow the method which the decoder uses to decode a bitstream containing the sequence of pictures given in Fig 2.4, and display them in the order given in Fig 2.3. The following description is simplified for clarity.

The input bitstream is accumulated in the Input Buffer until needed. The Variable Length Code (VLC) Decoder decodes the header of the first picture, picture 0, and determines that it is an I picture. The VLC Decoder produces quantized coefficients corresponding to the quantized DCT coefficients. These are assembled for each 8 by 8 block of pels in the image. The Inverse Quantizer produces the actual DCT coefficients using the quantization step size. The coefficients are then transformed into pel values by the Inverse DCT transformer and stored in the Previous Picture Store and the Display Buffer. The picture may be displayed at the appropriate time.

The VLC Decoder decodes the header of the next picture, picture 3, and determines that it is a P picture. For each block, the VLC Decoder decodes motion vectors giving the displacement from the stored previous picture, and quantized coefficients corresponding to the quantized DCT coefficients of the difference block. These quantized coefficients are inverse quantized produces the actual DCT coefficients. The coefficients are then transformed into pel difference values and added to the predicted block produced by applying the motion vectors to blocks in the stored previous picture. The resultant block is stored in the Future Picture Store and the Display Buffer. This picture cannot be displayed until B pictures 1 and 2 have been received, decoded, and displayed.

The VLC Decoder decodes the header of the next picture, picture 1, and determines that it is a B picture. For each block, the VLC decoder decodes motion vectors giving the displacement from the stored previous or future pictures or both, and quantized coefficients corresponding to the quantized DCT coefficients of the difference block. These quantized coefficients are inverse quantized to produce the actual DCT coefficients. The coefficients are then transformed into difference pel values and added to the predicted block produced by applying the motion vectors to the stored pictures. The resultant block is then stored in the Display Buffer. It may be displayed at the appropriate time.

The VLC Decoder decodes the header of the next picture, picture 2, and determines that it is a B picture. It is decoded using the same method as for picture 1. After decoding picture 2, picture 0, which is in the Previous Picture Store, is no longer needed and may be discarded.

The VLC Decoder decodes the header of the next picture, picture 6, and determines that it is a P picture. The picture in the Future Picture Store is copied into the Previous Picture Store, then decoding proceeds as for picture 3. Picture 6 should not be displayed until pictures 4 and 5 have been received and displayed.

The VLC Decoder decodes the header of the next picture, picture 4, and determines that it is a B picture. It is decoded using the same method as for picture 1.

The VLC Decoder decodes the header of the next picture, picture 5, and determines that it is a B picture. It is decoded using the same method as for picture 1.

The VLC Decoder decodes the header of the next picture, picture 9, and determines that it is a P picture. It then proceeds as for picture 6.

The VLC Decoder decodes the header of the next picture, picture 7, and determines that it is a B picture. It is decoded using the same method as for picture 1.

The VLC Decoder decodes the header of the next picture, picture 8, and determines that it is a B picture. It is decoded using the same method as for picture 1.

The VLC Decoder decodes the header of the next picture, picture 12, and determines that it is an I picture. It is decoded using the same method as for picture 0. This process is repeated for the subsequent pictures.

2-D.3 PREPROCESSING

The source material may exist in many forms, e.g. computer files or CCIR 601 format, but in general, it must be processed before being encoded. This clause discusses some aspects of preprocessing.

For a given data rate and source material, there is an optimum picture rate and spatial resolution at which to code if the best perceived quality is desired. If the resolution is too high, then too many bits will be expended on the overhead associated with each block leaving too few to code the values of each pel accurately. If the resolution is too low, the pel values will be rendered accurately, but high frequency detail will be lost. The optimum resolution represents a tradeoff between the various coding artifacts (e.g. noise and blockiness) and the perceived resolution and sharpness of the image. This tradeoff is further complicated by the unknowns of the final viewing conditions, e.g. screen brightness and the distance of the viewer from the screen.

At data rates of 1 to 1.5 Mbps, reasonable choices for the picture rate include 24, 25 and 30 pictures per ssecond, for the horizontal resolution is between 250 and 400 pels, and for the vertical resolution is between 200 and 300 pels. Note that these values are not normative and other picture rates and resolutions are valid.

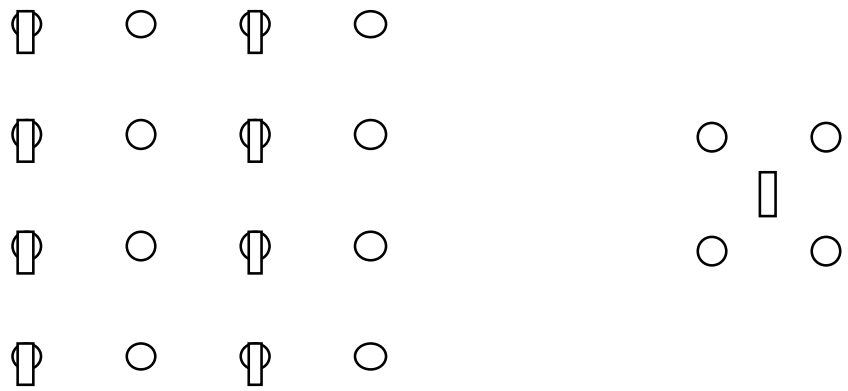
2-D.3.1 Conversion from CCIR 601 Video to MPEG SIF

A popular source resolution is that specified by CCIR 601 [5]. This international standard for digital TV consists of component coded video Y, Cb, and Cr. Y is the luminance or luminance signal and gives gray scale video. Cb and Cr are two independent color signals. There are two options in the number of lines, picture rate, and pel aspect ratio. One option has 525 lines per frame at 60 Hz field rate, another has 625 lines per frame at 50 Hz field rate. The luminance pels are sampled at resolutions of 720x480 and 720x576 respectively.

These field rates and resolutions are too large for effective coding at data rates between 1 and 1.5 Mbps. More appropriate values are a picture rate equal to the frame rate, and a linear resolution half that of a frame. This reduces the pel rate by a factor of four.

One way of converting the source video rate is to use only the odd or even fields. This reduces the picture rate to 25 or 30 Hz. If the other field is simply discarded, spatial aliasing will be introduced, and this may produce visible and objectionable artifacts. More sophisticated methods of rate conversion require more computational power, but can perceptibly reduce the aliasing artifacts.

The horizontal and vertical resolutions may be halved by filtering and subsampling. Consider a picture in the 4:2:2 format. See the CCIR 601 sampling pattern of Figure 2-D.8(a). Such a sampling pattern may be converted to the 4:2:0 SIF sampling pattern of Figure 2-D.8(b) as follows. The odd field may be sampled, reducing the number of lines by two, and then a horizontal decimation filter is used on the remaining lines to reduce the horizontal resolution by a factor of two. In addition the chrominance values are vertically decimated. The filter for luminance and chrominance have to be chosen carefully since particular attention has to be given to the location of the samples in the respective standards.



(a) Sampling pattern for 4:2:2 (CCIR 601) (b) Sampling pattern for MPEG (SIF)
Circles represent luminance; Boxes represent Chrominance

Figure 2-D.8 Conversion of CCIR 601 to SIF

The following odd number of tap FIR filter has been found to give good results in decimating the luminance:

-29	0	88	138	88	0	-29

// 256

Figure 2-D.9 Luminance Subsampling Filter Tap Weights

Use of a power of two for the divisor allows a simple hardware implementation.

The chrominance samples have to be placed at a horizontal position in the middle of the luminance samples. A linear filter with a phase shift of 1/2 a sample is used for this task e.g.

1	3	3	1

// 8

Figure 2-D.10 Chrominance Subsampling Filter Tap Weights

To recover the samples consistent with the CCIR 601 grid of Figure 2-D.8(a), the process of interpolation is used. The interpolation filter applied to a zero-padded signal can be chosen to be equal to the decimation filter employed for the luminance and the two chrominance values in the encoder.

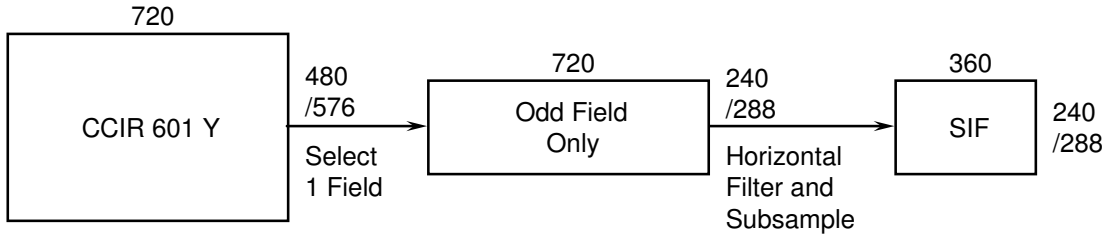
Note that these filters are not part of the standard, and other filters may be used.

At the end of the lines some special technique such as renormalizing the filter or replicating the last pel, must be adopted. The following example shows a horizontal line of 16 luminance pels and the same line after filtering and subsampling:

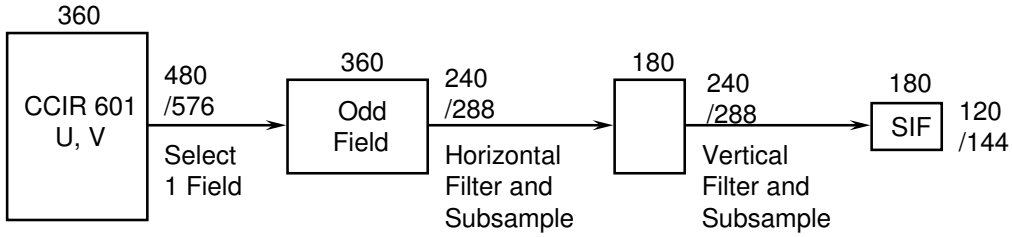
10	12	20	30	35	15	19	11	11	19	26	45	80	90	92	90
	12		32		23		9		12		49		95		95

Figure 2-D.11 Example of Filtering and Subsampling of a Line of Pixels

The result of this filtering and subsampling is a source input format (SIF) which has a luminance resolution of 360 x 240 or 360 x 288, and a chrominance resolution which is half that of the luminance in each dimension:



(a) Luma



(b) Chroma

Figure 2-D.12 Conversion from CCIR 601 into SIF

The SIF is not quite optimum for processing by MPEG video coders. MPEG video divides the luminance component into macroblocks of 16x16 pels. The horizontal resolution, 360, is not divisible by 16. A better match is obtained by discarding the leftmost 4 pels and the rightmost 4 pels from each line of the subsampled picture. The remaining picture is called the significant pel area, and corresponds to the shaded area in Figure 2-D.13:

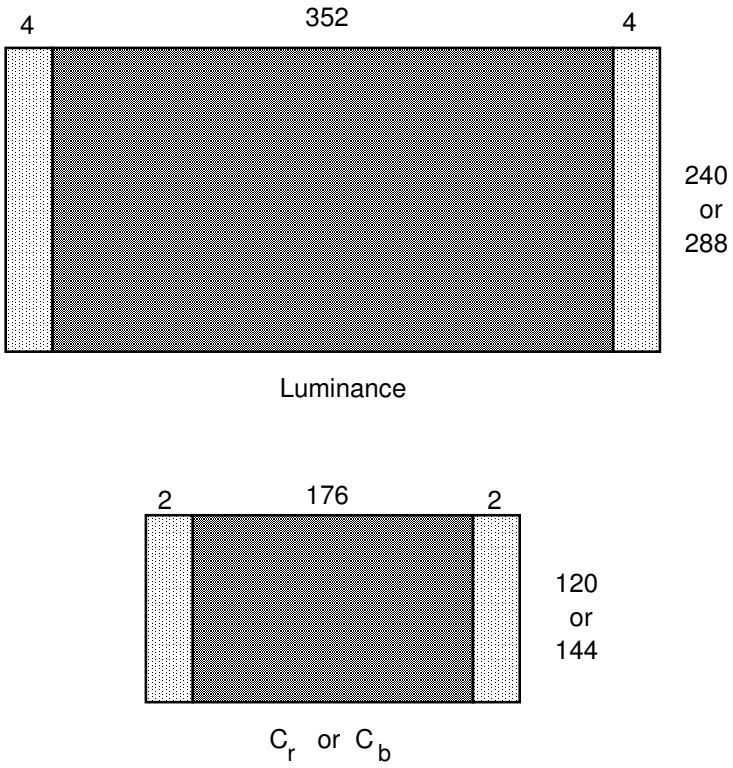


Figure 2-D.13 Source Input with Significant Pixel Area Shaded Dark

The conversion process is summarized in the following table:

Picture Rate (Hz)	30	25
Picture Aspect Ratio (width:height)	4:3	4:3
Luminance (Y)		
CCIR Sample Resolution	720 x 480	720 x 576
SIF	360 x 240	360 x 288
Significant Pixel Area	352 x 240	352 x 288
Chroma (Cb Cr)		
CCIR Sample Resolution	360 x 480	360 x 576
SIF	180 x 120	180 x 144
Significant Pixel Area	176 x 120	176 x 144

Table 2-D.1 Conversion of Source Formats

The preprocessing into the SIF format is not normative, other processing steps and other resolutions may be used. The picture size need not even be a multiple of 16. In this case an MPEG video coder adds padding pels to the right or bottom edges of a picture in order to bring the transmitted resolution up to a multiple of 16, and the decoder discards these after decoding the picture. For example, a horizontal resolution of 360 pels would be coded by adding 8 padding pels to the right edge of each horizontal row bringing the total up to 368 pels. 23 macroblocks would be coded in each row. The decoder would discard the extra padding pels after decoding, giving a final decoded horizontal resolution of 360 pels.

2-D.3.2 Conversion from Film

If film material can be digitized at 24 pps, then it forms an excellent source for an MPEG video bitstream. It may be digitized at the desired spatial resolution. The picture_rate field in the video sequence header, see Clause 2.4.2.3, allows the picture rate of 24 pps to be specified exactly.

Sometimes the source material available for compression consists of film material which has been converted to video at some other rate. The encoder may detect this and recode at the original film rate. For example, 24 pps film material may have been digitized and converted to a 30 fps system by the technique of 3:2 pulldown. In this mode digitized pictures are shown alternately for 3 and for 2 television field times. This alternation may not be exact since the actual frame rate might be 29.97 fps and not the 30 fps that the 3:2 pulldown technique gives. In addition the pulldown timing might have been changed by editing and splicing after the conversion. A sophisticated encoder might detect the duplicated fields, average them to reduce digitization noise, and code the result at the original 24 pps rate. This should give a significant improvement in quality over coding at 30 pps, since a simple minded coding at 30 pps destroys the 3:2 pulldown timing and gives a jerky appearance to the final decoded video.

2-D.4 MODEL DECODER

2-D.4.1 Need for a Decoder Model

A coded bitstream contains different types of pictures, and each type ideally requires a different number of bits to encode. In addition, the video may vary in complexity with time and an encoder may wish to devote more coding bits to one part of a sequence than to another. For constant bit rate coding, varying the number of bits allocated to each picture requires that the decoder have a buffer to store the bits not needed to decode the immediate picture. The extent to which an encoder can vary the number of bits allocated to each picture depends on the size of this buffer. If the buffer is large an encoder can use greater variations, increasing the picture quality, but at the cost of increasing the decoding delay. The delay is the time taken to fill the input buffer from empty to its current level. Encoders need to know the size of the decoders input buffer in order to determine to what extent they can vary the distribution of coding bits among the pictures in the sequence.

In constant bit rate applications (for example decoding a bitstream from a CD ROM), problems of synchronization may occur. In these applications the encoder should generate a bitstream that is perfectly matched to the device. The decoder will display the decoded pictures at their specified rate. If the display clock is not locked to the channel data rate, and this is typically the case, then any mismatch between the encoder and channel clock and the display clock will eventually cause a buffer overflow or underflow problem. For example, assume that the display clock runs one part per million too slow with respect to the channel clock. If the data rate is one million bits per second then the input buffer will fill at an average rate of one bit per second, eventually causing an overflow problem. If the encoder uses all the buffer to allocate bits between pictures, the overflow could occur quite quickly. For example, suppose the encoder fills the buffer completely except for one byte at the start of each picture, then overflow will occur after only eight seconds!

The model decoder is defined to solve three problems. It constrains the variability in the number of bits that may be allocated to different pictures; it allows a decoder to initialise its buffer when the system is started; and it allows the decoder to maintain synchronisation while the stream is played. It should be noted that Part 1 of this standard addresses the initialisation of buffers and the maintenance of synchronisation during play in the case when more than two or elementary streams (for example one audio and one video stream) are multiplexed together. The tools defined in Part1 of for the maintenance of synchronisation should be used by decoders when multiplexed streams are being played.

2-D.4.2 Decoder Model

2-Annex C contains defines a parameterized model decoder for this purpose. It is known as a Video Buffer Verifier (VBV). The parameters used by a particular encoder are defined in the bitstream. This really defines a model decoder that is needed if encoders are to be assured that the coded bitstreams they produce will be decodable. The model decoder looks like this:

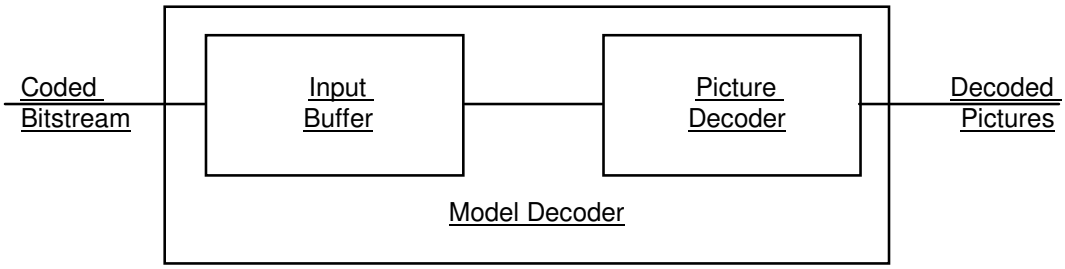


Figure 2-D.14 Model Decoder

A fixed-rate channel is assumed to put bits at a constant rate into the Input Buffer. At regular intervals, set by the picture rate, the Picture Decoder instantaneously removes all the bits for the next picture from the Input Buffer. If there are too few bits in the Input Buffer, i.e. all the bits for the next picture have not been received, then the Input Buffer underflows and there is an underflow error. If, during the time between picture starts, the capacity of the Input Buffer is exceeded, then there is an overflow error.

Practical decoders differ from this model in several important ways. They may not remove all the bits required to decode a picture from the Input Buffer instantaneously, they may not be able to control the start of decoding very precisely as required by the buffer fullness parameter in the picture header, and they take a finite time to decode. They may also be able to delay decoding for a short time to reduce the chances of underflow occuring. But these differences depend in degree and kind on the exact method of implementation. To satisfy requirements of different implementations, the MPEG video committee chose a very simple model for the decoder. Practical implementations of decoders must ensure that they can decode the bitstream constrained by this model. In many cases this will be achieved by using an Input Buffer that is larger than the minimum required, and by using a decoding delay that is larger than the value derived from the buffer fullness parameter. The designer must compensate for any differences between the actual design and the model in order to guarantee that the decoder can handle any bitstream that satisfies the model.

Encoders monitor the status of the model to control the encoder so that overflow problems do not occur. The calculated buffer fullness is transmitted at the start of each picture so that the decoder can maintain synchronization.

2-D.4.3 Buffer Size and Delay

For constant bit rate operation each picture header contains a vbv_delay parameter to enable decoders to synchronize their decoding correctly. This parameter defines the time needed to fill the Input Buffer of Figure 2-D.14 from an empty state to the correct level immediately before the Picture Decoder removes all the bits for the picture. This time is thus a delay and is measured in units of 1/90000 second. This number was chosen because it is an exact multiple of the picture durations: 1/24, 1/25, 1/29.97 and 1/30, and because it is comparable in duration to an audio sample.

The delay is given by:

$$D = \text{vbv\_delay} / 90\,000 \text{ seconds}$$

For example, if BF were 9000, then the delay would be 0.1 sec. This means that at the start of a picture the Input Buffer of the model decoder should contain exactly 0.1 seconds worth of data from the input bitstream.

The bit rate, R, is defined in the sequence header. The number of bits in the Input Buffer at the beginning of the picture is thus given by:

$$B = D * R = \text{vbv\_delay} * R / 90\,000 \text{ bits}$$

For example, if vbv_delay were 9000 and R were 1.2 Mbps, then the number of bits in the Input Buffer would be 120000.

The constrained parameter bitstream requires that the Input Buffer have a capacity of 327680 bits, and B should never exceed this value.

2-D.5 MPEG VIDEO BITSTREAM SYNTAX

This clause describes the video bitstream in a top-down fashion. A sequence is the top video level of coding. It begins with a sequence header which defines important parameters needed by the decoder. The sequence header is followed by one or more groups of pictures. Groups of pictures, as the name suggests, consist of one or more individual pictures. The sequence may contain additional sequence headers. A sequence is terminated by a sequence_end_code. this standard allows considerable flexibility in specifying application parameters such as bit rate, picture rate, picture resolution, and picture aspect ratio. These parameters are specified in the sequence header.

If these parameters, and some others, fall within certain limits, then the bitstream is called a constrained parameter bitstream. All conforming MPEG video decoders should be able to decode a constrained parameter bitstream. If encoders are to produce bitstreams intended to be decoded by decoders from other manufacturers, then they should produce only constrained parameter bitstreams to be certain that they can be decoded.

2-D.5.1 Sequence

A 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. Additional sequence headers may appear within the sequence. In each such repeated sequence header, all of the data elements with the permitted exception of those defining 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 the first sequence header. Repeating the sequence header with its data elements makes random access into the video sequence possible. The quantization matrices may be redefined as required with each repeated sequence header.

The encoder may set such parameters as the picture size and aspect ratio in the sequence header, to define the resources that a decoder requires In addition, user data may be included.

Sequence Header Code

A coded sequence begins with a sequence header and the header starts with the sequence start code. Its value is:

hex: 00 00 01 B3
binary: 0000 0000 0000 0000 0000 0001 1011 0011

This is a unique string of 32 bits that cannot be emulated anywhere else in the bitstream, and is byte-aligned, as are all start codes. To achieve byte alignment the encoder may precede the sequence start code with any number of zero bits. These can have a secondary function of preventing decoder input buffer underflow. This procedure is called *bit stuffing*, and may be done before any start code. The stuffing bits must all be zero. The decoder discards all such stuffing bits.

The sequence start code, like all video start codes, begins with a string of 23 zeros. The coding scheme ensures that such a string of consecutive zeros cannot be produced by any other combination of codes, i.e. it cannot be emulated by other codes in the bitstream. This string of zeros can only be produced by a start code, or by stuffing bits preceding a start code.

Vertical Size

This is a 12-bit number representing the height of the picture in pels, i.e. the vertical resolution. It is an unsigned integer with the most significant bit first. A value of zero is not allowed (to avoid start code emulation) so the legal range is from 1 to 4095. In practice values are usually a multiple of 16. At 1.5 Mbps, a popular vertical resolution is 240 to 288 pels. Values of

240 pels are convenient for interfacing to 525-line NTSC systems, and values of 288 pels are more appropriate for 625-line PAL and SECAM systems.

If the vertical resolution is not a multiple of 16 lines, the encoder must fill out the picture at the bottom to the next higher multiple of 16 so that the last few lines can be coded in a macroblock. The decoder will discard these extra lines before display.

For efficient coding, replicating the last line of pels is usually better than filling in the remaining pels with a grey level.

Horizontal Size

This is a 12-bit number representing the width of the picture in pels, i.e. the horizontal resolution. It is an unsigned integer with the most significant bit first. A value of zero is not allowed (to avoid start code emulation) so the legal range is from 1 to 4095. In practice values are usually a multiple of 16. At 1.5 Mbps, a popular horizontal resolution is 352 pels. The value 352 is derived from half the CCIR 601 horizontal resolution of 720, rounded down to the nearest multiple of 16 pels. Otherwise the encoder must fill out the picture on the right to the next higher multiple of 16 so that the last few pels can be coded in a macroblock. The decoder will discard these extra pels before display.

For efficient coding of the extra pels, the encoder should add pel values that reduce the number of bits generated in the transformed block . Replicating the last column of pels is usually superior to filling in the remaining pels with a gray level.

Pel Aspect Ratio

This is a four-bit number which defines the shape of the pel on the viewing screen. This is needed since the horizontal and vertical picture sizes by themselves do not specify the shape of the displayed picture.

The pel aspect ratio does not give the shape directly, but is an index to the following look up table:

CODE	HEIGHT/WIDTH	COMMENT
0000	undefined	Forbidden
0001	1.0	square pels
0010	0.6735	
0011	0.7031	16:9 625-line
0100	0.7615	
0101	0.8055	
0110	0.8437	16:9 525-line
0111	0.8935	
1000	0.9375	720x576 at 4:3 = 0.9375
1001	0.9815	
1010	1.0255	
1011	1.0695	
1100	1.1250	720x485 at 4:3 = 1.1134
1101	1.1575	
1110	1.2015	
1111	undefined	reserved

Table 2-D.2 Pixel Aspect Ratio

The code 0000 is forbidden to avoid start code emulation. The code 0001 has square pels. This is appropriate for many computer graphics systems. The code 1000 is suitable for displaying a resolution of 720 x 576 pels with a 4:3 aspect ratio. The aspect ratio can be calculated (see [5]) as:

$$\text{width / height} = 0.75 * 720 / 576 = 0.9375$$

The code 1100 is suitable for displaying a resolution of 720 x 485 pels with a 4:3 aspect ratio. The aspect ratio can be calculated as:

width / height = 0.75 * 720 / 485 = 1.1134

The code 1111 is reserved for possible future extension to the standard.

The remaining points in the table were filled in by interpolating between these two points 1000 and 1100 using the formula:

aspect ratio = 0.5855 + 0.044N

where N is the value of the code in Table 5.1. These additional aspect ratios might be useful for HDTV where ratios of 16:9 and 5:3 have been proposed.

It is evident that the specification does not allow all possible aspect ratios to be specified. We therefore presume that a certain degree of tolerance is allowable. Encoders will convert the actual aspect ratio to the nearest value in the table, and decoders will display the decoded values to the nearest aspect ratio of which they are capable.

Picture Rate

This is a four-bit integer which is an index to the following table:

CODE	PICTURES PER SECOND
0000	Forbidden
0001	23.976
0010	24
0011	25
0100	29.97
0101	30
0110	50
0111	59.94
1000	60
1001	Reserved
	.
.	.
1111	Reserved

Table 2-D.3 Picture rate

The allowed picture rates are commonly available sources of analog or digital sequences. One advantage in not allowing greater flexibility in picture rates is that standard techniques may be used to convert to the display rate of the decoder if it does not match the coded rate.

Bit Rate

The bit rate is an 18-bit integer giving the bit rate of the data channel in units of 400 bps. The bit rate is assumed to be constant for the entire sequence. The actual bit rate is rounded up to the nearest multiple of 400 bps. For example, a bit rate of 830100 bps would be rounded up to 830400 bps giving a coded bit rate of 2076 units.

If all 18 bits are 1 then the bitstream is intended for variable bit rate operation. The value zero is forbidden.

For constant bit rate operation, the bit rate is used by the decoder in conjunction with the vbv_delay parameter in the picture header to maintain synchronization of the decoder with a constant rate data channel. If the stream is multiplexed using Part 1 of this standard, the time-stamps and system clock reference information defined in Part 1 provide a more appropriate tool for performing this function.

Marker Bit

The bit rate is followed by a single reserved bit which is always set to 1. This bit prevents emulation of start codes.

VBV Buffer Size

The buffer size is a 10-bit integer giving the minimum required size of the input buffer in the model decoder in units of 16384 bits (2048 bytes). For example, a buffer size of 20 would require an input buffer of 20 x 16384 = 327680 bits (= 40960 bytes). Decoders may provide more memory than this, but if they provide less they will probably run into buffer overflow problems while the sequence is being decoded.

Constrained Parameter Flag

If certain parameters specified in the bitstream fall within predefined limits, then the bitstream is called a constrained parameter bitstream. Thus the constrained parameter bitstream provides a base level standard of performance that should be met by all decoders and not exceeded by encoders if coded bitstreams are to be exchanged between different systems.

The bit rate parameter allows values up to about 100 Mbps, but a constrained parameter bitstream must have a bit rate of 1.856 Mbps or less. Thus the bit rate parameter must be 3712 or less.

The picture rate parameter allows picture rates up to 60 pictures per second (pps), but a constrained parameter bitstream must have a picture rate of 30 pps or less.

The resolution of the coded picture is also specified in the sequence header. Horizontal resolutions up to 4095 pels are allowed by the syntax, but in a constrained parameter bitstream the resolution is limited to 720 pels or less. Vertical resolutions up to 4095 pels are allowed, but that in a constrained parameter bitstream is limited to 576 pels or less. In a constrained parameter bitstream, the total number of macroblocks is limited to 396. This sets a limit on the maximum area of the picture which is only about one quarter of the area of a 720x576 pel picture. In a constrained parameter bitstream, the pel rate is limited to 2534400 pels per second. For a given picture rate, this sets a another limit on the maximum area of the picture. If the picture has the maximum area of 396 macroblocks, then the picture rate is restricted to 25 pps or less. If the picture rate has the maximum constrained value of 30 pps the maximum area is limited to 330 macroblocks.

A constrained parameter bitstream can be decoded by a model decoder with a buffer size of 327680 bits without overflowing or underflowing during the decoding process. The maximum buffer size that can be specified for a constrained parameter bitstream is 20 units.

If all these conditions are met, then the bitstream is constrained and the constrained_parameters_flag in the sequence header should be set to 1. If any parameter is exceeded, the flag should be set to 0 to inform decoders that more than a minimum capability is required to decode the sequence.

Load Intra Quantizer Matrix

This is a one-bit flag. If it is set to 1, 64 8-bit integers follow. These define an 8 by 8 set of weights which are used to quantize the DCT coefficients. They are transmitted in the zigzag scan order shown in Figure 2-D.25. None of these weights can be zero. The first weight must be eight which matches the fixed quantization level of the DC coefficient.

If the flag is set to zero, the intra quantization matrix must be reset to the following default value:

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

Figure 2-D.15 Default intra quantization matrix

The default quantization matrix was derived by the MPEG video committee from the default quantization matrix developed by the JPEG committee based on a series of experiments [9]. Experience has shown that it gives good results over a wide range of video material. For resolutions close to 350x250 there should normally be no need to redefine the intra quantization matrix. If the picture resolution departs significantly from this nominal resolution, then some other matrix may give perceptibly better results.

The weights increase to the right and down. This reflects the human visual system which is less sensitive to quantization noise at higher frequencies.

Load Non–Intra Quantizer Matrix

This is a one-bit flag. If it is set to 1, 64 8-bit integers follow in zigzag scan order. None of these integers can be zero.

If the flag is set to zero, the non-intra quantization matrix must be reset to the following default value which consists of all 16s.

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

Figure 2-D.16 Default non-intra quantization matrix

This flat default quantization matrix was adopted from H.261 which uses a flat matrix for the equivalent of P pictures [6]. Little work has been done to determine the optimum non-intra matrix for MPEG video coding, but evidence suggests that it is more dependent on video material than is the intra matrix. The optimum non-intra matrix may be somewhere between the flat default non-intra matrix and the strongly frequency-dependent values of the default intra matrix.

Extension Data

This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B5
binary: 0000 0000 0000 0000 0000 0001 1011 0101

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes are reserved for future extensions to the MPEG video standard, and should not be generated by encoders.MPEG video decoders should have the capability to discard ay extension data found.

User Data

A user data start code may follow the optional extension data. This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B2
binary: 0000 0000 0000 0000 0000 0001 1011 0110

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes can be used by the encoder for any purpose. The only restriction on the data is that they cannot emulate a start code, even if not byte aligned. This means that a string of 23 consecutive zeros must not occur. One way to prevent emulation is to force the most significant bit of alternate bytes to be a 1.

In closed encoder-decoder systems the decoder may be able to use the data. In the more general case decoders should be capable of discarding the user data.

2-D.5.2 Group of Pictures

Two distinct picture orderings exist, the display order and the bitstream order (as they appear in the video bitstream). A group of pictures (gop) is a set of pictures which are contiguous in display order. A group of pictures must contain at least

one I picture. This required picture may be followed by any number of I and P pictures. Any number of B pictures may be interspersed between each pair of I or P pictures, and may also precede the first I picture.

Property 1. A group of pictures, in bitstream order, must start with an I picture and may be followed by any number of I, P or B pictures in any order.

Property 2. Another property of a group of pictures is that it must begin, in display order, with an I or a B picture, and must end with an I or a P picture. The smallest group of pictures consists of a single I picture, whereas the largest size is unlimited.

The original concept of a group of pictures was a set of pictures that could be coded and displayed independently of any other group. In the final version of the standard this is not always true, and any B pictures preceding (in display order) the first I picture in a group may require the last picture in the previous group in order to be decoded. Nevertheless encoders can still construct groups of pictures which are independent of one another. One way to do this is to omit any B pictures preceding the first I picture. Another way is to allow such B pictures, but to code them using only backward motion compensation.

Property 3. From a coding point of view, a concisely stated property is that a group of pictures begins with a group of pictures header, and either ends at the next group of pictures header or at the next sequenceheader or at the end of sequence, whichever comes first.

Some examples of groups of pictures are given below:

I
I P P
I B **P** B **P**
B B **I** B **P** B **P**
B B **I** B B **P** B B **P** B B **P**
B **I** B B B B **P** B **I** B B **I** **I**

Figure 2-D.17 Examples of groups of pictures in display order

These examples illustrate what is possible, and do not constitute a suggestion for structures of groups of pictures.

Group of Pictures Start Code

The group of pictures header starts with the Group of Pictures start code. This code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B8
binary: 0000 0000 0000 0000 0000 0001 1011 1000

It may be preceded by any number of zeros. The encoder may have inserted some zeros to get byte alignment, and may have inserted additional zeros to prevent buffer underflow. An editor may have inserted zeros in order to match the vbv_delay parameter of the first picture in the group.

Time Code

A time code of 25 bits immediately follows the group of pictures start code. This time code conforms to the SMPTE time code [6].

The time code can be broken down into six fields as shown in the following table:

FIELD	BITS	VALUES
Drop frame flag	1	
Hours	5	0 to 23
Minutes	6	0 to 59
Fixed	1	1
Seconds	6	0 to 59
Picture number	6	0 to 60

Table 2-D.4 Time code fields

The time code refers to the first picture in the group in display order, i.e. the first picture with a temporal reference of zero.

Closed GOP

A one bit flag follows the time code. It denotes whether the group of pictures is open or closed. Closed groups can be decoded without using decoded pictures of the previous group for motion compensation, whereas open groups require such pictures to be available.

A typical example of a closed group is shown in Figure 2-D.18a.

(a) closed group

(b) open or closed group

Figure 2-D.18 Example groups of pictures in display order

A less typical example of a closed group is shown in Figure 2-D.18b. In this example, the B pictures which precede the first I picture must use backward motion compensation only, i.e. any motion compensation must be based only on picture number 2 in the group.

If the closed_gop flag is set to 0 then the group is open. The first B pictures in the group may have been encoded using the last picture in the previous group for motion compensation.

Broken Link

A one bit flag follows the closed gop flag. It denotes whether the previous group of pictures can be used to decode the current group. Encoders normally set this flag to 0 indicating that the previous group can be used for decoding. If the sequence has been edited so that the original group of pictures no longer precedes the current group, then this flag must be set to 1 by the editor.

If the group is closed, then the flag is less useful. It is suggested that encoders still set it to zero, and editors set it to 1 so that decoders can detect if the bitstream has been edited at that point.

Extension Data

This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B5
binary: 0000 0000 0000 0000 0000 0001 1011 0101

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes are reserved for future extensions to the MPEG video standard, and should not be generated by encoders. MPEG video decoders should have the capability to discard any extension data found.

User Data

A user data start code may follow the optional extension data. This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B2
binary: 0000 0000 0000 0000 0000 0001 1011 0110

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes can be used by the encoder for any purpose. The only restriction on the data is that they cannot emulate a start code, even if not byte aligned. This means that a string of 23 consecutive zeros must not occur. One way to prevent emulation is to force the most significant bit of alternate bytes to be a 1.

In closed encoder-decoder systems the decoder may be able to use the data. In the more general case decoders should be capable of discarding the user data.

2-D.5.3 Picture

The picture layer contains all the coded information for one picture. The header identifies the temporal reference of the picture, the picture coding type, the delay in the video buffer verifier (VBV) and, if appropriate, the range of the vectors used.

Picture Header and Start Code

A *picture* begins with a picture header. The header starts with a picture start code. This code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 00
binary: 0000 0000 0000 0000 0000 0001 0000 0000

It may be preceded by any number of zeros.

Temporal Reference

The Temporal Reference is a ten-bit number which can be used to define the order in which the pictures must be displayed. It is useful since pictures are not transmitted in display order, but in the order which the decoder needs to decode them. The first picture, in display order, in each group must have Temporal Reference equal to zero. This is incremented by one for each picture in the group.

Some example groups of pictures with their Temporal Reference numbers are given below:

Example (a) in	**I**	B	**P**	B	**P**							
display order	0	1	2	3	4							
Example (a) in	**I**	**P**	B	**P**	B							
decoding order	0	2	1	4	3							
Example (b) in	B	B	**I**	B	B	**P**	B	B	**P**	B	B	**P**
display order	0	1	2	3	4	5	6	7	8	9	10	11
Example (b) in	**I**	B	B	**P**	B	B	**P**	B	B	**P**	B	B
decoding order	2	0	1	5	3	4	8	6	7	11	9	10
Example (c) in	B	**I**	B	B	B	B	**P**	B	**I**	B	B	**I**	**I**
display order	0	1	2	3	4	5	6	7	8	9	10	11	12
Example (c) in	**I**	B	**P**	B	B	B	B	**I**	B	**I**	B	B	**I**
decoding order	1	0	6	2	3	4	5	8	7	11	9	10	12

Figure 2-D.19 Examples of Groups of Pictures and Temporal References

If there are more then 1023 pictures in a group, then the Temporal Reference is reset to zero and then increments anew. This is illustrated below:

display c

Figure 2-D.20 Example Group of Pictures containing 1500 pictures

Picture Coding Type

A three bit number follows the temporal reference. This is an index into the following table defining the type of picture.

CODE	PICTURE TYPE
000	Forbidden
001	I Picture
010	P Picture
011	B Picture
100	D Picture
101	Reserved
110	Reserved
111	Reserved

Table 2-D.5 Picture types

The various types of pictures are described in 2.1. Codes 101 through 111 are reserved for future extensions to the standard. Decoders should be capable of discarding all pictures of this type, and scan for the next picture start code, group start code or sequence start code. Code 000 will never be used to avoid start code emulation.

VBV Delay

For constant bit rate operation, vbv_delay defines the current state of the VBV buffer (VBV is an acronym for Video Buffer Verifier - the model decoder). It specifies how many bits it should contain when bits for all previous pictures have been removed, and the model decoder is about to start decoding the current picture.

Its purpose is to allow the decoder to synchronize its clock with the encoding process, and to allow the decoder to determine when to start decoding a picture after random access in order not to run into future problems of buffer overflow or underflow.

The buffer fullness is not specified in bits but rather in units of time. The vbv_delay is a 16-bit number defining the time needed in units of 1/90000 second to fill the input buffer of the model decoder from an empty state to the current state at the bit rate specified in the sequence header.

For example, suppose the vbv_delay had a decimal value of 30000, then the time delay would be:

D = 30 000 / 90 000 = 1 / 3 second

If the channel bit rate were 1.2 Mbps then the contents of the buffer before the picture is decoded would be:

B = 1 200 000 / 3 = 400 000 bits

If the decoder determined that its actual buffer fullness differed significantly from this value, then it would have to adopt some strategy for regaining synchronization.

The meaning of vbv_delay is undefined for variable bit rate operation.

Full Pel Forward Vector

This is a one bit flag giving the precision of the forward motion vectors. If it is 1 then the precision of the vectors is in integer pels, if it is zero then the precision is half a pel. Thus if the flag is set to 1 the vectors have twice the range than were the flag set to zero.

This flag is present only in the headers of P pictures and B pictures. It is absent in I pictures and D pictures.

Forward F Code

This is a three-bit number and, like the full pel forward vector flag, is present only in the headers of P pictures and B pictures. It provides information used for decoding the coded forward vectors and controls the maximum size of the forward vectors that can be coded. It can take only values of 1 through 7; a value of zero is forbidden.

Two parameters used in decoding the forward motion vectors are derived from forward_f_code: forward_f_size and forward_f.

The forward_f_size is one less than the forward_f_code and so can take values 0 through 6.

The forward_f parameter is given by the following table:

forward_f_code	forward_f
1	1
2	2
3	4
4	8
5	16
6	32
7	64

Table 2-D.6

Full Pel Backward Vector

This is a one bit flag giving the precision of the backward motion vectors. If it is 1 then the precision of the vectors is in integer pels, if it is zero then the precision is half a pel.

This flag is only present in the headers of B pictures. It is absent in I pictures, P pictures and D pictures.

Backward F Code

This is a three-bit number and, like the full pel backward vector flag, is present only in the headers of B pictures. It provides information used for decoding the coded backward vectors. It can take only values of 1 through 7; a value of zero is forbidden.

The backward_f parameter is derived from the backward_f_code and is given by the following table:

backward_f_code	backward_f
1	1
2	2
3	4
4	8
5	16
6	32
7	64

Table 2-D.7

Extra Picture Information

Extra picture information is the next field in the picture header. Any number of information bytes may be present. An information byte is preceded by a flag bit which is set to 1. Information bytes are therefore generally not byte-aligned. The last information byte is followed by a zero bit. The smallest size of this field is therefore one bit, a 0, that has no information bytes. The largest size is unlimited. The following example has 16 bits of extra information denoted by E:

1 E E E E E E E E 1 E E E E E E E E 0

The extra information bytes are reserved for future extensions to the standard. The meaning of these bytes is currently undefined, so encoders must not generate such bytes and decoders must be capable of discarding them.

Extension Data

This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B5
binary: 0000 0000 0000 0000 0000 0001 1011 0101

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes are reserved for future extensions to the MPEG video standard, and should not be generated by encoders. MPEG video decoders must be capable of discarding them.

User Data

This start code is byte-aligned and is 32 bits long. Its value is:

hex: 00 00 01 B2
binary: 0000 0000 0000 0000 0000 0001 1011 0110

It may be preceded by any number of zeros. If it is present then it will be followed by an undetermined number of data bytes terminated by the next start code. These data bytes can be used by the encoder for any purpose. The only restriction on the data is that they cannot emulate a start code, even if not byte aligned. One way to prevent emulation is to force the most significant bit of alternate bytes to be a 1.

In closed encoder-decoder systems the decoder may be able to use the data. In the more general case decoders should be capable of discarding the user data.

2-D.5.4 Slice

Pictures are divided into slices. Each slice consists of an integral number of macroblocks in raster scan order. Slices can be of different sizes within a picture, and the division in one picture need not be the same as the division in any other picture. Slices can begin and end at any macroblock in a picture subject to the following restrictions. The first slice must begin at the top left of the picture, and the end of the last slice must be the bottom right macroblock of the picture. There can be no gaps between slices, nor can slices overlap. The minimum number of slices in a picture is one, the maximum number is equal to the number of macroblocks.

Each slice starts with a slice start code. This is followed by a code that defines its position, and a code that sets the quantization step size. Since the quantization step size also can be set for any macroblock (see Clause 2-D.5.5), there is no purely coding reason for using slices. Instead they are intended to help error recovery in noisy environments. If the coded data is corrupted, and the decoder detects this, then the decoder can search for the next slice start code and resume decoding the picture from that point. If the data is corrupted, and the decoder fails to detect this, then the decoder may encounter an unexpected slice start code. It should then start decoding from the position indicated in the slice header.

If the data is to be used in an error free environment, then one slice per picture may be appropriate. If the environment is noisy, then one slice per row of macroblocks may be more desirable, as shown in Figure 2-D.21.

1 begin	end 1
2 begin	end 2
3 begin	end 3
4 begin	end 4
5 begin	end 5
6 begin	end 6
7 begin	end 7
8 begin	end 8
9 begin	end 9
10 begin	end 10
11 begin	end 11
12 begin	end 12
13 begin	end 13

Figure 2-D.21 Possible Arrangement of Slices in a 256x192 Picture

In this Figure and in the next, each strip is one macroblock high, i.e. 16 pels high.

Since each slice header requires 40 bits, there is some penalty for including more than the minimum number of slices. For example, a sequence with a vertical resolution of 240 lines coded at 30 pictures per second requires approximately 40 x 30 = 1200 bps for the slice headers using one slice per picture, and 40 x 15 x 30 = 18000 bps with one slice per row, an additional overhead of 16800 bps. The calculation is approximate since the inclusion of a slice imposes additional requirements that the macroblock immediately before the slice header be coded, as well as the first macroblock in the slice.

The coding structure permits great flexibility in dividing a picture up into slices. One possible arrangement is shown in Figure 2-D.22:

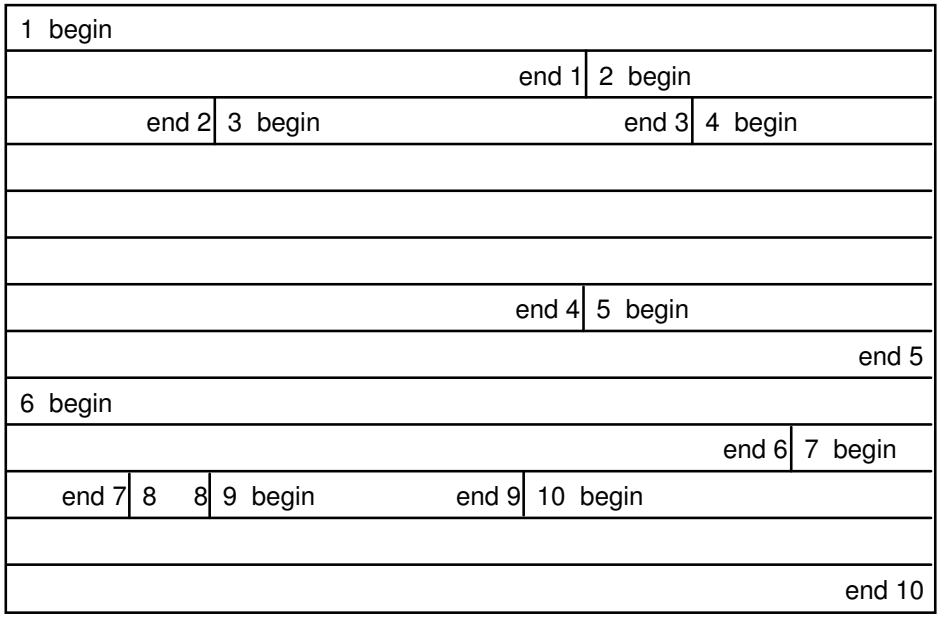


Figure 2-D.22 Possible Arrangement of Slices in a 256x192 Picture

This division into slices is given for illustrative purposes only. It is not intended as a suggestion on how to divide a picture into slices.

Slice Header and Start Code

Slices start with a slice header. Each slice header starts with a slice start code. This code is byte-aligned and is 32 bits long. The last eight bits can take on a range of values which define the vertical position of the slice in the picture. The permitted slice start codes are:

hex:	from	00 00 01 01
	to	00 00 01 AF
binary:	from	0000 0000 0000 0000 0000 0001 0000 0001
	to	0000 0000 0000 0000 0000 0001 1010 1111

Each slice start code may be preceded by any number of zeros.

The last 8 bits of the slice start code gives the slice vertical position, i.e the vertical position of the first macroblock in the slice in units of macroblocks starting with position 1 at the top of the picture. A useful variable is macroblock row. This is similar to slice vertical position except that row 0 is at the top of the picture. Thus:

slice vertical position = macroblock row + 1

For example, a slice start code of 00000101 hex means that the first macroblock in the slice is at vertical position 1 or macroblock row 0, i.e. at the top of the picture. A slice start code of 00000120 hex means that the first macroblock is at vertical position 32 or macroblock row 31, i.e. at the 496th row of pels. It is possible for two or more slices to have the same vertical position.

The maximum vertical position is 175 units. A slice with this position would require a vertical size of 175 x 16 = 2800 pels.

The horizontal position of the first macroblock in the slice can be calculated from its macroblock address increment. Thus its position in the picture can be determined without referring to any previous slice or macroblock. Thus a decoder may decode any slice in a picture without having decoded any other slice in the same picture. This feature allows decoders to recover from bit errors by searching for the next slice start code and then resuming decoding.

Quantizer Scale

The quantizer scale is a five-bit integer which is used by the decoder to calculate the DCT coefficients from the transmitted quantized coefficients. A value of 0 is forbidden, so the quantizer scale can have any value between 1 and 31 inclusive.

Note that the quantizer scale may be set at any macroblock.

Extra Slice Information

Extra slice information forms the last field in the slice header. Any number of information bytes may be present. An information byte is preceded by a flag bit which is set to 1. Information bytes are therefore generally not byte-aligned. The last information byte is followed by a zero bit. The smallest size of this field is therefore one bit, a 0, that has no information bytes. The largest size is unlimited. The following example has 24 bits of extra information denoted by E:

1 E E E E E E E E 1 E E E E E E E E 1 E E E E E E E E 0

The extra information bytes are reserved for future extensions to the standard. The meaning of these bytes is currently undefined, so encoders must not generate such bytes and decoders must discard them.

The slice header is followed by code defining the macroblocks in the slice.

2-D.5.5 Macroblock

Slices are divided into macroblocks of 16 x 16 pels. Macroblocks are coded with a header that contains information on the macroblock address, macroblock type, and the optional quantizer scale. The header is followed by data defining each of the six blocks in the macroblock It is convenient to discuss the macroblock header fields in the order in which they are coded.

Macroblock Stuffing

The first field in the macroblock header is "macroblock stuffing". This is an optional field, and may be inserted or omitted at the discretion of the encoder. If present it consists of any number of 11-bit strings with the pattern "0000 0001 111". This stuffing code is used by the encoder to prevent underflow, and is discarded by the decoder. If the encoder determines that underflow is about to occur, then it can insert as many stuffing codes into the first field of the macroblock header it likes.

Note that an encoder has other strategies to prevent buffer underflow. It can insert stuffing bits immediately before a start code. It can reduce the quantizer scale to increase the number of coded coefficients. It can even start a new slice.

Macroblock Address Increment and Macroblock Escape

Macroblocks have an address which is the number of the macroblock in raster scan order. The top left macroblock in a picture has address 0, the next one to the right has address 1 and so on. If there are M macroblocks in a picture, then the bottom right macroblock has an address M-1.

The address of a macroblock is indicated by transmitting the difference between the addresses of the current macroblock and that of the previously coded macroblock. This difference is called the macroblock address increment. In I pictures, all macroblocks are coded and so the macroblock address increment is nearly always one. There is one exception. At the beginning of each slice the macroblock address is set to that of the right hand macroblock of the previous row. At the beginning of the picture it is set to -1. If a slice does not start at the left edge of the picture, then the macroblock address increment for the first macroblock in the slice will be larger than one. For example, the picture of Figure 2-D.22 has 16

macroblocks per row. At the start of slice 2 the macroblock address is set to 15 which is the address of the macroblock at the right hand edge of the top row of macroblocks. If the first slice contained 26 macroblocks, 10 of them would be in the second row, so the address of the first macroblock in slice 2 would be 26 and the macroblock address increment would be 11.

Macroblock address increments are coded using the VLC codes in the table in 2-Annex B.1.

It can be seen that there is no code to indicate a macroblock address increment of zero. This is why the macroblock address is set to -1 rather than zero at the top of a picture. The first macroblock will have an increment of one making its address equal to zero.

The macroblock address increments allow the position of the macroblock within the picture to be determined. For example, assume that a slice header has the start code equal to 00 00 01 0A hex, that the picture width is 320 pels, and that a macroblock address increment code 0000111 is in the macroblock header of the first macroblock in the slice. The picture width tells us that there are 20 macroblocks per row in this picture. The slice start code tells us that the slice vertical position is 10, and so the macroblock row is 9. The slice header sets the previous macroblock address to the last macroblock on row 8, which has address 143. The macroblock address increment VLC tells us that the macroblock address increment is 8, and so the macroblock address of the first macroblock in the slice is 143 + 8 = 151.

The macroblock row may be calculated from the address:

macroblock row = macroblock address / macroblock width

= 151 / 16

= 9

The division symbol signifies integer truncation, not rounding.

The macroblock column may also be calculated from the address:

macroblock column = macroblock address % macroblock width
= 151 % 16
= 7

Columns are numbered from the left of the picture starting at 0.

There are two special codewords: escape and stuffing.

The escape code means "add 33 to the following macroblock address increment". This allows increments greater than 33 to be coded. For example, an increment of 40 would be coded as escape plus an increment of 7:

0000 0001 0000 0010

An increment of 70 would be coded as two escape codes followed by the code for an increment of 4:

0000 0001 0000 0000 0010 0000 11

The stuffing code is included since the decoder must be able to distinguish it from increment codes. It is used by the encoder to prevent underflow, and is discarded by the decoder.

Macroblock Types

Each of the picture types I, P, and B, have their own macroblock types. See, respectively, 2-D.6.2, 2-D.6.4, and 2-D.6.5 for the codes and their descriptions.

Motion Horizontal/Vertical Forward/Backward Codes

The interpretation of these codes is explained in Clause 2-D.6.2.3.

Motion Horizontal/Vertical Forward/Backward R

The interpretation of these codes is explained in Clause 2-D.6.2.3.

Coded Block Pattern

This code describes which blocks within the macroblock are coded and transmitted. The interpretation of this code is explained in Clause 2-D.6.4.2.

End of Macroblock

This code is used only in D-pictures and is described in Clause 2-D.6.6.

2-D.5.6 Block

A block is an array of 8 by 8 component pel values, treated as a unit and input to the Discrete Cosine Transform (DCT). Blocks of 8 by 8 pels are transformed into arrays of 8 by 8 DCT coefficients using the two dimensional discrete cosine transform.

2-D.6 CODING MPEG VIDEO

2-D.6.1 Rate Control

The encoder must control the bit rate so that the model decoder input buffer neither overflows nor underflows. Since the model decoder removes all the bits associated with a picture from the input buffer instantaneously, it is necessary to control only the total number of bits per picture. The encoder should allocate the total numbers of bits among the various types of pictures so that the perceived quality is suitably balanced. The distribution will vary with the scene content and with the particular distribution of the three picture types (I, P and B pictures).

Within a picture the encoder should allocate the total number of bits available among the macroblocks to maximize the visual quality of the picture.

One method by which an encoder controls the bit rate is to vary the quantizer scale. This is set in each slice header, and may be set at the beginning of any macroblock, giving the encoder excellent control over the bit rate within a picture.

Rate Control within a Sequence

For a typical coding scheme represented by the following group of pictures in display order:

B B I B B P B B P B B P B B P

it has been found that good results can be obtained by matching the visual quality of the I and P pictures, and by reducing the code size of the B pictures to save bits giving a generally lower quality for the B pictures.

The best allocation of bits among the picture types depends on the scene content. Work of the MPEG video committee suggests that allotting P pictures about 2-5 times as many bits as B pictures, and allotting I pictures up to 3 times as many bits

as P pictures gives good results for typical natural scenes. If there is little motion or change in the video, then a greater proportion of the bits should be allotted to the I pictures. If there is a lot of motion or change, then the proportion allotted to I pictures should be reduced and most of the savings given to the P pictures.

A reasonable encoder algorithm is to start with the foregoing estimates, then reallocate bits dynamically depending on the nature of the video.

Rate Control within a Picture

If the buffer is heading toward overflow, the quantizer scale should be increased. If this action is not sufficient to prevent an impending overflow then, as a last resort, the encoder could discard high frequency DCT coefficients and only transmit low frequency ones. Although this would probably produce visible artifacts in the decoded video, it would in no way compromise the validity of the coded bitstream.

If the buffer is heading toward underflow, the quantizer scale should be reduced. If this is not sufficient, the encoder can insert macroblock stuffing into the bitstream, or add leading zeros to start codes.

Under normal circumstances, the encoder calculates and monitors the state of the model decoder buffer and changes the quantizer scale to avert both overflow and underflow problems.

One simple algorithm that helps accomplish this is to monitor the buffer fullness. Assume that the bits have been allocated among the various picture types, and that an average quantizer scale for each picture type has been established. The actual buffer fullness at any macroblock in a picture can be calculated and compared with the nominal fullness, i.e. the value that would be obtained if the bits were uniformly distributed among all the macroblocks in the picture. If the buffer fullness is larger than the nominal value, then the quantizer scale should be set higher than the average, whereas if the buffer fullness is smaller than the nominal, the quantizer scale should be set lower than the average.

If the quantizer scale is kept constant over a picture, then, for a given number of coding bits, the total mean square error of the coded picture tends to be close to the minimum. However, the visual appearance of most pictures can be improved by varying the quantizer scale over the picture, making it smaller in smooth areas of the picture and larger in busy areas. This technique reduces the visibility of blockiness in smooth areas at the expense of increased quantization noise in the busy areas where, however, it is masked by the image detail.

Thus a good picture rate control algorithm adjusts the quantizer scale depending on both the calculated buffer fullness and on the local image content.

Buffer Fullness

To give the best visual quality, the encoder should almost fill the input buffer before instructing the decoder to start decoding.

2-D.6.2 Motion Estimation and Compensation

2-D.6.2.1 Motion Compensation

P pictures use motion compensation to exploit temporal redundancy in the video. Decoders construct a predicted block of pels from pels in a previously transmitted picture. Motion within the pictures (e.g. a pan) usually implies that the pels in the previous picture will be in a different position from the pels in the current block, and the displacement is given by motion vectors encoded in the bitstream. The predicted block is usually a good estimate of the current block, and it is usually more efficient to transmit the motion vector plus the difference between the predicted block and the current block, than to transmit a description of the current block by itself.

Consider the following typical group of pictures.

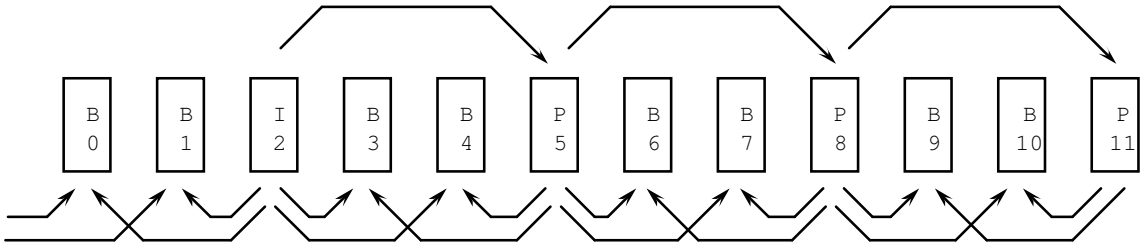


Figure 2-D.23 Group of Pictures in Display Order

The I picture, picture 2, is decoded without requiring any motion vectors. The first P picture, number 5, is decoded using motion vectors from picture 2. This motion compensation is called forward motion compensation since it is forward in time. Motion vectors define the motion of a macroblock, i.e. the motion of a 16x16 block of luminance pels and the associated chrominance components. Most macroblocks in a P picture use motion compensation. Non-zero motion vectors are transmitted differentially with reference to the last transmitted motion vector.

The transmitted vectors usually have a precision of half a pel. The maximum range of the vector is set by the forward_f parameter in the picture header. Sometimes, if the motion is unusually large, the range may be doubled and the accuracy reduced to integer pels, by the full_pel_forward_vector bit in the picture header.

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pels in the referenced picture which are spatially to the right or below the pels being predicted.

Not all macroblocks in a P picture necessarily use motion compensation. Some macroblocks, as defined by the transmitted macroblock type (see Table 2-B.2b in 2-Annex B of this standard), may be intra-coded, and these are reconstructed without motion compensation. Full details defining the method of decoding the vectors and constructing the motion-compensated macroblock are given in Clause 2.4.4.2 of Part 2 of this standard.

P picture 8 in Figure 2-D.23 uses forward motion compensation from picture 5. P pictures always use forward motion compensation from the last transmitted I or P picture.

B pictures may use motion compensation from the previous I or P picture, from the next (in display order) I or P picture, or both; i.e., from the last two transmitted I or P pictures.

Prediction is called forward if reference is made to a picture in the past and called backward if reference is made to a picture in the future. For example, B picture 3 in Figure 2-D.23 uses forward motion compensation from I picture 2, and backward motion compensation from P picture 5. B pictures may use both forward and backward motion compensation and average the result. This operation is called interpolative motion compensation.

All three types of motion compensation are useful, and are typically used in coding B pictures. Interpolative motion compensation has the advantage of averaging any noise present. Forward or backward motion compensation may be more useful near the edges of pictures, or where a foreground object is passing in front of a fixed or slow moving background.

Note that this technique of coding with P and B pictures increases the coding efficiency. B pictures can have greater errors of reconstruction than I or P pictures to conserve coding bits, but since they are not used as the basis of motion compensation for future pictures, these errors do not accumulate.

2-D.6.2.2 Motion Estimation

Motion compensation in a decoder is straightforward, but motion estimation, determining the best motion vectors, which must be done by the encoder, presents a formidable computational challenge.

Various methods are available to the encoder. The more computationally intensive methods tend to give better results, so there is tradeoff to be made in the encoder: computational power, and hence cost, versus coded video quality.

Using a search strategy the encoder attempts to match the pels in a macroblock with those in a previous or future picture. The vector corresponding to the best match is reported after the search is completed.

Block Matching Criteria

In seeking a match, the encoder must decide whether to use the decoded past and future pictures as the reference, or use the original past and future pictures. Note that the decoder does not perform motion estimation. It performs motion compensated prediction and interpolation using vectors calculated in the encoder and stored in the bitstream. In motion compensated prediction and interpolation, both the encoder and decoder must use the decoded pictures as the references. For motion estimation, use of the decoded pictures by the encoder gives the smallest error in the error picture, whereas use of the original pictures gives the most accurate motion vectors. The choice depends on whether the artifacts of increased noise, or greater spurious motion are judged to be the more objectionable. There is usually little or no difference in quality between the two methods.

Several matching criteria are available. The mean square error of the difference between the motion-compensated block and the current block is an obvious choice. Another possible criterion is the mean absolute difference between the motion-compensated block and the current block.

For half pel shifts, the pel values could be interpolated by several methods. Since the decoder uses a simple linear interpolation, there is little reason to use a more complex method in the encoder. The linear interpolation method given in the standard is equivalent to the following. Consider four pels having values A, B, D and E as shown in Figure 2-D.24:

A h B
v c
D E

Figure 2-D.24 Interpolation of half pel shifts

The value of the horizontally interpolated pel is:

$$h = (A + B) // 2$$

where the double division symbol means division with rounding to the nearest integer. Half integer values are to be rounded to the next higher value. Thus if A = 4 and B = 9 then h = 6.5 which is rounded up to 7.

The value of the vertically interpolated pel is:

$$v = (A + D) // 2$$

The value of the central interpolated pel is:

$$c = (A + B + D + E) // 4$$

Search Range

Once a block matching criterion has been selected, some kind of search strategy must be adopted. Part of the search strategy should recognize the limitations of the VLC tables used to code the vectors. The maximum range of the vector depends upon foward_f_code or backward_f_code. The motion vector ranges are given in the following table:

forward_f_code or backward_f_code	Motion vector range full_pel=0	full_pel=1
1	-8 to 7.5	-16 to 15
2	-16 to 15.5	-32 to 31
3	-32 to 31.5	-64 to 63
4	-64 to 63.5	-128 to 127
5	–128 to 127.5	-256 to 255
6	-256 to 255.5	-512 to 511
7	–512 to 511.5	–1024 to 1023

Table 2-D.7 Range of motion vectors

The range depends on the value of full_pel_forward_vector or full_pel_backward_vector in the picture header. Thus if all the motion vectors were found to be 15 pels or less, the encoder would usually select half pel accuracy and a forward_f_code or backward_f_code value of 2.

The search must be constrained to take place within the boundaries of the decoded reference picture. Motion vectors which refer to pels outside the picture are not allowed. Any bitstream which refers to such pels does not conform to this standard.

2-D Search Strategy

There are many possible methods of searching another picture for the best match to a current block, and a few simple ones will be described.

The simplest search is a full search. Within the chosen search range all possible displacements are evaluated using the block matching criterion.

The full search is computationally expensive, and practical encoders may not be able to afford the time required for a full search.

A simple modification of the full search is to search using only integer pel displacements. Once the best integer match has been found, the eight neighboring half-integer pel displacements are evaluated, and the best one selected as illustrated below:

Figure 2-D.25 Integer pel and half pel displacements

Assume that the position x+2,y+2 gives the best integer displacement matching using the selected block matching criterion, then the encoder would evaluate the eight positions with half pel displacements marked by + signs in Figure 2-D.25. If one of them were a better match then it would become the motion vector, otherwise the motion vector would remain that of the integer displacement x+2,y+2.

If during the integer pel search, two or more positions have the same block matching value, the encoder can adopt a consistent tie-breaking rule.

The modified full search algorithm is approximately an order of magnitude simpler than the full search. Using only integer displacements for the first stage of the search reduces the number of evaluations by a factor of four. In addition, the evaluations are simpler since the pel differences can be calculated directly and do not have to be interpolated.

For some applications even the modified full search may be too time consuming, and a faster search method may be required. One such method is the logarithmic search.

Logarithmic Search

In this search method, grids of 9 displacements are examined, and the search continued based on a smaller grid centered on the position of the best match. If the grids are reduced in size by a factor of 3 at each step then the search is maximally efficient in the sense that any integer shift has a unique selection path to it. Unfortunately this efficient method will only find the best match only for a rather limited set of image types. A more robust method is to reduce the size of the grids by a smaller factor at each step, e.g. by a factor of 2. The scaling factors can also be adjusted to match the search ranges of Table 2-D.7.

The method will be illustrated with an example. Consider the set of integer shifts in Figure 2-D.26:

```
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  1  *  *  *  1  *  *  *  1  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  2  *  2  *  2  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  2  *  1  *  2  *  1  *  *  *  1  *  *  *
*  *  *  *  3  3  3  *  *  *  *  *  *  *  *
*  2  *  2  3  2  3  *  *  *  *  *  *  *  *
*  *  *  *  3  3  3  *  *  *  *  *  *  *  *
*  *  *  1  *  *  *  1  *  *  *  1  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
```

Figure 2-D.26 Logarithmic search method for integer pel shifts

The first grid has a spacing of 4 pels. The first step examines pels at shifts of 0, 4, or -4 pels in each direction, marked 1 in Figure 2-D.26. The best position is used as the center point of the second grid. Assume it is the pel marked 1 directly to the left of the center pel. The second grid has a spacing of 2 pels. The second step examines pels at shifts of 0, 2, or -2 pels in each direction from the center of the new grid, marked 2 in the figure. The best position is used as the center point of the third grid, assume it is the lower right pel of the second grid. The third grid has a spacing of 1 pel. The third step examines pels at shifts of 0, 1, or -1 pels in each direction from the center of the grid. The best position is used as the center point of the fourth grid. The fourth grid has a spacing of 1/2 pel. The fourth step examines pels at shifts of 0, 1/2, or -1/2 pels in each direction from the center of the grid using the same method as in the modified full search. The best position determines the motion vector.

Some possible grid spacings for various search ranges are given in the following table:

forward_f_c	RANGE	STEPS	GRID SPACINGS
1	±7.5	4	4 2 1 1/2
2	±15.5	5	8 4 2 1 1/2
4	±31.5	6	16 8 4 2 1 1/2

Table 2-D.8 Grid spacings for logarithmic searches

For P pictures only forward searches are done, but B pictures require both forward and backward searches. Not all the vectors calculated during the search are necessarily used. In B pictures either forward or backward motion compensation might be used instead of interpolated motion compensation, and in both P and B pictures the encoder might decide that a block is better coded as intra, in which case no vectors are transmitted.

Telescopic Search

Even with the faster methods of the modified full search, or the logarithmic search, the search might be quite expensive. For example, if the encoder decides to use a maximum search range of 7 pels per picture interval, and if there are 4 B pictures preceding a P picture, then the full search range for the P picture would be 35 pels. This large search range may exceed the capabilities of the encoder.

One way of reducing the search range is to use a telescopic search technique. This is best explained by illustrating with an example. Consider the group of pictures in Figure 2-D.27:

1 2 3 4 5 6 7 8 9 10 11 12

Figure 2-D.27 Example group of pictures in display order

The encoder might proceed using its selected block matching criterion and 2-D search strategy. For each P picture and the preceding B pictures, it first calculates all the forward vectors, then calculates all the backward vectors. The first set of pictures consists of pictures 0 through 4.

To calculate the complete set of forward vectors, the encoder first calculates all the forward vectors from picture 0 to picture 1 using a 2-D search strategy centered on zero displacement. It next calculates all the forward vectors from picture 0 to picture 2 using a 2-D search strategy centered on the displacements calculated for the corresponding block of picture 1. It next calculates all the forward vectors from picture 0 to picture 3 using a 2-D search strategy centered on the displacements calculated for the corresponding block of picture 2. Finally, it calculates all the forward vectors from picture 0 to picture 4 using a 2-D search strategy centered on the displacements calculated for the corresponding block of picture 3.

To calculate the complete set of backward vectors, the encoder first calculates all the backward vectors from picture 4 to picture 3 using a 2-D search strategy centered on zero displacement. It next calculates all the backward vectors from picture 4 to picture 2 using a 2-D search strategy centered on the displacements calculated for the corresponding block of picture 3. Finally, it calculates all the backward vectors from picture 4 to picture 1 using a 2-D search strategy centered on the displacements calculated for the corresponding block of picture 2.

Further methods of motion estimation are given by Netravali and Haskell [1].

2-D.6.2.3 Coding of Motion Vectors

The motion vector of a macroblock tends to be well correlated with the vector of the previous macroblock. For example, in a pan all vectors would be roughly the same. Motion vectors are coded using a DPCM technique to make use of this correlation.

In P pictures the motion vector used for DPCM, the prediction vector, is set to zero at the start of each slice and at each intra-coded macroblock. Note that macroblocks which are coded as predictive but which have no motion vector, also set the prediction vector to zero.

In B pictures there are two prediction vectors, forward and backward. Each vector is coded relative to the predicted vector of the same type. Both prediction vectors are set to zero at the start of each slice and at each intra-coded macroblock. Note that predictive macroblocks which have only a forward vector do not affect the value of the predicted backward vector. Similarly, predictive macroblocks which have only a backward vector do not affect the value of the predicted forward vector.

The range of the vectors is set by two parameters. The full pel forward vector and full pel backward vector flags in the picture header determine whether the vectors are defined in half-pel or integer-pel units.

A second parameter, forward-f_code or backward_f_code, is related to the number of bits appended to the VLC codes in the following table.

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 11	-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

Table 2-D.9 Differential motion code.

Advantage is taken of the fact that the range of displacement vector values is constrained. Each VLC represents a pair of difference values. Only one of the pair will yield a motion vector falling within the permitted range.

The range of the vector is limited to the values shown in Table 2-D.7. The values obtained by decoding the differential values must be kept within this range by adding or subtracting a modulus which depends on the f value as shown below:

forward_f_code or backward_f_code	MODULUS
1	32
2	64
3	128
4	256
5	512
6	1024
7	2048

Table 2-D.10 Modulus for motion vectors

The use of the modulus, which refers only to the numbers in tables 2-D.8 through 2-D.10, will be illustrated by an example. Assume that a slice has the following vectors, expressed in the units set by the full pel flag:

3 10 30 30 -14 -16 27 24

The range is such that an f value of 2 can be used. The initial prediction is zero, so the differential values are:

3 7 20 0 -44 -2 43 -3

The differential values are reduced to the range -32 to +31 by adding or subtracting the modulus 64 corresponding to the forward_f_code of 2:

3 7 20 0 20 -2 -21 -3

These values are divided by forward_f and a code word is created by concatenating the VLC (from Table 2-D.9) of the signed quotient with the fixed length code of the unsigned remainder (forward_f_size bits).

2-D.6.3 Coding I-Pictures

In coding I pictures, the encoder has two main decisions to make that are not mandated by this standard. These are: how to divide the picture up into slices, and how to set the quantizer scale.

2-D.6.3.1 Slices in I-Pictures

Division of the picture into slices is described in Clause 2-D.5.4.

2-D.6.3.2 Macroblocks in I Pictures

Macroblock Types in I Pictures

There are two types of macroblock in I pictures. Both use intra coding. One uses the current quantizer scale, whereas the other defines a new value for the quantizer scale. They are identified in the coded bitstream by the VLC codes given in the Table 2-B2a.

TYPE	QUANT	VLC
intra-d		1
intra-q	1	01

Table 2-D.11 Macroblock type VLC for I pictures (table 2-B.2a.)

The types are referred to names in this annex. Intra-d is the default type where the quantizer scale is not changed. Intra-q sets the quantizer scale.

In order to allow for possible future extension to MPEG video, the VLC for intra-q is 01 rather than 0. Additional types could be added to this table without interfering with the existing entries. The VLC table is thus open for future additions, and not closed. A policy of making the coding tables open in this way was adopted by the MPEG video committee in developing the standard. The advantage of future extension was judged to be worth the slight coding inefficiency.

Quantizer Scale

If the macroblock type is intra-q, then the macroblock header contains a five-bit integer which defines the quantizer scale. This is used by the decoder to calculate the DCT coefficients from the transmitted quantized coefficients. A value of 0 is forbidden, so the quantizer scale can have any value between 1 and 31 inclusive.

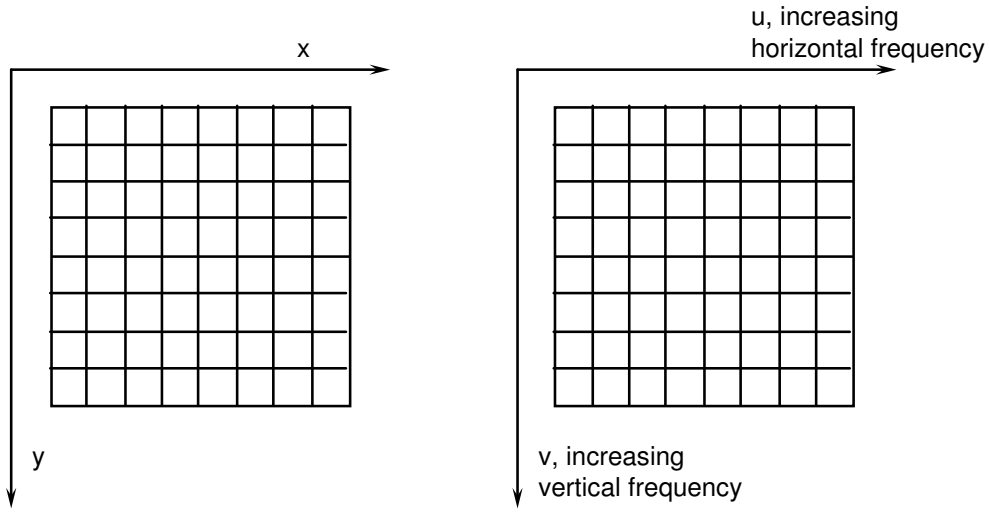
Note that the quantizer scale is set in a slice header.

If the block type is intra-d, then no quantizer scale is transmitted and the decoder uses the previously set value. For a discussion on strategies encoders might use to set the quantizer scale, see Clause 2-D.6.1.

Note that the cost of transmitting a new quantizer scale is six bits: one for the extra length of the macroblock type code, and five to define the value. Although this is normally a small fraction of the bits allocated to coding each macroblock, the encoder should exercise some restraint and avoid making a large number of very small changes.

2-D.6.3.3 DCT Transform

The DCT is illustrated in the following figure:



(a) Pels (b) DCT Coefficients

Figure 2-D.28 Transformation of Pixels to Coefficients

The pels are shown in raster scan order, whereas the coefficients are arranged in frequency order. The top left coefficient is the DC term and is proportional to the average value of the component pel values. The other coefficients are called AC coefficients. AC coefficients to the right of the DC coefficient represent increasing horizontal frequencies, whereas AC coefficients below the DC coefficient represent increasing vertical frequencies. The remaining AC coefficients contain both horizontal and vertical frequency components. Note that an image containing only vertical lines contains only horizontal frequencies.

The coefficient array contains all the information of the pel array and the pel array can be exactly reconstructed from the coefficient array, except for information lost by the use of finite arithmetic precision.

The two-dimensional DCT is defined as:

$$F(u,v) = 1/4\, C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos(\pi(2x+1)u/16)\cos(\pi(2y+1)v/16)$$

with: u, v, x, y = 0, 1, 2, ... 7

where x, y = spatial coordinates in the pel domain
u, v = coordinates in the transform domain

$$C(u),C(v) = 1/\sqrt{2} \text{ for } u, v = 0$$
$$= 1 \quad \text{otherwise}$$

This transform is separable, i.e. a one-dimensional DCT transform may be applied first in the horizontal direction and then in the vertical direction. The formula for the one dimensional transform is:

$$F(u) = 1/2\, C(u) \sum_{x=0}^{7} f(x)\cos(\pi(2x+1)u/16)$$

$$C(u)=1/\sqrt{2} \quad \text{for } u = 0$$
$$= 1 \quad \text{otherwise}$$

Fast DCT transforms exists, analogous to fast Fourier transforms. See ref [3].

The input pel values have a range from 0 to 255, giving a dynamic range for the DC coefficient from 0 to 2040. The maximum dynamic range for any AC coefficient is about -1000 to 1000. Note that for P and B pictures the component pels represent difference values and range from -255 to 255. This gives a maximum dynamic range for any coefficient of about -2000 to 2000. The encoder may thus represent the coefficients using 12 bits whose values range from -2048 to 2047.

2-D.6.3.4 Quantization

Each array of 8 by 8 coefficients produced by the DCT transform operation is quantized to produce an 8 by 8 array of quantized coefficients. Normally the number of non-zero quantized coefficients is quite small, and this is one of the main reasons why the compression scheme works as well as it does.

The coefficients are quantized with a uniform quantizer. The characteristic of this quantizer, only for I-blocks, is shown below:

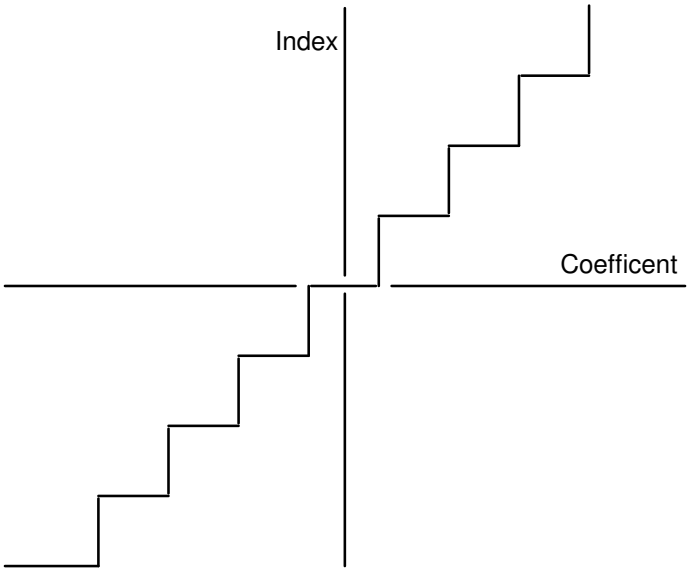


Figure 2-D.29. Uniform Quantizer Characteristics

The value of the coefficient is divided by the quantizer step size and rounded to the nearest whole number to produce the quantized coefficient. Half integer values may be rounded up or down without directly affecting image quality. However, rounding towards zero tends to give the smallest code size and so is preferred. For example, with a step size of 16 all coefficients with values between 25 and 40 inclusive would give an quantized coefficient of 2.

The quantizer step size is derived from the quantization matrix and the quantizer scale. It can thus be different for different coefficients, and may change between macroblocks. The only exception is the DC coefficient which is treated differently.

The eye is quite sensitive to large area luminance errors, and so the accuracy of coding the DC value is fixed. The quantizer step size for the DC coefficients of the luminance and chrominance components is fixed at eight. The DC quantized coefficient is obtained by dividing the DC coefficient by eight and rounding to the nearest whole number. This effectively quantizes the average DC value to one part in 256 for the reconstructed pels.

For example, a DC coefficient of 21 is quantized to a value of 3, independent of the value of the quantizer scale.

AC coefficients are quantized using the intra quantization matrix. The quantized coefficient i[u,v] produced by quantizing the coefficient c[u,v] for I-blocks. One equation is given by the formula:

$$i[u,v] = 8 * c[u,v] / (q * m[u,v])$$

where m[u,v] is the corresponding element of the intra quantization matrix, and q is the quantizer scale. The quantized coefficient is clipped to the range -255 to +255.

The intra quantization matrix might be the default matrix, or it might have been downloaded in the sequence header.

2-D.6.3.4 Coding of Quantized Coefficients

The top left coefficient in Figure 2-D.28b is called the DC coefficient, the remainder are called AC coefficients. The DC coefficient is correlated with the DC coefficient of the preceding block, and advantage is taken of this in coding. The AC coefficients are not well correlated, and are coded independently.

After the DC coefficient of a block has been quantized it is coded losslessly by a DPCM technique. Coding of the luminance blocks within a macroblock follows the raster scan order of Figure 2-D.5, 0 to 3. Thus the DC value of block 3 becomes the DC predictor for block 0 of the following macroblock. Chrominance blocks are coded using the corresponding value of the previous block. At the beginning of each slice, all three DC predictors for Y, Cb and Cr, are set to 1024 (128*8).

The differential DC values thus generated are categorized according to their absolute value as shown in the table below.

DIFFERENTIAL DC (absolute value)	SIZE	VLC CODE (luminance)	VLC CODE (chrominance)
0	0	100	00
1	1	00	01
2 to 3	2	01	10
4 to 7	3	101	110
8 to 15	4	110	1110
16 to 31	5	1110	1111 0
32 to 63	6	1111 0	1111 10
64 to 127	7	1111 10	1111 110
128 to 255	8	1111 110	1111 1110

Table 2-D.12. Differential DC size and VLC

The size is transmitted using a VLC. This VLC is different for luminance and chrominance since the statistics are different.

The size defines the number of additional bits required to define the level uniquely. Thus a size of 6 is followed by 6 additional bits. These bits define the level in order, from low to high. Thus the first of these extra bits gives the sign: 0 for negative and 1 for positive. A size of zero requires no additional bits.

The additional codes are given in the following table:

DIFFERENTIAL DC	SIZE	ADDITIONAL CODE
-255 to -128	8	00000000 to 01111111
-127 to -64	7	0000000 to 0111111
-63 to -32	6	000000 to 011111
-31 to -16	5	00000 to 01111
-15 to -8	4	0000 to 0111
-7 to -4	3	000 to 011
3 to -2	2	00 to 01
-1	1	0
0	0	
1	1	1
2 to 3	2	10 to 11
4 to 7	3	100 to 111
8 to 15	4	1000 to 1111
16 to 31	5	10000 to 11111
32 to 63	6	100000 to 111111
64 to 127	7	1000000 to 1111111
128 to 255	8	10000000 to 11111111

Table 2-D.13. Differential DC additional code

For example, a luminance DC change of 10 would be coded as 1101010. Table 2-D.12 shows that the first three bits 110 indicate that the size is 4. This means that four additional bits are required to define the exact value. The next bit is a 1, and Table 2-D.13 shows that the differential DC value must be somewhere between 8 and 15 inclusive. The last three bits, 010, show that the exact value is 10.

The decoder reconstructs DC quantized coefficients by following the inverse procedure.

The AC quantized coefficients are coded using a run length and level technique. The quantized coefficients are first scanned in the zigzag order shown in Figure 2-D.30:

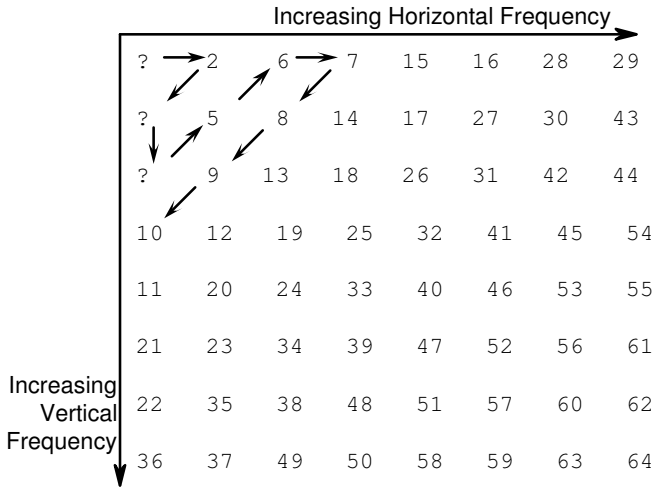


Figure 2-D.30. Quantized coefficient Block in Zigzag Scan Order

The scanning order starts at 1, passes through 2, 3 etc in order, eventually reaching 64 in the bottom right corner. The length of a run is the number of zero quantized coefficients skipped over. For example, the quantized coefficients in Figure 2-D.31 produce the list of run lengths and levels in Table 2-D.14:

```
1   0   0   0   0   0   0   0   0
2  -3   0   0   0   0   0   0   0
4  -5   0   0   0   0   0   0   0
1   0   0 130   0   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0
```

Figure 2-D.31. Example Quantized coefficients

RUN-LENGTH	LEVEL
1	2
0	4
0	-3
3	-5
0	1
14	130
end	

Table 2-D.14. Example Run Lengths and Levels

The scan starts at position 1 since the top left quantized coefficient is coded separately as the DC quantized coefficient.

Using a zig zag scan rather than a raster scan is more efficient as it gives fewer runs and can be coded with shorter VLC codes.

The list of run lengths and levels is coded using Table 2-D.15 Not all possible combinations of run length and level are in these tables, only the more common ones. For combinations not in the tables, an escape sequence is used. In Table 2-D.15, the last bit 's' denotes the sign of the level; 0 means a positive level and 1 means a negative level. The escape code is used followed by the run length derived from Table 2-D.16 and then the level from Table 2-D.17.

RUN	LEVEL	VLC CODE
EOB		10
0	1	1s IF FIRST COEFF
0	1	11s NOT FIRST COEFF
0	2	0100 s
0	3	0010 1s
0	4	0000 110s
0	5	0010 0110 s
0	6	0010 0001 s
0	7	0000 0010 10s
0	8	0000 0001 1101 s
0	9	0000 0001 1000 s
0	10	0000 0001 0011 s
0	11	0000 0001 0000 s
0	12	0000 0000 1101 0s
0	13	0000 0000 1100 1s
0	14	0000 0000 1100 0s
0	15	0000 0000 1011 1s
0	16	0000 0000 0111 11s
0	17	0000 0000 0111 10s
0	18	0000 0000 0111 01s
0	19	0000 0000 0111 00s
0	20	0000 0000 0110 11s
0	21	0000 0000 0110 10s
0	22	0000 0000 0110 01s
0	23	0000 0000 0110 00s
0	24	0000 0000 0101 11s
0	25	0000 0000 0101 10s
0	26	0000 0000 0101 01s
0	27	0000 0000 0101 00s
0	28	0000 0000 0100 11s
0	29	0000 0000 0100 10s
0	30	0000 0000 0100 01s
0	31	0000 0000 0100 00s
0	32	0000 0000 0011 000s
0	33	0000 0000 0010 111s
0	34	0000 0000 0010 110s
0	35	0000 0000 0010 101s
0	36	0000 0000 0010 100s
0	37	0000 0000 0010 011s
0	38	0000 0000 0010 010s
0	39	0000 0000 0010 001s
0	0	0000 0000 0010 000s
1	1	011s
1	2	0001 10s
1	3	0010 0101 s
1	4	0000 0011 00s
1	5	0000 0001 1011 s

1	6	0000 0000 1011 0s
1	7	0000 0000 1010 1s
1	8	0000 0000 0011 111s
1	9	0000 0000 0011 110s
1	10	0000 0000 0011 101s
1	11	0000 0000 0011 100s
1	12	0000 0000 0011 011s
1	13	0000 0000 0011 010s
1	14	0000 0000 0011 001s
1	15	0000 0000 0001 0011s
1	16	0000 0000 0001 0010s
1	17	0000 0000 0001 0001s
1	18	0000 0000 0001 0000s
2	1	0101 s
2	2	0000 100s
2	3	0000 0010 11s
2	4	0000 0001 0100 s
2	5	0000 0000 1010 0s
3	1	0011 1s
3	2	0010 0100 s
3	3	0000 0001 1100 s
3	4	0000 0000 1001 1s
4	1	0011 0s
4	2	0000 0011 11s
4	3	0000 0001 0010 s
5	1	0001 11s
5	2	0000 0010 01s
5	3	0000 0000 1001 0s
6	1	0001 01s
6	2	0000 0001 1110 s
6	3	0000 0000 0001 0100s
7	1	0001 00s
7	2	0000 0001 0101 s
8	1	0000 111s
8	2	0000 0001 0001 s
9	1	0000 101s
9	2	0000 0000 1000 1s
10	1	0010 0111 s
10	2	0000 0000 1000 0s
11	1	0010 0011 s
11	2	0000 0000 0001 1010s
12	1	0010 0010 s
12	2	0000 0000 0001 1001s
13	1	0010 0000 s
13	2	0000 0000 0001 1000s
14	1	0000 0011 10s
14	2	0000 0000 0001 0111s
15	1	0000 0011 01s
15	2	0000 0000 0001 0110s
16	1	0000 0010 00s
16	2	000 0000 0001 0101s
17	1	0000 0001 1111 s
18	1	0000 0001 1010 s
19	1	0000 0001 1001 s
20	1	0000 0001 0111 s

21	1	0000 0001 0110 s
22	1	0000 0000 1111 1s
23	1	0000 0000 1111 0s
24	1	0000 0000 1110 1s
25	1	0000 0000 1110 0s
26	1	0000 0000 1101 1s
27	1	0000 0000 0001 1111s
28	1	0000 0000 0001 1110s
29	1	0000 0000 0001 1101s
30	1	0000 0000 0001 1100s
31	1	0000 0000 0001 1011s
ESCAPE	-	0000 01

Table 2-D.15. Combination codes for DCT quantized coefficients
s = 0 for positive level, s = 1 for negative level

RUN-LENGTH	VLC CODE
0	0000 00
1	0000 01
2	0000 10
.	.
62	1111 10
63	1111 11

Table 2-D.16. Zero Run length codes

LEVEL	VLC CODE
-256	FORBIDDEN
-255	1000 0000 0000 0001
-254	1000 0000 0000 0010
.	.
-129	1000 0000 0111 1111
-128	1000 0000 1000 0000
-127	1000 0001
-126	1000 0010
.	.
-2	1111 1110
-1	1111 1111
0	FORBIDDEN
1	0000 0001
2	0000 0010
.	.
126	0111 1110
127	0111 1111
128	0000 0000 1000 0000
129	0000 0000 1000 0001
.	.
254	0000 0000 1111 1110
255	0000 0000 1111 1111

Table 2-D.17. Level codes for DCT quantized coefficients

Using tables 2-D.15 through 2-D.17 we can derive the VLC codes for the example of Table 2-D.14:

RUN	VALUE	CODE	COMMENT
1	2	0001 100	
0	4	0000 1100	
0	-3	0010 11	
3	-5	0000 0100 0011 1111 1011	esc seq
0	1	110	
14	130	0000 0100 1110 0000 0000 1000 0010	esc seq
EOB		10	

Table 2-D.18. Example Run Lengths, Values, and VLC Codes

The first three codes in Table 2-D.18 are derived directly from Table 2-D.15. The next code cannot since Table 2-D.15 does not contain an entry corresponding to a run length of 3 and a level of 5. Instead the escape code 000001 is used. This is followed by the six-bit code 000011 from Table 2-D.16 indicating a run length of 3. Last we append the eight-bit code 11111011 from Table 2-D.17 indicating a level of -5.

After the last coefficient has been coded, an EOB code is used to inform the decoder that there are no more quantized coefficients in the current 8 by 8 block. This EOB code is used even if the last quantized coefficient is at the bottom right of the block.

There are two codes for the 0,1 run length, level combination, as indicated in Table 2-D.15. Intra block coding always has the first quantized coefficient, the DC quantized coefficient, coded using the DC size method. Consequently intra blocks always use the code 11s to denote a run length, level combination of 0,1. It will be seen later that predictively coded blocks code the DC quantized coefficient differently, and may use the shorter code.

2-D.6.4 Coding P-Pictures

As in I pictures, each P picture is divided up into one or more slices, which are, in turn, divided into macroblocks. Coding is more complex than for I pictures, since motion-compensated macroblocks may be constructed. The difference between the motion compensated macroblock and the current macroblock is transformed with a 2-dimensional DCT giving an array of 8 by 8 transform coefficients. The coefficients are quantized to produce a set of quantized coefficients. The quantized coefficients are then encoded using a run-length value technique.

As in I pictures, the encoder needs to store the decoded P picture since this may be used as the starting point for motion compensation. Therefore, the encoder will reconstruct the image from the quantized coefficients.

In coding P pictures, the encoder has more decisions to make than in the case of I pictures. These decisions are: how to divide the picture up into slices, determine the best motion vectors to use, decide whether to code each macroblock as intra or predicted, and how to set the quantizer scale.

2-D.6.4.1 Slices in P-Pictures

P pictures are divided into slices in the same way as I pictures. The same considerations as to the best method of dividing a picture into slices apply, see 2-D.5.4.

2-D.6.4.2 Macroblocks in P Pictures

Slices are divided into macroblocks in the same way as for I pictures The major difference is the complexity introduced by motion compensation.

The macroblock header may contain stuffing. The position of the macroblock is determined by the macroblock address. Whereas the macroblock address increment within a slice for I pictures is restricted to one, it may be larger for P pictures. Any macroblocks thus skipped over are called "skipped macroblocks". The decoder copies them from the previous picture

into the current picture. Skipped macroblocks are as predicted macroblocks with a zero motion vector for which no additional correction is available. They require very few bits to transmit.

The next field in the macroblock header defines the macroblock type.

Macroblock Types in P Pictures

There are eight types of macroblock in P pictures:

TYPE	VLC	INTRA	MOTION FORWARD	CODED PATTERN	QUANT
pred-mc	1		1	1	
pred-c	01			1	
pred-m	001		1		
intra-d	0001 1	1			
pred-mcq	0001 0		1	1	1
pred-cq	0000 1			1	1
intra-q	0000 01	1		1	
skipped					

Table 2-D.19 Macroblock type VLC for P pictures (Table 2-B.2b)

Not all possible combinations of motion compensation, coding, quantization, and intra coding occur. For example, with intracoded macroblocks, intra-d and intra-q, motion vectors are not transmitted.

Skipped macroblocks have no VLC code. Instead they are coded by having the macroblock address increment code skip over them.

Quantizer Scale

If the macroblock type is pred-mcq, pred-cq or intra-q, i.e. if the QUANT column in Table 2-D.19 has a 1, then a quantizer scale is transmitted .

If the macroblock types are pred-mc, pred-c or intra-d, then the DCT correction is coded using the previously established value for the quantizer scale.

Motion Vectors

If the macroblock type is pred-m, pred-mc or pred-mq, i.e. if the MOTION FORWARD column in Table 2-D.19 has a 1, then horizontal and vertical forward motion vectors are transmitted in succession..

Coded Block Pattern

If the macroblock type is pred-c, pred-mc, pred-cq or pred-mcq, i.e. if the CODED PATTERN column in Table 2-D.19 has a 1, then a coded block pattern is transmitted. This informs the decoder which of the six blocks in the macroblock are coded, i.e. have transmitted DCT quantized coefficients, and which are not coded, i.e. have no additional correction after motion compensation.

The coded block pattern is a number from 0 to 63 that indicates which of the blocks are coded, i.e. have at least one transmitted coefficient, and which are not coded. To understand the structure of the coded block pattern, we refer to Figure 2-D.5 and introduce the variables PN to indicate the status of each of the six blocks. If block N is coded then PN has the value one, if it is not coded then PN is zero. The coded block pattern is defined by the equation:

CBP = 32*P0 + 16*P1 + 8*P2 + 4*P3 + 2*P4 + P5

This is equivalent to the definition given in Clause 2.4.3.6.

For example, if the top two luminance blocks and the Cb block are coded, and the other three are not, then P0 = 1, P1 = 1, P2 = 0, P3 = 0, P4 = 1, and P5 = 0. The coded block pattern is:

$$CBP = 32*1 + 16*1 + 8*0 + 4*0 + 2*1 + 0 = 50$$

Certain patterns are more common than others. Advantage is taken of this fact to increase the coding efficiency and transmit a VLC representing the coded block pattern, rather than the coded block pattern itself. The VLC codes are given in the following table:

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

Table 2-D.20 VLC table for Coded Block Pattern

Thus the coded block pattern of the previous example, 50, would be represented by the code "00010110".

Note that there is no code representing the state in which none of the blocks are coded, a coded block pattern equal to zero. Instead, this state is indicated by the macroblock type.

For macroblocks in I pictures, and for intra coded macroblocks in P and B pictures, the coded block pattern is not transmitted, but is assumed to have a value of 63, i.e. all the blocks in the macroblock are coded.

The use of coded block patterns instead of transmitting end of block codes for all blocks follows the practice in H.261.

2-D.6.4.3 Selection of Macroblock Type

An encoder has the difficult task of choosing between the different types of macroblocks.

An exhaustive method is to try coding a macroblock to the same degree of accuracy using each type, then choose the type that requires the least number of coding bits.

A simpler method, and one that is computationally less expensive, is to make a sequential series of decisions. One way to order these decisions is:

1: motion compensation or no motion compensation, i.e. is a
motion vector transmitted or is it assumed to be zero.
2: intra or non intra coding, i.e. is the macroblock type intra
or is it predicted using the motion vector found in step 1.
3: if the macroblock type is non-intra, is it coded or not
coded, i.e. is the residual error large enough to be coded
using the DCT transform.
4: decide if the quantizer scale is satisfactory or should be
changed.

These decisions are summarized in the following diagram:

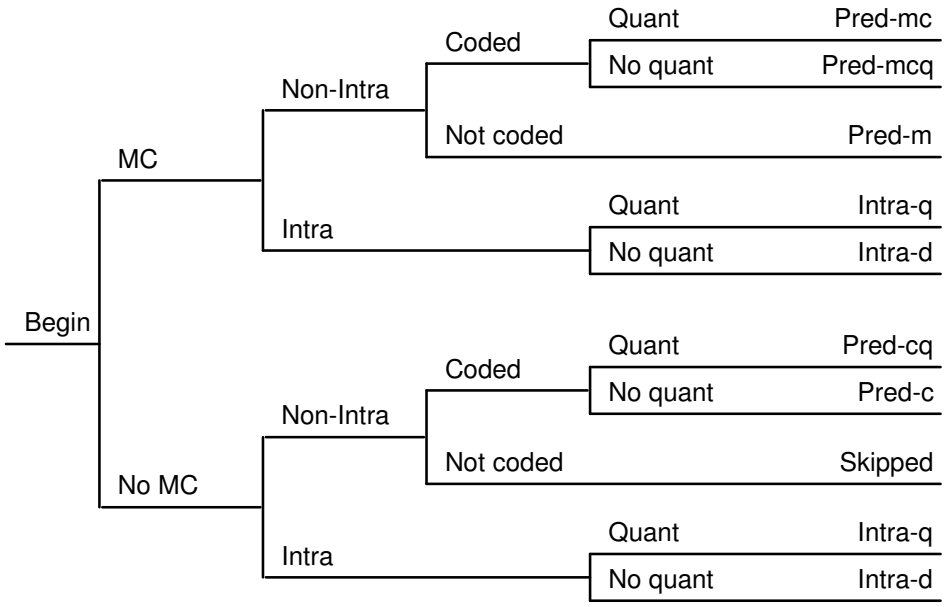


Figure 2-D.32 Selection of Macroblock Types in P Pictures

The four decision steps are discussed in the next four clauses.

Motion Compensation Decision

The encoder has an option on whether to transmit motion vectors or not for predictive-coded macroblocks. If the motion vector is zero then some code may be saved by not transmitting the motion vectors. Thus one algorithm is to search for the best match and compare the error of the predicted block with that formed with a zero vector. If the motion-compensated block is only slightly better than the uncompensated block, using the selected block matching criterion, then the zero vector might be used to save coding bits.

An algorithm used in the development of both H.261 and this standard was as follows:

The block-matching criterion is the sum of absolute differences of all the luminance pels in a macroblock, when compared with the motion-compensated macroblock. If the sum is M for the motion-compensated block, and Z for the zero vector, then the decision of whether to make use of the motion vector is defined by Figure 2-D.33.

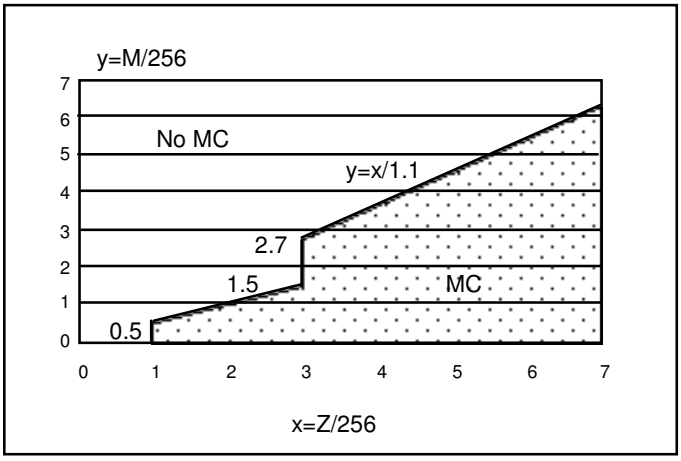


Figure 2-D.33 Characteristic MC/No MC

Points on the line dividing the No MC (no motion compensation, i.e. zero vector), from the MC (motion compensation) regions, are regarded as belonging to the no motion compensation region.

It can be seen that if the error is sufficiently low, then no motion compensation should be used. Thus a way to speed up the decision is to examine the zero vector first and decide if it is good enough.

The foregoing algorithm was designed for telecommunications sequences in which the camera was fixed, and in which any movement of the background caused by the "drag along effect" of nearby moving objects was very objectionable. Great care was taken to reduce this spurious motion, and this accounts for the curious shape of the boundary between the two regions in Figure 2-D.33.

Intra/Non–Intra Coding Decision

After the encoder has determined the best motion vector, it is in a position to decide whether to use it, or disregard it entirely and code the macroblock as intra. The obvious way to do this is to code the block as intra, and compare the total number of bits required when coded as motion compensated plus correction with the same quantizer scale. The method using the fewest bits may be used.

This may be too computationally expensive for the encoder to do, and a faster algorithm may be required. One such algorithm, used in the simulation model during the development of this standard, was based on the variance of the luminance component of the macroblock. The variance of the current macroblock and of the difference macroblock (current - motion-compensated previous) is compared. It is calculated using the method represented by the following C program fragment. Note that in calculating the variance of the difference macroblock, the average value is assumed to be zero.

```
int pelp[16][16]; /*Pixel values in the Previous macroblock
      after motion compensation */
int pelc[16][16]; /*Pixel values in the Current macroblock */
long dif;           /*Difference between two pel values*/
long sum;           /*Sum of the current pel values*/
long vard;          /*Variance of the Difference macroblock*/
long varc;          /*Variance of the Current macroblock*/
int x,y;            /*coordinates*/

sum = 0;
vard = 0;
varc = 0;
for (y=0;y<16;y++)
{
```

```
        for (x=0;x<16;x++)
        {
              sum = sum + pelc[y][x];
              varc = varc + pelc[y][x]*pelc[y][x];

              dif = pelc[y][x] - pelp[y][x];
              vard = vard + dif*dif;
        }
  }
  vard = vard/256;    /* assumes mean is close to zero */
  varc = varc/256 - (sum/256)*(sum/256);
```

The decision as to whether to code as intra or non intra is then based on Figure 2-D.34.

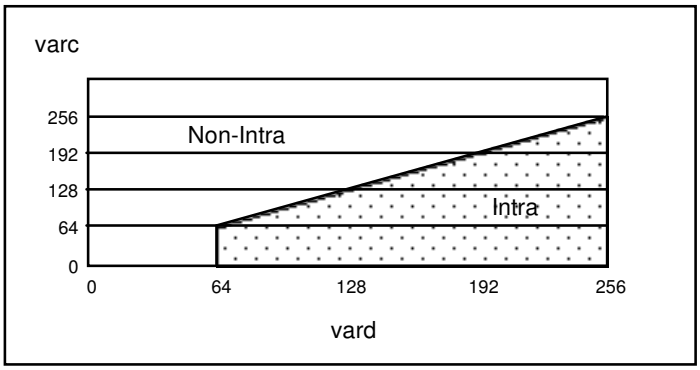


Figure 2-D.34 Characteristic Intra/Non-Intra

Points on the line dividing the non-intra from the intra regions, are regarded as belonging to the non-intra region.

It can be seen that if var is small enough the macroblock should be coded as intra. This may allow the algorithm to be speeded up slightly.

Coded/Not Coded Decision

The choice of coded or not coded is a result of quantization; when all coefficients are zero then a block is not coded. A macroblock is not coded if no block in it is coded, else it is coded.

Quantizer/No Quantizer Decision

Generally the quantizer scale is changed based on local scene content to improve the picture quality, and on the buffer fullness of the model decoder to prevent overflow and underflow.

2-D.6.4.4 DCT Transform

Coefficients of intra blocks are transformed into quantized coefficients in the same way that they were for intra blocks in I pictures. Prediction of the DC coefficient differs, however. The DC predicted values are all set to 128 for intra blocks in P and B pictures, unless the previous block was intra coded.

Coefficients of non-intra blocks are coded in a similar way. The main difference is that the coefficients to be transformed represent differences between pel values rather than the pel values themselves. The differences are obtained by subtracting the motion-compensated pel values from the previous picture from the pel values in the current macroblock. Since the coding is of differences, there is no spatial prediction of the DC term.

2-D.6.4.5 Quantization of P Pictures

Intra macroblocks in P and B pictures are quantized using the same method as described for I pictures.

Non-intra macroblocks in P and B pictures are quantized using the quantizer scale and the non-intra quantization matrix. Both DC and the AC coefficients are quantized the same way.

The following quantization formula was derived by inverting the reconstruction formula given in Clause 2.4.4.2. Note that the divisor indicates truncation towards zero.

```
int coefforig; /*original coefficient*/
int coeffqant; /*quantized coefficient*/
int coeffrec;  /*reconstructed coefficient*/
int niqmatrix; /*non-intra quantization matrix*/
int quantscale; /*quantizer scale*/

coeffqant = (8 * coefforig) / (quantscale * niqmatrix);
```

The process is illustrated below:

niqmatrix					
quantscale					
coefforig	-3	-1	2	4	6
coeffqant					
coeffrec					

The last line shows the reconstructed coefficient values. The following diagram shows the characteristics of this quantizer. The flat spot around zero gives this type of quantizer its name: a dead-zone quantizer.

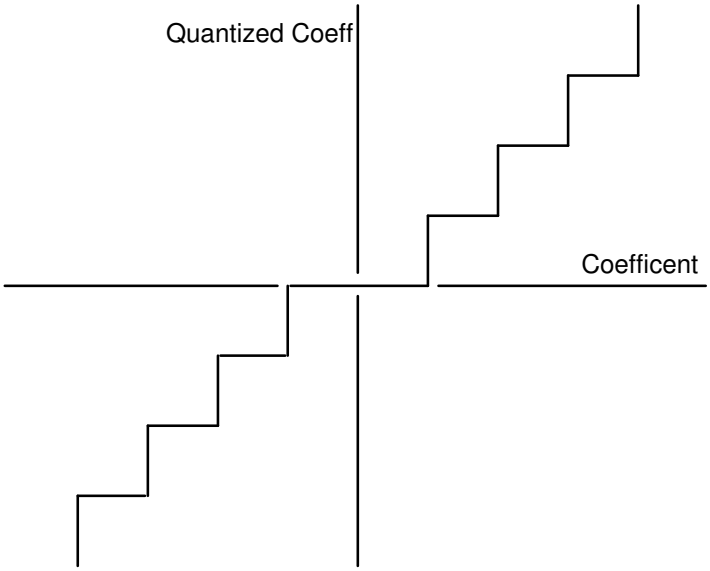


Figure 2-D.35 Dead Zone Quantizer Characteristic

2-D.6.4.6 Coding of Quantized Coefficients

Coding of Intra Blocks

Intra blocks in P pictures are coded the same way as intra blocks in I pictures. The only difference lies is the prediction of the DC coefficient. The DC predicted value is 128, unless the previous block was intra coded.

Coding of Non–Intra Blocks

The coded block pattern is transmitted indicating which blocks have coefficient data. These are coded in a similar way to the coding of intra blocks except that the DC coefficient is coded in the same way as the AC coefficients.

2-D.6.5 Coding B-Pictures

As in I and P pictures, each B picture is divided up into one or more slices, which are, in turn, divided into macroblocks. Coding is more complex than for P pictures, since several types of motion compensated macroblock may be constructed: forward, backward, and interpolated. The difference between the motion-compensated macroblock and the current macroblock is transformed with a 2-dimensional DCT giving an array of 8 by 8 transform coefficients. The coefficients are quantized to produce a set of quantized coefficients. The quantized coefficients are then encoded using a run-length value technique.

The encoder does not need to store the decoded B pictures since they will not be used for motion compensation.

In coding B pictures, the encoder has more decisions to make than in the case of P pictures. These decisions are: how to divide the picture up into slices, determine the best motion vectors to use, decide whether to use forward or backward or interpolated motion compensation or to code as intra, and how to set the quantizer scale.

2-D.6.5.1 Slices in B-Pictures

B pictures are divided into slices in the same way as I and P pictures. Since B pictures are not used as a reference for motion compensation, errors in B pictures are slightly less important than in I or P pictures. Consequently, it might be appropriate to use fewer slices for B pictures.

2-D.6.5.2 Macroblocks in B Pictures

Slices are divided into macroblocks in the same way as for I pictures.

The macroblock header may contain stuffing. The position of the macroblock is determined by the macroblock address.Whereas the macroblock address increment within a slice for I pictures is restricted to one, it may be larger for B pictures. Any macroblocks thus skipped over are called "skipped macroblocks". Skipped macroblocks in B pictures differ from skipped macroblocks in P pictures. Whereas in P pictures skipped macroblocks have a motion vector equal to zero, in B pictures skipped macroblocks have the same motion vector and the same macroblock type as the previous block. They also have the same macroblock type as the previous block, which cannot be intra coded, but there is no additional DCT correction. They require very few bits to transmit.

The next field in the macroblock header defines the macroblock type.

Macroblock Types in B Pictures

There are 12 types of macroblock in B pictures:

TYPE	VLC	INTRA	MOTION FORWARD	MOTION BACKWARD	CODED PATTERN	QUANT
pred-i	10		1	1		
pred-ic	11		1	1	1	
pred-b	010			1		
pred-bc	011			1	1	
pred-f	0010		1			
pred-fc	0011		1		1	
intra-d	0001 1	1				
pred-icq	0001 0		1	1	1	1
pred-fcq	0000 11		1		1	1
pred-bcq	0000 10			1	1	1
intra-q	0000 01	1				1
skipped						

Table 2-D.21 Macroblock type VLC for B pictures (Table 2-B.2d)

Compared with P pictures, there are extra types due to the introduction of the backward motion vector. If only a forward motion vector is present, then the motion compensated-macroblock is constructed from a previous picture, as in P pictures. If only a backward motion vector is present, then the motion-compensated macroblock is constructed from a future picture. If both forward and backward motion vectors are present, then motion-compensated macroblocks are constructed from both previous and future pictures, and the result is averaged to form the "interpolated" motion-compensated macroblock.

Quantizer Scale

If the macroblock type is pred-icq, pred-fcq, pred-bcq, or intra-q, i.e. if the QUANT column in Table 2-D.21 has a 1, then a quantizer scale is transmitted. For the remaining macroblock types, the DCT correction is coded using the previously established value for the quantizer scale.

Motion Vectors

If the MOTION FORWARD column in Table 2-D.21 has a 1, then horizontal and vertical forward motion vectors are transmitted in succession. If the MOTION BACKWARD column in Table 2-D.21 has a 1, then horizontal and vertical backward motion vectors are transmitted in succession. If both types are present then four component vectors are transmitted in the following order:

horizontal forward
vertical forward
horizontal backward
vertical backward

Coded Block Pattern

If the CODED PATTERN column in Table 2-D.21 has a 1, then a coded block pattern is transmitted. This informs the decoder which of the six blocks in the macroblock are coded, i.e. have transmitted DCT quantized coefficients, and which are not coded, i.e. have no additional correction after motion compensation.

2-D.6.5.4 Selection of Macroblock Type

The encoder has more types of macroblock to choose from in B pictures, than in P pictures, and consequently its job is a little harder.

For the simulation model, the following sequential decision algorithm was used:

1: motion compensation mode, i.e. is forward or backward or interpolative motion compensation best? What of the vector values?
2: intra or non intra coding, i.e. is the macroblock type intra or is it motion compensated using mode and the vectors found in step 1?
3: if the macroblock type is non-intra, is it coded or not coded, i.e. is the residual error large enough to be coded using the DCT transform.
4: decide if the quantizer scale is satisfactory or should be changed.

These decisions are summarized in the following diagram:

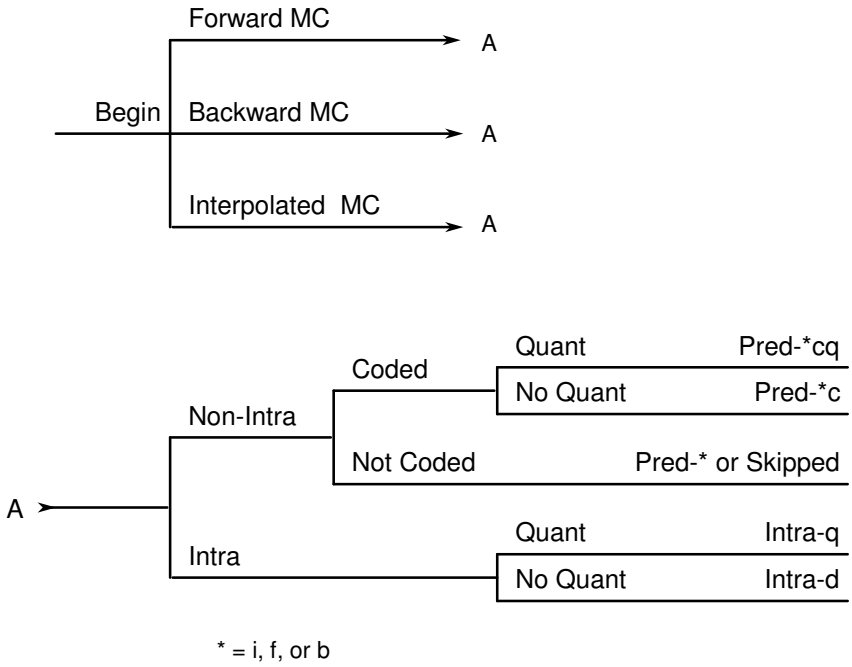


Figure 2-D.36 Selection of Macroblock Type in B Pictures

The four decision steps are discussed in the next four clauses.

Selecting Motion–Compensation Mode

An encoder should attempt to code B pictures using skipped macroblocks if possible. This suggests that the encoder should first examine the case where the motion compensation is the same as for the previous macroblock. If the previous macroblock was non-intra, and if the motion-compensated block is good enough, there will be no additional DCT correction required and the block can be coded as skipped.

If the macroblock cannot be coded as skipped, then the following procedure may be followed.

For the simulation model, the selection of a motion compensation mode for a macroblock was based on the minimization of a cost function. The cost function was the MSE of the luminance difference between the motion-compensated macroblock and the current macroblock. The encoder calculated the best motion-compensated macroblock for forward motion compensation. It then calculated the best motion-compensated macroblock for backward motion compensation by a similar method. Finally it averaged the two motion-compensated macroblocks to produce the interpolated macroblock. It then selected the one that had the smallest mean square difference between it and the current macroblock. In the event of a tie, interpolative mode was chosen.

Intra/Non–Intra Coding

Based on the smallest MSE, a decision is made between the best of the three possible prediction modes and the Intra mode. The calculation is similar to that of P pictures. The variances of the difference macroblock, vard, and of the current macroblock, varc, are calculated.

In the simulation model the final decision was based on simply the macroblock type with the smallest variance. If the two variances were equal, non-intra coding was chosen.

Coded/Not–Coded Decision

The choice of coded or not coded is a result of quantization, when all coefficients are zero then a block is not coded. A macroblock is not coded if no block in it is coded, else it is coded.

2-D.6.5.5 DCT Transform

Coefficients of blocks are transformed into quantized coefficients in the same way that they are for blocks in P pictures.

2-D.6.5.6 Quantization

Blocks in B pictures are quantized in the same way as for P pictures.

2-D.6.5.7 Coding

Blocks in B pictures are coded the same way as blocks in P pictures.

2-D.6.6 Coding D-Pictures

D pictures contain only low frequency information. They are intended to be used for fast visible search modes. It is intended that the low frequency information they contain is sufficient for the user to locate the desired video.

D pictures are not part of the constrained parameter bitstream.

D pictures are coded as the DC coefficients of blocks. There is a bit transmitted for the macroblock type, although only one macroblock type exists. In addition there is a bit denoting end of macroblock.

2-D.7 DECODING MPEG VIDEO

2-D.7.1 Decoding a Sequence

Decoding for Forward Play

At the beginning of a sequence, the decoder will decode the sequence header including the sequence parameters. If the bitstream is not constrained and a parameter exceeds the capability of the decoder, then the decoder should report this. If the decoder determines that it can decode the bitstream, then it will set up its parameters to match those defined in the sequence header. This will include the horizontal and vertical resolutions and aspect ratio, the bit rate, and the quantization matrices.

Next the decoder will decode the group of pictures header, including the closed_gop and broken_link information, and take any appropriate action. It will decode the first picture header in the group of pictures and, for constant bit rate operation, determine the buffer_fullness. It will then delay decoding the rest of the sequence until the input buffer is filled to the correct level. Only by doing this can the decoder be sure than no buffer overflow or underflow problems will occur during decoding. Normally the input buffer size will be larger than the minimum required by the bitstream, giving a range of fullness at which the decoder may start to decode.

If the closed-gop flag is 0, indicating that the group is open, and the broken_link flag is 1, then any B pictures preceding (in display order) the first I picture in the group cannot be decoded. The decoder may adopt one of several strategies. It may display the first I picture during the time that the undecodable B pictures would be displayed. This strategy maintains audio synchronization and buffer fullness. However it is likely that the broken link has occured because of post coding editing, in which case audio may be discontinuous, and the buffer fullness may also exhibit a discontinuity. Therefore a better strategy might be to discard the B pictures entirely, and delay decoding the I picture until the buffer fullness is within limits.

If play begins from a random point in the bitstream, the decoder should discard all the bits until it finds a sequence start code, a group of pictures start code, or a picture start code which introduces an I picture. The slices and macroblocks in the picture are decoded and written into a display buffer, and perhaps into another buffer. The decoded pictures may be post processed and displayed in the order defined by the temporal reference at the picture rate defined in the sequence header.

Subsequent pictures are processed at the appropriate times to avoid buffer overflow and underflow.

Decoding for Fast Play

Fast forward can be supported by D pictures. It can also be supported by an appropriate spacing of I pictures in a sequence. For example, if I pictures were spaced regularly every 10 pictures, then a decoder might be able to play the sequence at 10 times the normal speed by decoding and displaying only the I pictures. This simple concept places considerable burdens on the media and the decoder. The media must be capable of speeding up and delivering 10 times the data rate, the decoder must be capable of accepting this higher data rate and decoding the I pictures. Since I pictures typically require significantly more bits to code than P or B pictures, the decoder will have to decode significantly more than 10% of the data, even if it can search for picture start codes and discard the data for P and B pictures.

For example, a sequence might be coded as follows:

I B **P** B **P** B **P** B **P** B **I** B **P** B **P** B **P** B **P** B **I** ...

Assume that the average code size per picture is C, that each B picture requires 0.3C, that each P picture requires 1.5C, and that each I picture requires 2.5C, then the I pictures require 25% of the code for their 10% of the display time.

Another way to achieve fast forward in a constant bit rate application, is for the media itself to sort out the I pictures and transmit them. This would allow the data rate to remain constant. Since this selection process can be made to produce a valid

MPEG video bitstream, the decoder should be able to decode it. If every I picture of the preceding example were selected, then one I picture would be transmitted every 2.5 picture periods, and the speed up rate would be 10/2.5 = 4 times. The decoder might be able to display the I pictures at exactly 2.5 periods, or it might alternate displays at 2 and 3 periods.

If alternate I pictures of the preceding example were selected, then one I picture would again be transmitted every 2.5 picture periods, but the speed up rate would be 20/2.5 = 8 times.

If one in N I pictures of the preceding example were selected, then the speed up rate would be 10N/2.5 = 4N times.

Decoding for Pause and Step Modes

Decoding for pause requires the decoder to be able to control the incoming bitstream, and display a decoded picture without decoding any additional pictures. If the decoder has full control over the bitstream, then it can be stopped for pause and resumed when play resumes. If the decoder has less control, as in the case of a CD ROM, then there may be a delay before play can be resumed.

Decoding for Reverse Play

To decode a bitstream and play in reverse, the decoder must decode each group of pictures in the forward direction, store the decoded pictures, then display them in reverse order. This places severe storage requirements on the decoder in addition to any problems in gaining access to the coded bitstream in the correct order.

To reduce decoder memory requirements, groups of pictures should be small. There is no mechanism in the syntax for the encoder to state what the decoder requirements are in order to play in reverse.

The amount of display buffer storage may be reduced by reordering the pictures, either by having the storage unit read and transmit them in another order, or by reordering the coded pictures in a decoder buffer. To illustrate the savings, consider the following typical group of pictures:

B	B	**I**	B	B	**P**	B	B	**P**	B	B	**P**	pictures in display order
0	1	2	3	4	5	6	7	8	9	10	11	temporal reference
I	B	B	**P**	B	B	**P**	B	B	**P**	B	B	pictures in decoding order
2	0	1	5	3	4	8	6	7	11	9	10	temporal reference
I	**P**	**P**	**P**	B	B	B	B	B	B	B	B	pictures in new order
2	5	8	11	10	9	7	6	4	3	1	0	temporal reference

Figure 2-D.37 Example Group of Pictures

The decoder would decode the pictures in the new order, and display them in the reverse of the normal display order. Since the B pictures are not decoded until they are ready to be displayed, the display buffer storage is minimized. The first two B pictures, 0 and 1, would remain stored in the input buffer until the last P picture in the previous group of pictures is decoded.

2-D.8 POST PROCESSING

2-D.8.1 Editing

Editing of a video sequence is best done before compression, but situations arise where only the coded bitstream is available. One possible method would be to decode the bitstream, perform the required editing, and recode the bitstream. This usually leads to a loss in video quality, and it is better, if possible, to edit the coded bitstream itself.

Although editing may take several forms, the following discussion pertains only to editing at the picture level: deletion of coded video material from a bitstream, insertion of coded video material into a bitstream, or rearrangement of coded video material within a bitstream.

If editing is anticipated, e.g. clip video is provided analogous to clip art for still pictures, then the video can be encoded with well defined cutting points. These cutting points are places at which the bitstream may be broken apart or joined. Each cutting point should be followed by a closed group of pictures. This allows smooth play after editing.

To allow the decoder to play edited video without having to adopt any unusual strategy to avoid buffer overflow and underflow, the encoder should make the buffer fullness take the same value at the first I picture following every cutting point. This value should be the same as that of the first picture in the sequence. If this suggestion is not followed, then the editor may make an adjustment either by adding padding (stuffing bits or macroblocks) or by recoding a few images to make them smaller.

If the buffer fullness is mismatched and the editor makes no correction, then the decoder will have to make some adjustment when playing over an edited cut. For example, consider a coded sequence consisting of three clips, A, B, and C in order. Assume that clip B is completely removed by editing, so that the edited sequence consists only of clip A followed immediately by clip C as illustrated below:

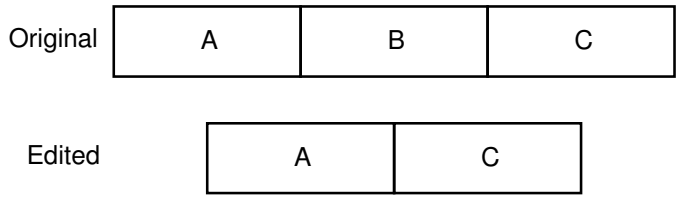


Figure 2-D.38 Sequences

Assume that in the original sequence the buffer is three quarters full at the beginning of clip B, and one quarter full at the beginning of clip C. A decoder playing the edited sequence will encounter the beginning of clip C with its buffer three quarters full, but the first picture in clip C will contain a buffer_fullness value corresponding to a quarter full buffer. In order to avoid buffer overflow problems, the decoder may try to pause the input bitstream, or discard pictures without displaying them (preferably B pictures), or change the decoder timing.

For another example, assume that in the original sequence the buffer is one quarter full at the beginning of clip B, and three quarters full at the beginning of clip C. A decoder playing the edited sequence will encounter the beginning of clip C with its buffer one quarter full, but the first picture in clip C will contain a buffer_fullness value corresponding to a three quarters full buffer. In order to avoid buffer underflow problems, the decoder may display one or more pictures for longer than the normal time.

If editing was not specifically provided for in the coded bit stream, or if it must be available at any picture, then the editing task is more complex, and places a greater burden on the decoder to manage buffer overflow and underflow problems.

The easiest editing task is to cut at the beginning of groups of pictures. If the group of pictures following the cut is open, which can be detected by examining the closed_gop flag in the group of pictures header,, then the editor must set the broken_link bit to 1 to indicate to the decoder that the previous group of pictures cannot be used for decoding any B pictures.

2-D.8.2 Resampling

The decoded bitstream may not match the picture rate or the spatial resolution of the display device. In this, quite frequent, situation the decoded video must be resampled or scaled.

One example, considered under preprocessing, is the case where the decoded video has SIF resolution and must be converted to CCIR 601 resolution.

2-D.8.2.1 Conversion of MPEG SIF to CCIR 601 Format

A SIF is converted to its corresponding CCIR 601 format by spatial upsampling. A linear phase FIR filter is applied after the insertion of zeroes between samples.
A filter that can be used for upsampling the luminance is shown in Figure 2-D.39:

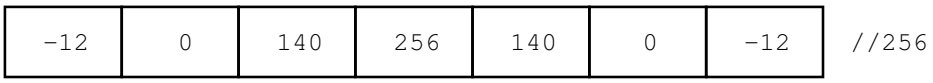


Figure 2-D.39 Upsampling Filter for Luminance

At the end of the lines some special technique, such as replicating the last pel, must be adopted. The following example shows a horizontal line of 8 pels being upsampled to 16 pels using the suggested filter:

```
      12  32  23  9   12  49  95  95
10  12  22  32  29  23  15  9   8   12  28  49  74  95  97  95
```

Figure 2-D.40 Example of upsampling a line of pels

According to CCIR Rec. 601 he chrominance samples need to be cosited with the luminance samples 1, 3, 5,... In order to achieve the proper location, the upsampling filter

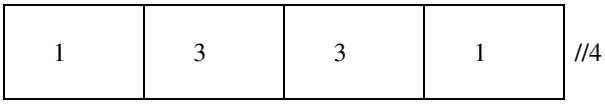


Figure 2-D.41 Upsampling filter for Chrominance

The SIF may be reconstructed by adding four black pels to each end of the horizontal luminance lines in the decoded bitmap, and two gray pels to each end of the horizontal chrominance lines. The luminance SIF may then be upsampled horizontally and vertically. The chrominance SIF should be upsampled once horizontally and twice vertically. This process is illustrated by the following diagram:

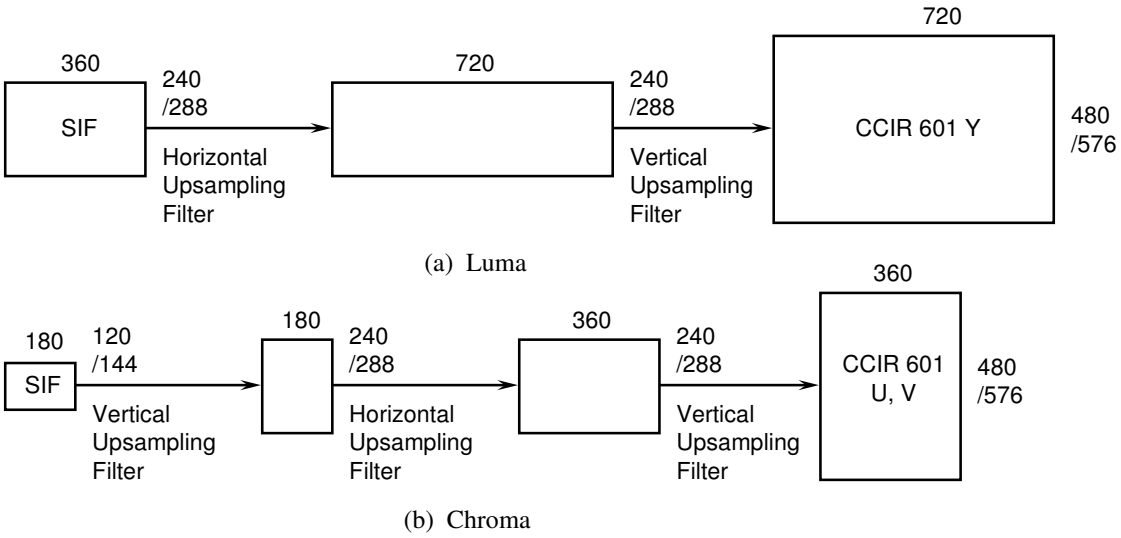


Figure 2-D.42 Simplified Decoder Block Diagram

2-D.8.2.2 Temporal Resampling

Since the picture rates are limited to those commonly used in the television industry, the same techniques may be applied. For example, conversion from 24 pps to 60 fps may be achieved by the technique of 3:2 pulldown.

Video coded at 25 pps can be converted to 50 fields per sec by displaying the original decoded lines in the odd CCIR 601 fields, and the interpolated lines in the even fields. Video coded at 29.97 pps or 30 pps may be converted to a field rate twice as large using the same method.

Video coded at 23.976 pps or 24 pps may be converted to 50 fields per sec by speeding it up by about 4% and decoding it as if it had been encoded at 25 pps. The decoded pictures could be displayed in the odd fields, and interpolated pictures in the even fields. The audio must be maintained in synchronization, either by increasing the pitch, or by speeding it up without a pitch change.

Video coded at 23.976 pps or 24 pps may be converted to 59.94 pps or 60 pps using the technique of 3:2 pull down.

2-ANNEX E (informative)

BIBLIOGRAPHY

1. Arun N. Netravali & Barry G. Haskell "Digital Pictures, representation and compression" Plenum Press, 1988

2. Didier Le Gall "MPEG: A Video Compression Standard for Multimedia Applications" Trans ACM, April 1991

4. C Loeffler, A Ligtenberg, G S Moschytz "Practical fast 1-D DCT algorithms with 11 multiplications" IEEE, Feb 1989

5. See the Normative Reference for CCIR Rec 601

6. IEC Standard
Publication 461
Second edition 1986
"Time and control code for video tape recorders"

7. CCITT Recommendation H.261
Codec for audiovisual services at px64 kbit/s"
Geneva, 1990

8. IEEE Standard Specification for the Implementation of 8 by 8 Inverse Discrete Cosine Transform" P1180/D2 July 18 1990

9. ISO 10918-1 (JPEG) "Digital compression and coding of continuous-tone still images"

10. H.A. Peterson, H. Peng, T. H. Morgan, W. B. Pennebaker "Quantization of Color Image Components in the DCT Domain" SPIE/IS&T 1991 Symposium on Electronic Imaging Science & Technology, San Jose, Feb. 1991