INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC29/WG11 N0702rev

Incorporating N702 Delta of 24 March

25 March 1994

Video

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

Recommendation H.262

ISO/IEC 13818-2

Draft International Standard

Draft of: 10:18 Friday 25 March 1994

CONTENTS

Foreword

The ITU-T (the ITU Telecommunication Standardisation Sector) is a permanent organ of the International Telecommunications Union (ITU). The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to developing telecommunication standards on a world-wide basis.

The World Telecommunication Standardisation Conference, which meets every four years, establishes the program of work arising from the review of existing questions and new questions among other things. The approval of new or revised Recommendations by members of the ITU-T is covered by the procedure laid down in the ITU-T Resolution No. 1 (Helsinki 1993). The proposal for Recommendation is accepted if 70% or more of the replies from members indicate approval.

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This specification is a committee draft that is being submitted for approval to the ITU-T, ISO-IEC/JTC1 SC29. It was prepared jointly by SC29/WG11, also known as MPEG (Moving Pictures Expert Group), and the Experts Group for ATM Video Coding in the ITU-T SG15. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution and communication. The Experts Group for ATM Video Coding was formed in 1990 to develop video coding standard(s) appropriate for B-ISDN using ATM transport.

In this specification Annex A, Annex B and Annex C contain normative requirements and are an integral part of this specification. Annex D, Annex E, Annex F and Annex G are informative and contain no normative requirements.

<u>ISO/IEC</u>

This International Standard is published in four Parts.

13818-1 systems —	specifies the system coding of the specification. It defines a multiplexed structure for combining audio and video data and means of representing the timing information needed to replay synchronised sequences in real-time.
13818-2 video —	specifies the coded representation of video data and the decoding process required to reconstruct pictures.
13818-3 audio —	specifies the coded representation of audio data.
13818-4 conformance—	specifies the procedures for determining the characteristics of coded bitstreams and for testing compliance with the requirements stated in 13818-1, 13818-2 and 13818-3.

I Introduction

I.1 Purpose

This Part of this specification was developed in response to the growing need for a generic coding method of moving pictures and of associated sound for various applications such as digital storage media, television broadcasting and communication. The use of this specification means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

I.2 Application

The applications of this specification cover, but are not limited to, such areas as listed below:

BSS	Broadcasting Satellite Service (to the home)
CATV	Cable TV Distribution on optical networks, copper, etc.
CDAD	Cable Digital Audio Distribution
DAB	Digital Audio Broadcasting (terrestrial and satellite broadcasting)
DTTB	Digital Terrestrial Television Broadcast
EC	Electronic Cinema
ENG	Electronic News Gathering (including SNG, Satellite News Gathering)
FSS	Fixed Satellite Service (e.g. to head ends)
HTT	Home Television Theatre
IPC	Interpersonal Communications (videoconferencing, videophone, etc.)
ISM	Interactive Storage Media (optical disks, etc.)
MMM	Multimedia Mailing
NCA	News and Current Affairs
NDB	Networked Database Services (via ATM, etc.)
RVS	Remote Video Surveillance
SSM	Serial Storage Media (digital VTR, etc.)

I.3 Profiles and levels

This specification is intended to be generic in the sense that it serves a wide range of applications, bitrates, resolutions, qualities and services. Applications should cover, among other things, digital storage media, television broadcasting and communications. In the course of creating this specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax. Hence this specification will facilitate the bitstream interchange among different applications.

Considering the practicality of implementing the full syntax of this specification, however, a limited number of subsets of the syntax are also stipulated by means of "profile" and "level". These and other related terms are formally defined in clause 3 of this specification.

A “profile” is a defined subset of the entire bitstream syntax that is defined by this specification. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by parameters in the bitstream. For instance it is possible to specify frame sizes as large as (approximately) 2^{14}

samples wide by 2^{14} lines high. It is currently neither practical nor economic to implement a decoder capable of dealing with all possible frame sizes.

In order to deal with this problem "levels" are defined within each profile. A level is a defined set of constraints imposed on parameters in the bitstream. These constraints may be simple limits on numbers. Alternatively they may take the form of constraints on arithmetic combinations of the parameters (e.g. frame width multiplied by frame height multiplied by frame rate).

Bitstreams complying with this specification use a common syntax. In order to achieve a subset of the complete syntax flags and parameters are included in the bitstream that signal the presence or otherwise of syntactic elements that occur later in the bitstream. In order to specify constraints on the syntax (and hence define a profile) it is thus only necessary to constrain the values of these flags and parameters that specify the presence of later syntactic elements.

I.4 The scalable and the non-scalable syntax

The full syntax can be divided into two major categories: One is the non-scalable syntax, which is structured as a super set of the syntax defined in ISO/IEC 11172-2. The main feature of the non-scalable syntax is the extra compression tools for interlaced video signals. The second is the scalable syntax, the key property of which is to enable the reconstruction of useful video from pieces of a total bitstream. This is achieved by structuring the total bitstream in two or more layers, starting from a standalone base layer and adding a number of enhancement layers. The base layer can use the non-scalable syntax, or in some situations conform to the ISO/IEC 11172-2 syntax.

I.4.1 Overview of the non-scalable syntax

The coded representation defined in the non-scalable syntax achieves a high compression ratio while preserving good image quality. The algorithm is not lossless as the exact sample values are not preserved during coding. Obtaining good image quality at the bitrates of interest demands very high compression, which is not achievable with intra picture coding alone. The need for random access, however, is best satisfied with pure intra picture coding. The choice of the techniques is based on the need to balance a high image quality and compression ratio with the requirement to make random access to the coded bitstream.

A number of techniques are used to achieve high compression. The algorithm first 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-sample by 16-line region of the picture. The difference signal, i.e., the prediction error, is further compressed using the discrete cosine transform (DCT) to remove spatial correlation before it is quantised in an irreversible process that discards the less important information. Finally, the motion vectors are combined with the residual DCT information, and encoded using variable length codes.

I.4.1.1 Temporal processing

Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intra coded pictures (I-Pictures) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only moderate compression. Predictive coded pictures (P-Pictures) are coded more efficiently using motion compensated prediction from a past intra or predictive coded picture and are generally used as a reference for further prediction. Bidirectionally-predictive coded pictures (B-Pictures) provide the highest degree of compression but require both past and future reference pictures for motion compensation. Bidirectionally-predictive coded pictures are never used as references for prediction (except in the case that the resulting picture is used as a reference in a spatially scalable enhancement layer). 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 among the three different picture types.

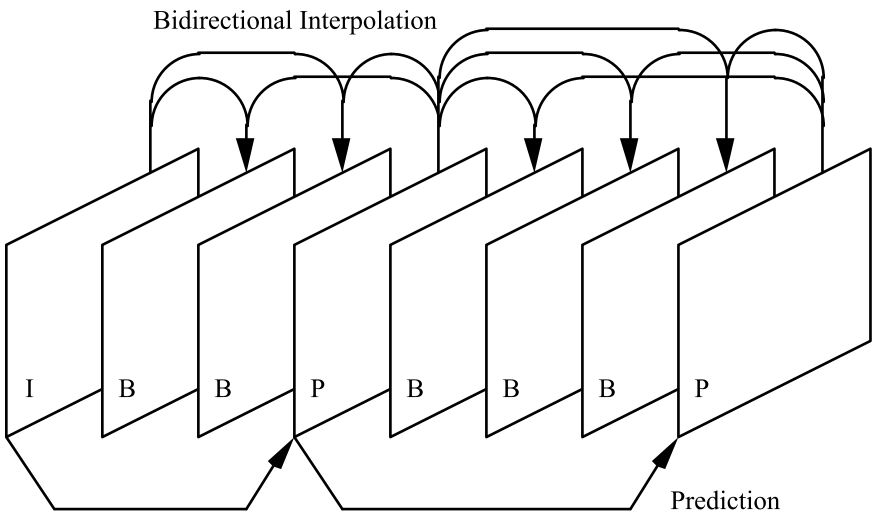


Figure I-1 Example of temporal picture structure

I.4.1.2 Coding interlaced video

Each frame of interlaced video consists of two fields which are separated by one field-period. The specification allows either the frame to be encoded as picture or the two fields to be encoded as two pictures. Frame encoding or field encoding can be adaptively selected on a frame-by-frame basis. Frame encoding is typically preferred when the video scene contains significant detail with limited motion. Field encoding, in which the second field can be predicted from the first, works better when there is fast movement.

I.4.1.3 Motion representation - macroblocks

As in ISO/IEC 11172-2, 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 represent it. Each macroblock can be temporally predicted in one of a number of different ways. For example, in frame encoding, the prediction from the previous reference frame can itself be either frame-based or field-based. Depending on the type of the macroblock, motion vector information and other side information is encoded with the compressed prediction error signal in each macroblock. The motion vectors are encoded differentially with respect to the last encoded motion vectors using variable length codes. The maximum length of the motion 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 specification does not specify how this should be done.

I.4.1.4 Spatial redundancy reduction

Both original pictures and prediction error signals have high spatial redundancy. This specification uses a block-based DCT method with visually weighted quantisation 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 quantised. After quantisation 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.4.1.5 Chrominance formats

In addition to the 4:2:0 format supported in ISO/IEC 11172-2 this specification supports 4:2:2 and 4:4:4 chrominance formats.

I.4.2 Scalable extensions

The scalability tools in this specification are designed to support applications beyond that supported by single layer video. Among the noteworthy applications areas addressed are video telecommunications, video on asynchronous transfer mode networks (ATM), interworking of video standards, video service hierarchies with multiple spatial, temporal and quality resolutions, HDTV with embedded TV, systems allowing migration to higher temporal resolution HDTV etc. Although a simple solution to scalable video is the simulcast technique which is based on transmission/storage of multiple independently coded reproductions of video, a more efficient alternative is scalable video coding, in which the bandwidth allocated to a given reproduction of video can be partially reutilised in coding of the next reproduction of video. In scalable video coding, it is assumed that given an encoded bitstream, decoders of various complexities can decode and display appropriate reproductions of coded video. A scalable video encoder is likely to have increased complexity when compared to a single layer encoder. However, this standard provides several different forms of scalabilities that address nonoverlapping applications with corresponding complexities. The basic scalability tools offered are: *data partitioning, SNR scalability*, *spatial scalability* and *temporal scalability.* Moreover, combinations of these basic scalability tools are also supported and are referred to as *hybrid scalability*. In the case of basic scalability, two layers of video referred to as the *lower layer* and the *enhancement layer* are allowed, whereas in hybrid scalability up to three layers are supported. The following Tables provide a few example applications of various scalabilities.

Table I-1 Applications of SNR scalability

Lower layer	Enhancement layer	Application
ITU-R-601	Same resolution and format as lower layer	Two quality service for Standard TV
High Definition	Same resolution and format as lower layer	Two quality service for HDTV
4:2:0 High Definition	4:2:2 chroma simulcast	Video production / distribution

Table I-2 Applications of spatial scalability

Base	Enhancement	Application
progressive(30Hz)	progressive(30Hz)	CIF/SCIF compatibility or scalability
interlace(30Hz)	interlace(30Hz)	HDTV/SDTV scalability
progressive(30Hz)	interlace(30Hz)	ISO/IEC 11172-2/compatibility with this specification
interlace(30Hz)	progressive(60Hz)	Migration to HR progressive HDTV

Table I-3. Applications of temporal scalability

Base	Enhancement	Higher	Application
progressive(30Hz)	progressive(30Hz)	progressive (60Hz)	Migration to HR progressive HDTV
interlace(30Hz)	interlace(30Hz)	progressive (60Hz)	Migration to HR progressive HDTV

I.4.2.1 Spatial scalable extension

Spatial scalability is a tool intended for use in video applications involving telecommunications, interworking of video standards, video database browsing, interworking of HDTV and TV etc., i.e., video systems with the primary common feature that a minimum of two layers of spatial resolution are necessary. Spatial scalability involves generating two spatial resolution video layers from a single video source such that the lower layer is coded by itself to provide the basic spatial resolution and the enhancement layer employs the spatially interpolated lower layer and carries the full spatial resolution of the input video source. The lower and the enhancement layers may either both use the coding tools in this specification, or the ISO/IEC 11172-2 standard for the lower layer and this specification for the enhancement layer. The latter case achieves a further advantage by facilitating interworking between video coding standards. Moreover, spatial scalability offers flexibility in choice of video formats to be employed in each layer. An additional advantage of spatial scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over channel with better error performance, while the less critical enhancement layer data can be sent over a channel with poor error performance.

I.4.2.2 SNR scalable extension

SNR scalability is a tool intended for use in video applications involving telecommunications, video services with multiple qualities, standard TV and HDTV, i.e., video systems with the primary common feature that a minimum of two layers of video quality are necessary. SNR scalability involves generating two video layers of same spatial resolution but different video qualities from a single video source such that the lower layer is coded by itself to provide the basic video quality and the enhancement layer is coded to enhance the lower layer. The enhancement layer when added back to the lower layer regenerates a higher quality reproduction of the input video. The lower and the enhancement layers may either use this specification or ISO/IEC 11172-2 standard for the lower layer and this specification for the enhancement layer. An additional advantage of SNR scalability is its ability to provide high degree of resilience to transmission errors as the more important data of the lower layer can be sent over channel with better error performance, while the less critical enhancement layer data can be sent over a channel with poor error performance.

I.4.2.3 Temporal scalable extension

Temporal scalability is a tool intended for use in a range of diverse video applications from telecommunications to HDTV for which migration to higher temporal resolution systems from that of lower temporal resolution systems may be necessary. In many cases, the lower temporal resolution video systems may be either the existing systems or the less expensive early generation systems, with the motivation of introducing more sophisticated systems gradually. Temporal scalability involves partitioning of video frames into layers, whereas the lower layer is coded by itself to provide the basic temporal rate and the enhancement layer is coded with temporal prediction with respect to the lower layer, these layers when decoded and temporal multiplexed to yield full temporal resolution of the video source. The lower temporal resolution systems may only decode the lower layer to provide basic temporal resolution, whereas more sophisticated systems of the future may decode both layers and provide high temporal resolution video while maintaining interworking with earlier generation systems. An additional advantage of temporal scalability is its ability to provide resilience to transmission errors as the more important data of the lower layer can be sent over channel with better error performance, while the less critical enhancement layer can be sent over a channel with poor error performance.

I.4.2.4 Data partitioning extension

Data partitioning is a tool intended for use when two channels are available for transmission and/or storage of a video bitstream, as may be the case in ATM networks, terrestrial broadcast, magnetic media, etc. The bitstream is partitioned between these channels such that more critical parts of the bitstream (such as headers, motion vectors, DC coefficients) are transmitted in the channel with the better error performance, and less critical data (such as higher DCT coefficients) is transmitted in the channel with poor error performance. Thus, degradation to channel errors are minimised since the critical parts of a bitstream are better protected. Data from neither channel may be decoded on a decoder that is not intended for decoding data partitioned bitstreams.

INTERNATIONAL STANDARD 13818-2

ITU-T RECOMMENDATION H.262

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

1 Scope

This Recommendation | International Standard specifies the coded representation of picture information for digital storage media and digital video communication and specifies the decoding process. The representation supports constant bitrate transmission, variable bitrate transmission, random access, channel hopping, scalable decoding, bitstream editing, as well as special functions such as fast forward playback, fast reverse playback, slow motion, pause and still pictures. This Recommendation | International Standard is forward compatible with ISO/IEC 11172-2 and upward or downward compatible with EDTV, HDTV, SDTV formats.

This Recommendation | International Standard is primarily applicable to digital storage media, video broadcast and communication. The storage media may be directly connected to the decoder, or via communications means such as busses, LANs, or telecommunications links.

2 Normative references

The following ITU-T Recommendations and International Standards contain provisions which through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | 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. The TSB (Telecommunication Standardisation Bureau) maintains a list of currently valid ITU-T Recommendations.

- 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"
- CCIR Volume X and XI Part 3 Recommendation 648: Recording of audio signals.
- CCIR Volume X and XI Part 3 Report 955-2: Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of advanced digital system II.
- ISO/IEC 11172 (1993) "Information technology — Coding of moving picture and associated audio for digital storage media at up to about 1,5 Mbit/s"
- IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990.
- IEC Publication 908:198, "CD Digital Audio System"
- IEC Standard Publication 461 Second edition, 1986 "Time and control code for video tape recorders"
- ITU-T Recommendation H.261 (Formerly CCITT Recommendation H.261) “Codec for audiovisual services at px64 kbit/s" Geneva, 1990
- ISO/IEC 10918-1 | ITU-T Rec. T.81 (JPEG) "Digital compression and coding of continuous-tone still images"

3 Definitions

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

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

3.2 **B-field picture**: A field structure B-Picture.

3.3 **B-frame picture**: A frame structure B-Picture.

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

3.5 **backward compatibility**: A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.

3.6 **backward motion vector**: A motion vector that is used for motion compensation from a reference frame or reference field at a later time in display order.

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

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

3.9 **block:** An 8-row by 8-column matrix of samples, or 64 DCT coefficients (source, quantised or dequantised).

3.10 **bottom field:** One of two fields that comprise a frame. Each line of a bottom field is spatially located immediately below the corresponding line of the top field.

3.11 **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.

3.12 **byte**: Sequence of 8-bits.

3.13 **channel**: A digital medium that stores or transports a bitstream constructed according to this specification.

3.14 **chrominance format:** Defines the number of chrominance blocks in a macroblock.

3.15 **chroma simulcast**: A type of scalability (which is a subset of SNR scalability) where the enhancement layer (s) contain only coded refinement data for the DC coefficients, and all the data for the AC coefficients, of the chrominance components.

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

3.17 **coded B-frame**: A B-frame picture or a pair of B-field pictures.

3.18 **coded frame**: A coded frame is a coded I-frame, a coded P-frame or a coded B-frame.

3.19 **coded I-frame**: An I-frame picture or a pair of field pictures, where the first one is an I-picture and the second one is an I-picture or a P-picture.

3.20 **coded P-frame**: A P-frame picture or a pair of P-field pictures.

3.21 **coded picture:** A coded picture is made of a picture header, the optionnal extensions immediately following it, and the following picture data. A coded picture may be a frame picture or a field picture.

3.22 **coded video bitstream**: A coded representation of a series of one or more pictures as defined in this specification.

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

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

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

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

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

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

3.29 **constant bitrate:** Operation where the bitrate is constant from start to finish of the coded bitstream.

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

3.31 **data partitioning**: A method for dividing a bitstream into two separate bitstreams for error resilience purposes. the two bitstreams have to be recombined before decoding.

3.32 **D-Picture**: A type of picture that shall not be used except in ISO/IEC 11172-2.

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

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

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

3.36 **decoder**: An embodiment of a decoding process.

3.37 **decoding (process):** The process defined in this specification that reads an input coded bitstream and produces decoded pictures or audio samples.

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

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

3.40 **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 Annex A of this specification.

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

3.42 **editing**: The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this specification.

3.43 **encoder**: An embodiment of an encoding process.

3.44 **encoding (process):** A process, not specified in this specification, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this specification.

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

3.46 **fast reverse playback**: The process of displaying the picture sequence in the reverse of display order faster than real-time.

3.47 **field:** For an interlaced video signal, a “field” is the assembly of alternate lines of a frame. Therefore an interlaced frame is composed of two fields, a top field and a bottom field.

3.48 **field period**: The reciprocal of twice the frame rate.

3.49 **field picture; field structure picture** : A field structure picture is a coded picture with picture_structure is equal to "Top field" or "Bottom field".

3.50 **flag**: A variable which can take one of only the two values defined in this specification.

3.51 **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.

3.52 **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.

3.53 **forward compatibility**: A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.

3.54 **forward motion vector**: A motion vector that is used for motion compensation from a reference frame or reference field at an earlier time in display order.

3.55 **frame:** A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and continuing through successive lines to the bottom of the frame. For interlaced video a frame consists of two fields, a top field and a bottom field. One of these fields will commence one field period later than the other.

3.56 **frame period**: The reciprocal of the frame rate.

3.57 **frame picture; frame structure picture** : A frame structure picture is a coded picture with picture_structure is equal to "Frame".

3.58 **frame rate**: The rate at which frames are be output from the decoding process.

3.59 **future reference frame (field)**: A future reference frame(field) is a reference frame(field) that occurs at a later time than the current picture in display order.

3.60 **header**: A block of data in the coded bitstream containing the coded representation of a number of data elements pertaining to the coded data that follow the header in the bitstream.

3.61 **hybrid scalability:** Hybrid scalability is the combination of two (or more) types of scalability.

3.62 **interlace:** The property of conventional television frames where alternating lines of the frame represent different instances in time. In an interlaced frame, one of the field is meant to be displayed first. This field is called the first field. The first field can be the top field or the bottom field of the frame.

3.63 **I-field picture**: A field structure I-Picture.

3.64 **I-frame picture**: A frame structure I-Picture.

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

3.66 **intra coding**: Coding of a macroblock or picture that uses information only from that macroblock or picture.

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

3.68 **level** : A defined set of constraints on the values which may be taken by the parameters of this specification within a particular profile. A profile may contain one or more levels.

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

3.70 **macroblock**: The four 8 by 8 blocks of luminance data and the two (for 4:2:0 chrominance format), four (for 4:2:2 chrominance format) or eight (for 4:4:4 chrominance format) 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 sample data and sometimes to the coded representation of the sample values and other data elements defined in the macroblock header of the syntax defined in this part of this specification. The usage is clear from the context.

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

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

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

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

3.75 **P-field picture**: A field structure P-Picture.

3.76 **P-frame picture**: A frame structure P-Picture.

3.77 **P-picture; predictive-coded picture** : A picture that is coded using motion compensated prediction from past reference fields or frame.

3.78 **parameter**: A variable within the syntax of this specification which may take one of a large range of values. A variable which can take one of only two values is a flag and not a parameter.

3.79 **past reference frame (field)**: A past reference frame(field) is a reference frame(field) that occurs at an earlier time than the current picture in display order.

3.80 **picture**: Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. For progressive video, a picture is identical to a frame, while for interlaced video, a picture can refer to a frame, or the top field or the bottom field of the frame depending on the context.

3.81 **prediction**: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.

3.82 **predictive-coded picture; P-picture**: A picture that is coded using motion compensated prediction from past reference frames or reference fields.

3.83 **prediction error**: The difference between the actual value of a sample or data element and its predictor.

3.84 **predictor**: A linear combination of previously decoded sample values or data elements.

3.85 **profile**: A defined subset of the syntax of this specification.

Note In this specification the word “profile” is used as defined above. It should not be confused with other definitions of “profile” and in particular it does not have the meaning that is defined by JTC1/SGFS.

3.86 **progressive:** The property of film frames where all the samples of the frame represent the same instances in time.

3.87 **quantisation matrix**: A set of sixty-four 8-bit values used by the dequantiser.

3.88 **quantised DCT coefficients**: DCT coefficients before dequantisation. A variable length coded representation of quantised DCT coefficients is transmitted as part of the compressed video bitstream.

3.89 **quantiser scale**: A scale factor coded in the bitstream and used by the decoding process to scale the dequantisation.

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

3.91 **reconstructed frame:** A reconstructed frame consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. A reconstructed frame is obtained by decoding a coded frame.

3.92 **reconstructed picture:** A reconstructed picture is obtained by decoding a coded picture. A reconstructed picture is either a reconstructed frame (when decoding a frame picture), or one field of a reconstructed frame (when decoding a field picture). If the coded picture is a field picture, then the reconstructed picture is the top field or the bottom field of the reconstructed frame.

3.93 **reference field:** A reference field is one field of a reconstructed frame. Reference fields are used for forward and backward prediction when P-pictures and B-pictures are decoded. Note that when field P-pictures are decoded, prediction of the second field P-picture of a coded frame uses the first reconstructed field of the same coded frame as a reference field.

3.94 **reference frame:** A reference frame is a reconstructed frame that was coded in the form of a coded I-frame or a coded P-frame. Reference frames are used for forward and backward prediction when P-pictures and B-pictures are decoded.

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

3.96 **sample aspect ratio:** (abbreviated to **SAR**). This specifies the distance between samples. It is defined (for the purposes of this specification) as the vertical displacement of the lines of luminance samples in a frame divided by the horizontal displacement of the luminance samples. Thus its units are (metres per line) ÷ (metres per sample)

3.97 **scalability:** Scalability is the ability of a decoder to decode an ordered set of bitstreams to produce a reconstructed sequence. Moreover, useful video is output when subsets are decoded. The minimum subset that can thus be decoded is the first bitstream in the set which is called the base layer. Each of the other bitstreams in the set is called an enhancement layer. When addressing a specific enhancement layer, "lower layer" refer to the bitstream which precedes the enhancement layer.

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

3.99 **skipped macroblock**: A macroblock for which no data is encoded.

3.100 **slice**: A series of macroblocks.

3.101 **SNR scalability**: A type of scalability where the enhancement layer (s) contain only coded refinement data for the DCT coefficients of the lower layer.

3.102 **spatial scalability**: A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer without using motion vectors. The layers can have different frame sizes, frame rates or chrominance formats

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

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

3.105 **temporal scalability**: A type of scalability where an enhancement layer also uses predictions from sample data derived from a lower layer using motion vectors. The layers have identical frame size, and chrominance formats, but can have different frame rates.

3.106 **top field:** One of two fields that comprise a frame. Each line of a top field is spatially located immediately above the corresponding line of the bottom field.

3.107 **variable bitrate:** Operation where the bitrate varies with time during the decoding of a coded bitstream.

3.108 **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.

3.109 **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.

3.110 **video sequence**: The highest syntactic structure of coded video bitstreams. It contains a series of one or more coded frames.

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

4 Abbreviations and symbols

The mathematical operators used to describe this specification are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin from zero.

4.1 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 minus infinity. For example 3 DIV 2 is rounded to 1, and -3 DIV 2 is rounded to -2.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
%	Modulus operator. Defined only for positive numbers.
Sign( )	$\mathrm{Sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & x == 0 \\ -1 & x < 0 \end{cases}$
Abs( )	$\mathrm{Abs}(x) = \begin{cases} x & x >= 0 \\ -x & x < 0 \end{cases}$

4.2 Logical operators

||	Logical OR.
&&	Logical AND.
!	Logical NOT.

4.3 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.

4.4 Bitwise operators

& AND

| OR

>> Shift right with sign extension.

<< Shift left with zero fill.

4.5 Assignment

= Assignment operator.

4.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in the specification. 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.

simsbf Signed integer, in twos complement format, most significant (sign) bit first.

vlclbf Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written. The byte order of multibyte words is most significant byte first.

4.7 Constants

π 3,141 592 653 59...

e 2,718 281 828 45...

5 Conventions

5.1 Method of describing bitstream syntax

The bitstream retrieved by the decoder is described in 6.2. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bitstream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in 6.3. 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 in the data stream. This repeats until the condition is not true.
data_element	
. . .	
}	
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 next in the data stream.
data_element	
. . .	
} else {	If the condition is not true, then the second group of data elements occurs next in the data stream.
data_element	
. . .	
}	
for (i = 0; i < n; i++) {	The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.
data_element	
. . .	
}	
/* comment ... */	Explanatory comment that may be deleted entirely without in any way altering the syntax.

This syntax uses the 'C-code' convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true. In many cases a literal string is used in a condition. For example;

if (scalable_mode == "spatial scalability") ...

In such cases the literal string is that used to describe the value of the bitstream element in 6.3. In this example, we see that "spatial scalability" is defined in Table 6-10 to be represented by the two bit binary number '01'.

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

While the syntax is expressed in procedural terms, it should not be assumed that 6.2 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include 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 standardised.

5.2 Definition of functions

Several utility functions for picture coding algorithm are defined as follows:

5.2.1 Definition of bytealigned() function

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

5.2.2 Definition of nextbits() function

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

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

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

5.3 Reserved, forbidden and marker_bit

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

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

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

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

5.4 Arithmetic precision

In order to reduce discrepancies between implementations of this specification, the following rules for arithmetic operations are specified.

(a) Where arithmetic precision is not specified, such as in the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values

(b) Where ranges of values are given by two dots, the end points are included if a bracket is present, and excluded if the 'less then' (<) and 'greater then' (>) characters are used. For example, [a .. b> means from a to b, including a but excluding b.

6 Video bitstream syntax and semantics

6.1 Structure of coded video data

Coded video data consists of an ordered set of video bitstreams, called layers. If there is only one layer, the coded video data is called non-scalable video bitstream. If there are two layers or more, the coded video data is called a scalable hierarchy.

The first layer (of the ordered set) is called base layer, and it can always be decoded independently. See 7.1 to 7.6 of this specification for a description of the decoding process for the base layer, except in the case of Data partitionning, described in 7.10.

Other layers are called enhancement layers, and can only be decoded together with all the lower layers (previous layers in the ordered set), starting with the base layer. See 7.7 to 7.11 of this specification for a description of the decoding process for scalable hierarchy.

See ITU-T Rec. xxx | ISO/IEC 13818-1 for a description of the way layers may be multiplexed together.

The base layer of a scalable hierarchy may conform to this specification or to other standards such as ISO/IEC 11172-2. See details in 7.7 to 7.11. Enhancement layers shall conform to this specification.

In all cases apart from Data partitioning, the base layer does not contain a sequence_scalable_extension(). Enhancement layers always contain sequence_scalable_extension().

In general the video bitstream can be thought of as a syntactic hierarchy in which syntactic structures contain one or more subordinate structures. For instance the structure “picture_data()” contains one or more of the syntactic structure “slice()” which in turn contains one or more of the structure “macroblock()”.

This structure is very similar to that used in ISO/IEC 11172-2.

6.1.1 Video sequence

The highest syntactic structure of the coded video bitstream is the video sequence.

A video sequence commences with a sequence header which may optionally be followed by a group of pictures header and then by one or more coded frames. The order of the coded frames in the coded bitstream is the order in which the decoder processes them, but not necessarily in the correct order for display. The video sequence is terminated by a sequence_end_code. At various points in the video sequence a particular coded frame may be preceded by either a repeat sequence header or a group of pictures header or both. (In the case that both a repeat sequence header and a group of pictures header immediately precede a particular picture the group of pictures header shall follow the repeat sequence header.)

6.1.2.5 Progressive and interlaced sequences

This specification deals with coding of both progressive and interlaced sequences.

The output of the decoding process, for interlaced sequences, consists of a series of reconstructed fields that are separated in time by a field period. The two fields of a frame may be coded separately (field-pictures). Alternatively the two fields may be coded together as a frame (frame-pictures). Both frame pictures and field pictures may be used in a single video sequence.

In progressive sequences each picture in the sequence shall be a frame picture. The sequence, at the output of the decoding process, consists of a series of reconstructed frames that are separated in time by a frame period.

6.1.2.4 Frame

A frame consists of three rectangular matrices of integers; a luminance matrix (Y), and two chrominance matrices (Cb and Cr).

The relationship between these Y, Cb and Cr components and the primary (analogue) Red, Green and Blue Signals (E'_R, E'_G and E'_B), the chromaticity of these primaries and the transfer characteristics of the source frame may be specified in the bitstream (or specified by some other means). This information does not affect the decoding process.

6.1.2.4 Field

A field consists of every other line of samples in the three rectangular matrices of integers representing a frame.

A frame is the union of a top field and a bottom field. The top field is the field that contains the top-most line of each of the three matrices. The bottom field is the other one.

6.1.2.4 Picture

A reconstructed picture is obtained by decoding a coded picture, i.e. a picture header, the optionnal extensions immediately following it, and the picture data. A coded picture may be a frame picture or a field picture. A reconstructed picture is either a reconstructed frame (when decoding a frame picture), or one field of a reconstructed frame (when decoding a field picture).

6.1.2.5.1 Field pictures

If field pictures are used then they shall occur in pairs (one top field and one bottom field) and together constitute a coded frame. The two field pictures that comprise a coded frame shall be encoded in the bitstream in the order in which they shall occur at the output of the decoding process.

When the first picture of the coded frame is a P-field picture, then the second picture of the coded frame shall also be a P- field picture. Similarly when the first picture of the coded frame is a B-field picture the second picture of the coded frame shall also be a B-field picture.

When the first picture of the coded frame is a I-field picture, then the second picture of the frame shall be either an I-field picture or a P-field picture.

6.1.2.5.2 Frame pictures

When coding interlaced sequences using frame pictures, the two fields of the frame shall be interleaved with one another and then the entire frame is coded as a single frame-picture.

6.1.2.4 Picture types

There are three 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.

6.1.1.2 Sequence header

A video sequence header commences with a sequence_header_code and is followed by a series of data elements. In this specification sequence_header() shall be followed by sequence_extension() which includes further parameters beyond those used by ISO/IEC 11172-2. When sequence_extension() is present, the syntax and semantics defined in ISO/IEC 11172-2 does not apply, and the present specification applies.

In repeated sequence headers all of the data elements with the permitted exception of those defining the quantisation matrices (load_intra_quantiser_matrix, load_non_intra_quantiser_matrix and optionally intra_quantiser_matrix and non_intra_quantiser_matrix) shall have the same values as in the first sequence header. The quantisation matrices may be redefined each time that a sequence header occurs in the bitstream (Note that quantisation matrices may also be updated using quant_matrix_extension()).

All of the data elements in the sequence_extension() that follows a repeat sequence_header() shall have the same values as in the first sequence_extension().

If a sequence_scalable_extension() occurs after the first sequence_header() all subsequent sequence headers shall be followed by sequence_scalable_extension() in which all data elements are the same as in the first sequence_scalable_extension(). Conversely if no sequence_scalable_extension() occurs between the first sequence_header() and the first picture_header() then sequence_scalable_extension() shall not occur in the bitstream.

If a sequence_display_extension() occurs after the first sequence_header() all subsequent sequence headers shall be followed by sequence_display_extension() in which all data elements are the same as in the first sequence_display_extension(). Conversely if no sequence_display_extension() occurs between the first sequence_header() and the first picture_header() then sequence_display_extension() shall not occur in the bitstream.

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 the coded bitstream, a repeat sequence header may precede either an I-picture or a P-picture but not a B-picture. In the case that an interlaced frame is coded as two separate field pictures a repeat sequence header shall not precede the second of these two field pictures.

If a bitstream is edited so that all of the data preceding any of the repeated sequence headers is removed (or alternatively random access is made to that sequence header) then the resulting bitstream shall be a legal bitstream that complies with this specification. In the case that the first picture of the resulting bitstream is a P-picture, it is possible that it will contain non-intra macroblocks. Since the reference picture(s) required by the decoding process are not available, the reconstructed picture may not be fully defined. The time taken to fully refresh the entire frame depends on the refresh techniques employed.

6.1.1.3 I-pictures and group of pictures header

I-pictures are intended to assist random access into the sequence. Applications requiring random access, fast-forward playback, or fast reverse playback may use I-pictures relatively frequently.

I-pictures may also be used at scene cuts or other cases where motion compensation is ineffective.

Group of picture header is an optional header that can be used immediately before a coded I-frame to indicate to the decoder if the first consecutive B-pictures immediately following the coded I-frame can be reconstructed properly in the case of a random access. In effect, if the preceding reference frame is not available, those B-pictures, if any, cannot be reconstructed properly unless they only use backward prediction. This is more precisely defined in the section describing closed_gop and broken_link. A group of picture header also contains a time code information that is not used by the decoding process.

In the coded bitstream, the first coded frame following a group of pictures header shall be a coded I-frame.

6.1.2.1 4:2:0 Format

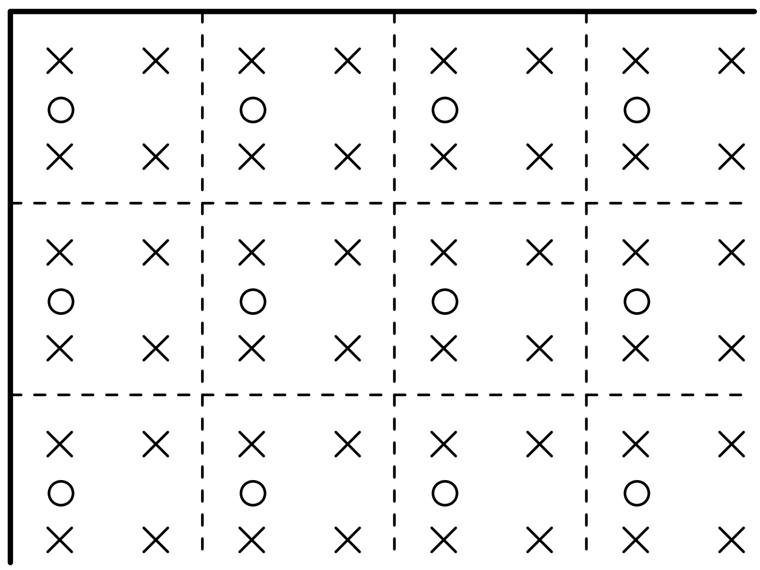
In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

Note When interlaced frames are coded as field pictures, the picture reconstructed from each of these field pictures shall have a Y-matrix with half the number of lines as the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by four.

The luminance and chrominance samples are positioned as shown in Figure 6-1.

In order to further specify the organisation, Figures 6-2a and 6-2b show the (vertical) positioning of the samples in an interlaced frame. Figures 6-3 shows the (vertical) positioning of the samples in an progressive frame.

In each field of an interlaced frame, the chrominance samples do not lie (vertically) mid way between the luminance samples of the field, this is so that the spatial location of the chrominance samples in the frame is the same whether the frame is represented as a single frame-picture or two field-pictures.



× Represent luminance samples

○ Represent chrominance samples

Figure 6-1 -- The position of luminance and chrominance samples. 4:2:0 data.

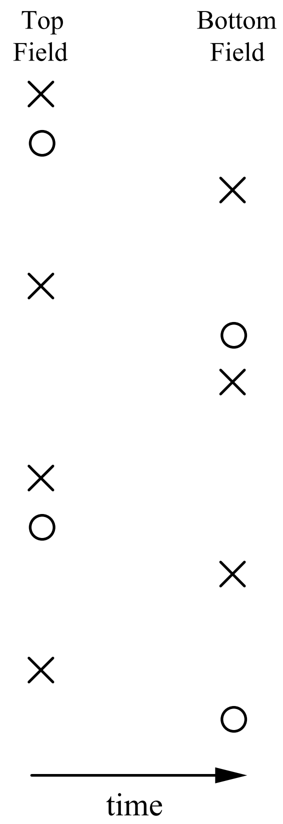


Figure 6-2a – Vertical and temporal positions of samples in an interlaced frame with top_field_first = 1.

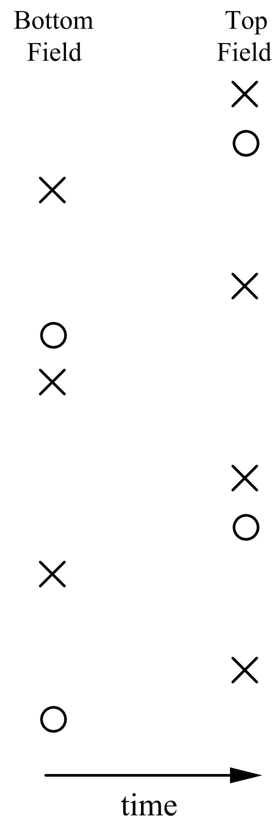


Figure 6-2b – Vertical and temporal positions of samples in an interlaced frame with top_field_first = 0.

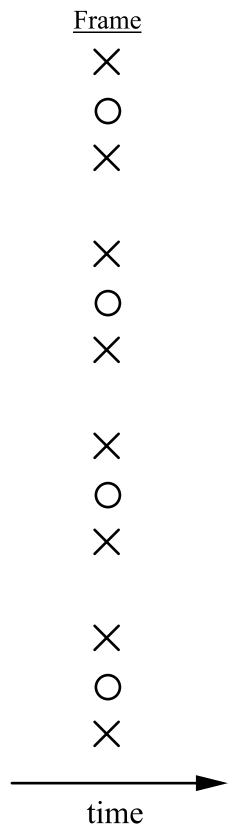


Figure 6-3 – Vertical and temporal positions of samples in a progressive frame.

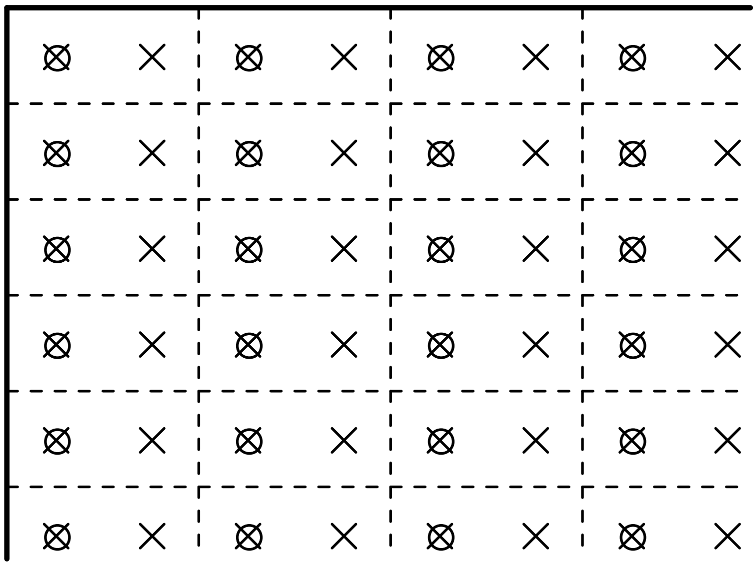
6.1.2.2 4:2:2 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in the horizontal dimension and the same size as the Y-matrix in the vertical dimension. The Y-matrix shall have an even number of samples.

Note When interlaced frames are coded as field pictures, the picture reconstructed from each of these field pictures shall have a Y-matrix with half the number of lines as the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by two.

The luminance and chrominance samples are positioned as shown in Figure 6-3.

In order to clarify the organisation, Figure 6-4 shows the (vertical) positioning of the samples when the frame is separated into two fields.



× Represent luminance samples

○ Represent chrominance samples

Figure 6-3 -- The position of luminance and chrominance samples. 4:2:2 data.

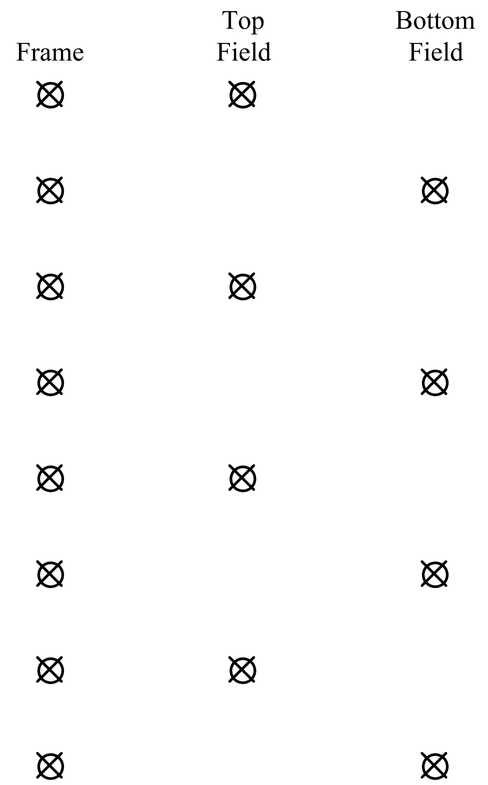


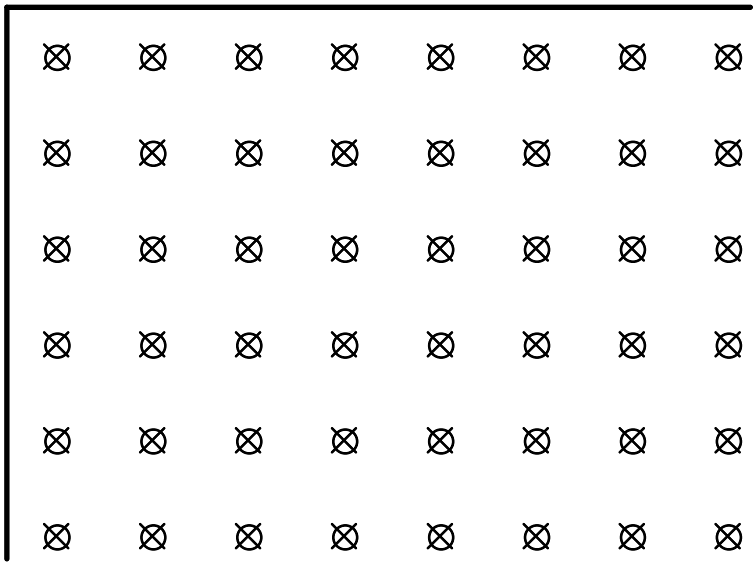
Figure 6-4 – Vertical positions of samples with 4:2:2 and 4:4:4 data

6.1.2.3 4:4:4 Format

In this format the Cb and Cr matrices shall be the same size as the Y-matrix in the horizontal and the vertical dimensions.

Note When interlaced frames are coded as field pictures, the picture reconstructed from each of these field pictures shall have a Y-matrix with half the number of lines as the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by two.

The luminance and chrominance samples are positioned as shown in Figures 6-4 and 6-5.



× Represent luminance samples

○ Represent chrominance samples

Figure 6-5 -- The position of luminance and chrominance samples. 4:4:4 data.

6.1.1.1 Frame reordering

The order of the coded frames in the coded bitstream is the order in which the decoder processes them. When B-pictures are used in a sequence, the order of the coded frames in the bitstream is sometimes different from the order at which the reconstructed frames or fields are output by the decoding process.

The following is an example of pictures taken from the beginning of a video sequence. In this example there are two coded B-frames between successive coded P-frames and also two coded B-frames between successive coded I- and P-frames and all pictures are frame-pictures. Frame '1I' is used to form a prediction for frame '4P'. Frames '4P' and '1I' are both used to form predictions for frames '2B' and '3B'. Therefore the order of coded frames in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder shall display them in the order '1I', '2B', '3B', '4P'.

At the encoder input,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

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

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

At the decoder output,

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

The number of consecutive coded B-frames is variable. Coded B-frames may not be present between successive coded P-frames (or between coded I-frame and coded P-frames). Within each group of consecutive coded B-frames the frames shall occur in the bitstream in the order in which they shall appear at the decoder output.

A sequence may also contain no coded I-frame in which case some care is required at the start of the sequence and within the sequence to effect both random access and error recovery.

A sequence may contain no coded P-frame.

A sequence shall not be composed of only coded B-frames.

6.1.3 Slice

A **slice** is a series of an arbitrary number of macroblocks. The first and last macroblocks of a slice shall not be skipped macroblocks. Every slice shall contain at least one macroblock. Slices shall not overlap. The position of slices may change from picture to picture.

The first and last macroblock of a slice shall be in the same horizontal row of macroblocks.

Slices shall occur in the bitstream in the order in which they are encountered, starting at the upper-left of the picture and proceeding by raster-scan order from left to right and top to bottom (illustrated in the Figures of this clause as alphabetical order).

6.1.3.1 The general slice structure

In the most general case it is not necessary for the slices to cover the entire picture. Figure 6-6 shows this case. Those areas that are not enclosed in a slice are not encoded and no information is encoded for such areas (in the specific picture).

If the slices do not cover the entire picture then it is a requirement that if the picture is subsequently used to form predictions then predictions shall only be made from those regions of the picture that were enclosed in slices. It is the responsibility of the encoder to ensure this.

This specification does not define what action a decoder shall take in the regions between the slices.

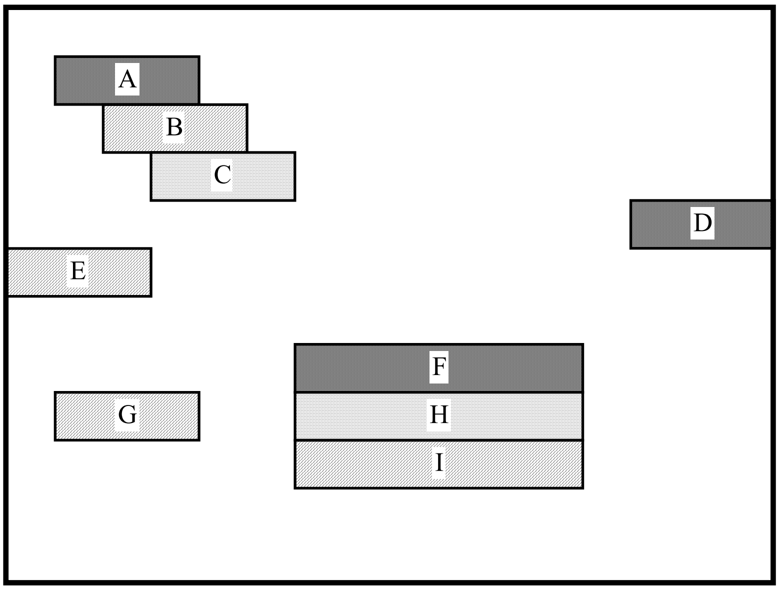


Figure 6-6. The most general slice structure.

6.1.3.2 Restricted slice structure

In certain defined levels of defined profiles a restricted slice structure illustrated in Figure 6-7 shall be used. In this case every macroblock in the picture shall be enclosed in a slice.

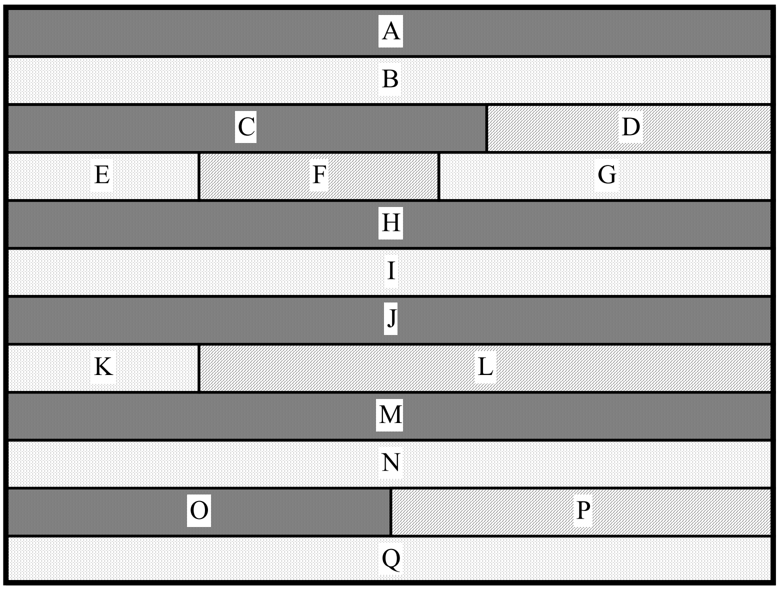


Figure 6-7. Restricted slice structure.

Where a defined level of a defined profile requires that the slice structure obeys the restrictions detailed in this clause, the term “restricted slice structure” may be used.

6.1.4 Macroblock

A **macroblock** contains a section of the luminance component and the spatially corresponding chrominance components. The term macroblock can either refer to source and decoded data or to the corresponding coded data elements. A skipped macroblock is one for which no information is transmitted (see 7.6.6). There are three chrominance formats for a macroblock, namely, 4:2:0, 4:2:2 and 4:4:4 formats. The orders of blocks in a macroblock shall be different for each different chrominance format and are illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in Figure 6-8.

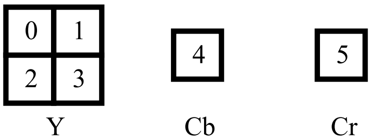


Figure 6-8 4:2:0 Macroblock structure

A 4:2:2 Macroblock consists of 8 blocks. This structure holds 4 Y, 2 Cb and 2 Cr Blocks and the block order is depicted in Figure 6-9.

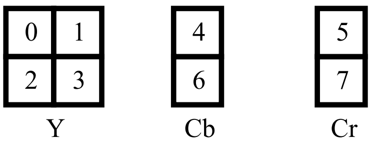


Figure 6-9 4:2:2 Macroblock structure

A 4:4:4 Macroblock consists of 12 blocks. This structure holds 4 Y, 4 Cb and 4 Cr Blocks and the block order is depicted in Figure 6-10.

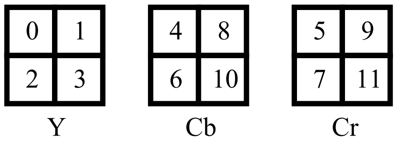


Figure 6-10 4:4:4 Macroblock structure

In frame pictures, where both frame and field DCT coding may be used, the internal organisation within the macroblock is different in each case.

- In the case of frame DCT coding, each block shall be composed of lines from the two fields alternately. This is illustrated in Figure 6-12.
- In the case of field DCT coding, each block shall be composed of lines from only one of the two fields. This is illustrated in Figure 6-13.

In the case of chrominance blocks the structure depends upon the chrominance format that is being used. In the case of 4:2:2 and 4:4:4 formats (where there are two blocks in the vertical dimension of the macroblock) the chrominance blocks are treated in exactly the same manner as the luminance blocks. However, In the 4:2:0 format the chrominance blocks shall always be organised in frame structure for the purposes of DCT coding. It should however be noted that field based predictions may be made for these blocks which will, in the general case, require that predictions for 8x4 regions (after half-sample filtering) must be made.

In the case of a progressive frame, frame DCT coding shall always be used as illustrated in Figure 6-12.

In field pictures, each picture only contains lines from one of the fields. In this case each block consists of lines taken from successive lines in the picture as illustrated by Figure 6-12.

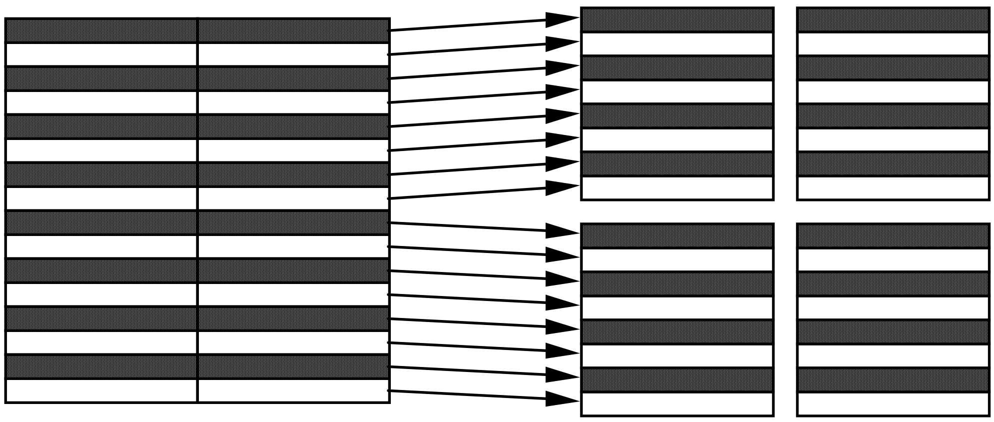


Figure 6-12 Luminance macroblock structure in frame DCT coding

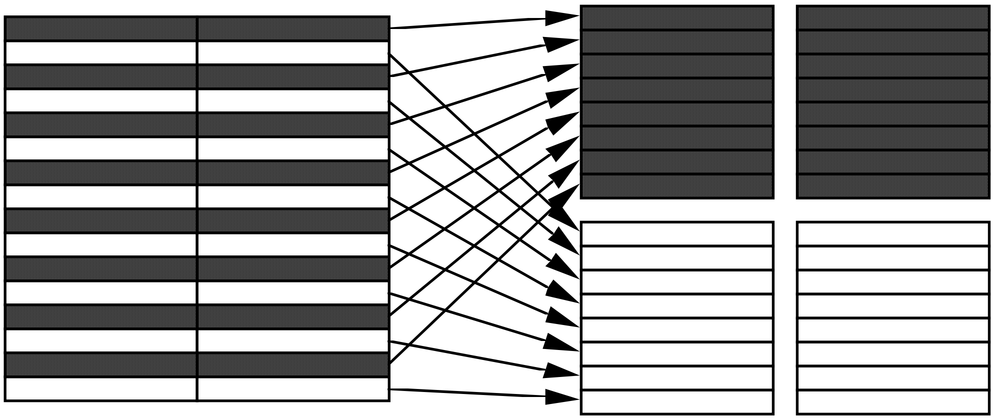


Figure 6-13 Luminance macroblock structure in field DCT coding

6.1.5 Block

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

When "block" refers to source and reconstructed data it refers to an orthogonal section of a luminance or chrominance component with the same number of lines and samples. There are 8 lines and 8 samples in the block.

6.2 Video bitstream syntax

6.2.1 Start codes

Start codes are specific bit patterns that do not otherwise occur in the video stream.

Each start code consists of a start code prefix followed by a start code value. The start code prefix is a string of twenty three bits with the value zero followed by a single bit with the value one. The start code prefix is thus the bit string "0000 0000 0000 0000 0000 0001".

The start code value is an eight bit integer which identifies the type of start code. Most types of start code have just one start code value. However slice_start_code is represented by many start code values, in this case the start code value is the slice_vertical_position for the slice.

All start codes shall be byte aligned. This shall be achieved by inserting bits with the value zero before the start code prefix such that the first bit of the start code prefix is the most significant bit of a byte.

Table 6-1 defines the slice code values for the start codes used in the video bitstream.

Table 6-1 — Start code values

name	**start code value (hexadecimal)**
picture_start_code	00
slice_start_code	01 through AF
reserved	B0
reserved	B1
user_data_start_code	B2
sequence_header_code	B3
sequence_error_code	B4
extension_start_code	B5
reserved	B6
sequence_end_code	B7
group_start_code	B8
system start codes (see note)	B9 through FF
NOTE - system start codes are defined in Part 1 of this specification	

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 a media interface to indicate where uncorrectable errors have been detected.

1 **6.2.2 Video Sequence**

video_sequence() {	**No. of bits**	**Mnemonic**
next_start_code()		
sequence_header()		
if (nextbits() == extension_start_code) {		
sequence_extension()		
do {		
extension_and_user_data(0)		
do {		
if (nextbits() == group_start_code) {		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
extensions_and_user_data(2)		
picture_data()		
} while ((nextbits() == picture_start_code) ||		
(nextbits() == group_start_code))		
if (nextbits() != sequence_end_code) {		
sequence_header()		
sequence_extension()		
}		
} while (nextbits() != sequence_end_code)		
} else {		
/* ISO/IEC 11172-2 */		
}		
sequence_end_code	32	bslbf
}		

2

1 **6.2.2.1 Sequence header**

sequence_header() {	**No. of bits**	**Mnemonic**
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	
load_intra_quantiser_matrix	1	
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8*64	uimsbf
load_non_intra_quantiser_matrix	1	
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8*64	uimsbf
next_start_code()		
}		

2

3 **6.2.2.2 Extension and user data**

extension_and_user_data(i) {	**No. of bits**	**Mnemonic**
while ((nextbits()== extension_start_code) ||		
(nextbits()== user_data_start_code)) {		
if (i != 1)		
if (nextbits()== extension_start_code)		
extension_data(i)		
if (nextbits()== user_data_start_code)		
user_data()		
}		
}		

4

1 **6.2.2.2.1 Extension data**

extension_data(i) {	**No. of bits**	**Mnemonic**
while (nextbits()== extension_start_code) {		
extension_start_code	32	bslbf
if (i == 0) { /* follows sequence_extension() */		
if (nextbits()== “Sequence Display Extension ID”)		
sequence_display_extension()		
if (nextbits()== “Sequence Scalable Extension ID”)		
sequence_scalable_extension()		
}		
/* Note: extension never follows a group_of_pictures_header() */		
if (i == 2) { /* follows picture_coding_extension() */		
if (nextbits()== “Quant Matrix Extension ID”)		
quant_matrix_extension()		
if (nextbits()== “Picture Pan Scan Extension ID”)		
picture_display_extension()		
if (nextbits()== “Picture Spatial Scalable Extension ID”)		
picture_spatial_scalable_extension()		
if (nextbits()== “Picture Temporal Scalable Ext. ID”)		
picture_temporal_scalable_extension()		
}		
}		
}		

2

3 **6.2.2.2.2 User data**

user_data() {	**No. of bits**	**Mnemonic**
user_data_start_code	32	bslbf
while(nextbits() != ‘0000 0000 0000 0000 0000 0001’) {		
user_data	8	
}		
next_start_code()		
}		

4

6.2.2.3 Sequence extension

sequence_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indication	8	uimsbf
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_extension	8	uimsbf
low_delay	1	uimsbf
frame_rate_extension_n	2	uimsbf
frame_rate_extension_d	5	uimsbf
next_start_code()		
}		

6.2.2.4 Sequence display extension

sequence_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
video_format	3	uimsbf
colour_description	1	uimsbf
if (colour_description) {		
colour_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coefficients	8	uimsbf
}		
display_horizontal_size	14	uimsbf
marker_bit	1	bslbf
display_vertical_size	14	uimsbf
next_start_code()		
}		

6.2.2.5 Sequence scalable extension

sequence_scalable_extension() {	**No. of bits**	**Mnemonic**
extension_start_code_identifier	4	uimsbf
scalable_mode	2	uimsbf
layer_id	4	uimsbf
if (scalable_mode == “spatial scalability”) {		
lower_layer_prediction_horizontal_size	14	uimsbf
marker_bit	1	bslbf
lower_layer_prediction_vertical_size	14	uimsbf
horizontal_subsampling_factor_m	5	uimsbf
horizontal_subsampling_factor_n	5	uimsbf
vertical_subsampling_factor_m	5	uimsbf
vertical_subsampling_factor_n	5	uimsbf
}		
if (scalable_mode == “temporal scalability”) {		
picture_mux_enable	1	uimsbf
if (picture_mux_enable)		
mux_to_progressive_sequence	1	uimsbf
picture_mux_order	3	uimsbf
picture_mux_factor	3	uimsbf
}		
next_start_code()		
}		

6.2.2.6 Group of pictures header

group_of_pictures_header() {	**No. of bits**	**Mnemonic**
group_start_code	32	bslbf
time_code	25	bslbf
closed_gop	1	uimsbf
broken_link	1	uimsbf
next_start_code()		
}		

6.2.3 Picture header

picture_header() {	**No. of bits**	**Mnemonic**
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 /* with the value “1” */	1	uimsbf
extra_information_picture	8	
}		
extra_bit_picture /* with the value “0” */	1	uimsbf
next_start_code()		
}		

1 **6.2.3.1 Picture coding extension**

picture_coding_extension() {	**No . of bits**	**Mnemonic**
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code[0][0] /* forward horizontal */	4	uimsbf
f_code[0][1] /* forward vertical */	4	uimsbf
f_code[1][0] /* backward horizontal */	4	uimsbf
f_code[1][1] /* backward vertical */	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top_field_first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment_motion_vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if (composite_display_flag) {		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

2

6.2.3.2 Quant matrix extension

quant_matrix_extension() {	**No. of bits**	**Mnemonic**
extension_start_code_identifier	4	uimsbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_intra_quantiser_matrix	1	uimsbf
if (load_chroma_intra_quantiser_matrix)		
chroma_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_non_intra_quantiser_matrix	1	uimsbf
if (load_chroma_non_intra_quantiser_matrix)		
chroma_non_intra_quantiser_matrix[64]	8 * 64	uimsbf
next_start_code()		
}		

6.2.3.3 Picture display extension

picture_display_extension() {	**No. of bits**	**Mnemonic**
extension_start_code_identifier	4	uimsbf
for (i=0; i<number_of_frame_centre_offsets; i++) {		
frame_centre_horizontal_offset	16	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset	16	simsbf
marker_bit	1	bslbf
}		
next_start_code()		
}		

6.2.3.4 Picture temporal scalable extension

picture_temporal_scalable_extension() {	**No. of bits**	**Mnemonic**
extension_start_code_identifier	4	uimsbf
reference_select_code	2	uimsbf
forward_temporal_reference	10	uimsbf
marker_bit	1	bslbf
backward_temporal_reference	10	uimsbf
next_start_code()		
}		

6.2.3.5 Picture spatial scalable extension

picture_spatial_scalable_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
lower_layer_temporal_reference	10	uimsbf
marker_bit	1	bslbf
lower_layer_horizontal_offset	15	simsbf
marker_bit	1	bslbf
lower_layer_vertical_offset	15	simsbf
spatial_temporal_weight_code_table_index	2	uimsbf
lower_layer_progressive_frame	1	uimsbf
lower_layer_deinterlaced_field_select	1	uimsbf
next_start_code()		
}		

6.2.3.6 Picture data

picture_data() {	No. of bits	Mnemonic
do {		
slice()		
} while (nextbits() == slice_start_code)		
next_start_code()		
}		

1 **6.2.4 Slice**

slice() {	**No. of bits**	**Mnemonic**
slice_start_code	32	bslbf
if (vertical_size > 2800)		
slice_vertical_position_extension	3	uimsbf
if (*<sequence_scalable_extension() is present in the bitstream>*)		
if (scalable_mode == “data partitioning”)		
priority_breakpoint	7	uimsbf
quantiser_scale_code	5	uimsbf
if (nextbits() == '1') {		
intra_slice_flag	1	bslbf
intra_slice	1	uimsbf
reserved_bits	7	uimsbf
while (nextbits() == '1') {		
extra_bit_slice /* with the value “1” */	1	uimsbf
extra_information_slice	8	
}		
}		
extra_bit_slice /* with the value “0” */	1	uimsbf
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

2

6.2.5 Macroblock

macroblock() {	No. of bits	Mnemonic
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	bslbf
macroblock_address_increment	1-11	vlclbf
macroblock_modes()		
if (macroblock_quant)		
quantiser_scale_code	5	uimsbf
if (macroblock_motion_forward ||		
(macroblock_intra && concealment_motion_vectors))		
motion_vectors(0)		
if (macroblock_motion_backward)		
motion_vectors(1)		
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	bslbf
if (macroblock_pattern)		
coded_block_pattern()		
for (i=0; i<block_count; i++) {		
block(i)		
}		
}		

6.2.5.1 Macroblock modes

macroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlclbf
if ((spatial_temporal_weight_code_flag == 1) &&		
(spatial_temporal_weight_code_table_index != ‘00’)) {		
spatial_temporal_weight_code	2	uimsbf
}		
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == ‘frame’) {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if (decode_dct_type) {		
dct_type	1	uimsbf
}		
}		

6.2.5.2 Motion vectors

motion_vectors (s) {	**No. of bits**	**Mnemonic**
if (motion_vector_count == 1) {		
if ((mv_format == field) && (dmv != 1))		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
} else {		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
motion_vertical_field_select[1][s]	1	uimsbf
motion_vector(1, s)		
}		
}		

6.2.5.2.1 Motion vector

motion_vector (r, s) {	**No. of bits**	**Mnemonic**
motion_code[r][s][0]	1-11	vlclbf
if ((f_code[s][0] != 1) && (motion_code[r][s][0] != 0))		
motion_residual[r][s][0]	1-8	uimsbf
if (dmv == 1)		
dmvector[0]	1-2	vlclbf
motion_code[r][s][1]	1-11	vlclbf
if ((f_code[s][0] != 1) && (motion_code[r][s][1] != 0))		
motion_residual[r][s][1]	1-8	uimsbf
if (dmv == 1)		
dmvector[1]	1-2	vlclbf
}		

6.2.5.3 Coded block pattern

coded_block_pattern () {	**No. of bits**	**Mnemonic**
coded_block_pattern_420	3-9	vlclbf
if (chroma_format == 4:2:2)		
coded_block_pattern_1	2	uimsbf
if (chroma_format == 4:4:4)		
coded_block_pattern_2	6	uimsbf
}		

6.2.6 Block

The detailed syntax for the terms “First DCT coefficient”, “Subsequent DCT coefficient” and “End of Block” is fully described in 7.2.

Note This clause does not adequately document the block layer syntax when data partitioning is used. See 7.10.

block(i) {	**No. of bits**	**Mnemonic**
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	2-9	vlclbf
if(dct_dc_size_luminance != 0)		
dct_dc_differential	1-11	uimsbf
} else {		
dct_dc_size_chrominance	2-10	vlclbf
if(dct_dc_size_chrominance !=0)		
dct_dc_differential	1-11	uimsbf
}		
} else {		
First DCT coefficient		
}		
while (nextbits() != End of block)		
Subsequent DCT coefficients		
End of block		
}		
}		

6.3 Video bitstream semantics

6.3.1 Semantic rules for higher syntactic structures

This clause details the rules that govern the way in which the higher level syntactic elements may be combined together to produce a legal bitstream. Subsequent clauses detail the semantic meaning of all fields in the video bitstream.

Figure 6-14 illustrates the high level structure of the video bitstream.

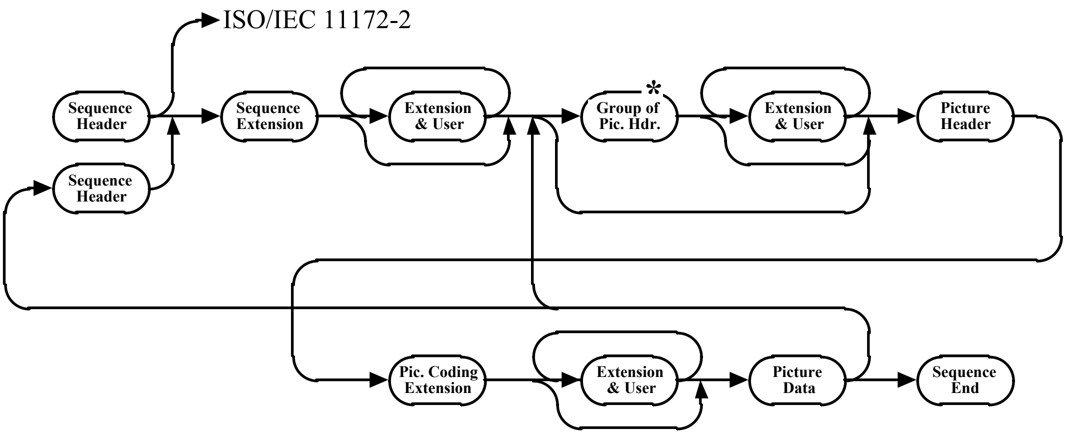
The following semantic rules apply:

- If the first sequence_header() of the sequence is not followed by sequence_extension() then the stream shall conform to ISO/IEC 11172-2 and is not documented within this specification.
- If the first sequence_header() of a sequence is followed by a sequence_extension() then all subsequent occurrences of sequence_header() shall also be immediately followed by a sequence_extension().
- sequence_extension() shall only occur immediately following a sequence_header().
- If sequence_extension() occurs in the bitstream then each picture_header() shall be followed immediately by a picture_coding_extension().
- picture_coding_extension() shall only occur immediately following a picture_header().
- The first coded picture following a group_of_pictures_header() shall be an I-picture.

A number of different extensions are defined in addition to sequence_extension() and picture_coding_extension(). The set of allowed extensions is different at each different point in the syntax where extensions are allowed. Table 6-2 defines a four bit extension_start_code_identifier for each extension.

At each point where extensions are allowed in the bitstream any number of the extensions from the defined allowable set may be included. However each type of extension shall not occur more than once.

In the case that a decoder encounters an extension with an extension identification that is described as “reserved” in this specification the decoder shall discard all subsequent data until the next start code. This requirement allows future definition of compatible extensions to this specification.



* After a GOP the first picture shall be an I-picture

Figure 6-14. High level bitstream organisation

Table 6-2. extension_start_code_identifier codes.

extension_start_code_identifier	Name
0000	reserved
0001	Sequence Extension ID
0010	Sequence Display Extension ID
0011	Quant Matrix Extension ID
0100	reserved
0101	Sequence Scalable Extension ID
0110	reserved
0111	Picture Display Extension ID
1000	Picture Coding Extension ID
1001	Picture Spatial Scalable Extension ID
1010	Picture Temporal Scalable Extension ID
1011	reserved
1100	reserved
...	...
1111	reserved

6.3.2 Video sequence

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

6.3.3 Sequence header

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

horizontal_size_value -- This word forms the 12 least significant bits of horizontal_size.

vertical_size_value -- This word forms the 12 least significant bits of vertical_size.

horizontal_size -- The horizontal_size is a 14-bit unsigned integer, the 12 least significant bits are defined in horizontal_size_value, the 2 most significant bits are defined in horizontal_size_extension. The horizontal_size_value is the width of the displayable part of the luminance component of pictures in samples. The width of the encoded luminance component of pictures in macroblocks, mb_width, is (horizontal_size + 15)/16. The displayable part is left-aligned in the encoded pictures.

In order to avoid start code emulation horizontal_size_value shall not be zero. This precludes values of horizontal_size that are multiples of 4096.

vertical_size -- The vertical_size is a 14-bit unsigned integer, the 12 least significant bits are defined in vertical_size_value, the 2 most significant bits are defined in vertical_size_extension. The vertical_size_value is the height of the displayable part of the luminance component of the frame in lines.

In the case that progressive_sequence is “1” the height of the encoded luminance component of pictures in macroblocks, mb_height, is (vertical_size + 15)/16.

In the case that progressive_sequence is “0” the height of the encoded luminance component of pictures in macroblocks, mb_height, is 2*((vertical_size + 31)/32). The height of the encoded luminance component of field pictures in macroblocks, mb_height, is ((vertical_size + 31)/32).

The displayable part is top-aligned in the encoded pictures.

In order to avoid start code emulation vertical_size_value shall not be zero. This precludes values of vertical_size that are multiples of 4096.

aspect_ratio_information -- This is a four-bit integer defined in the Table 6-3.

aspect_ratio_information either specifies that the "sample aspect ratio" (SAR) of the reconstructed frame is 1,0 (square samples) or alternatively it gives the "display aspect ratio" (DAR).

- If sequence_display_extension() is not present then it is intended that the entire reconstructed frame is intended to be mapped to the entire active region of the display. The sample aspect ratio may be calculated as follows:

$$SAR = DAR \times \frac{horizontal\_size}{vertical\_size}$$

Note In this case horizontal_size and vertical_size are constrained by the SAR of the source and the DAR selected.

- If sequence_display_extension() is present then the sample aspect ratio may be calculated as follows:

$$SAR = DAR \times \frac{display\_horizontal\_size}{display\_vertical\_size}$$

Table 6-3. aspect_ratio_information

aspect_ratio_information	Sample Aspect Ratio	DAR
0000	forbidden	forbidden
0001	1,0 (Square Sample)	-
0010	-	3÷4
0011	-	9÷16
0100	-	1÷2,21
0101	-	reserved
...		...
1111	-	reserved

frame_rate_code -- This is a four-bit integer used to define frame_rate_value as shown in Table 6-4. frame_rate may be derived from frame_rate_value, frame_rate_extension_n and frame_rate_extension_d as follows;

frame_rate = frame_rate_value * (frame_rate_extension_n + 1) ÷ (frame_rate_extension_d + 1)

When an entry for the frame rate exists directly in Table 6-4, frame_rate_extension_n and frame_rate_extension_d shall be zero.

If progressive_sequence is "1" the period between two successive frames at the output of the decoding process is the reciprocal of the frame_rate. See Figure 6-15.

If progressive_sequence is "0" the period between two successive fields at the output of the decoding process is half of the reciprocal of the frame_rate. See Figure 6-16.

The frame_rate signalled in the enhancement layer of temporal scalability is the combined frame rate after the temporal remultiplex operation if picture_mux_enable in the sequence_scalable_extension() is set to '1'.

Table 6-4 --- frame_rate_value

frame_rate_code	**frame_rate_value**
0000	forbidden
0001	24 000÷1001 (23,976)
0010	24
0011	25
0100	30 000÷1001 (29,97)
0101	30
0110	50
0111	60 000÷1001 (59,94)
1000	60
. . .	reserved
1111	reserved

bit_rate_value -- The lower 18 bits of bit_rate.

bit_rate -- This is a 30-bit integer. The lower 18 bits of the integer are in bit_rate_value and the upper 12 bits are in bit_rate_extension. The 30-bit integer specifies the bitrate of the bitstream measured in units of 400 bits/second, rounded upwards. The value zero is forbidden.

If the bitstream is a constant bitrate stream, the bitrate specified is the actual rate of operation of the VBV specified in annex C. If the bitstream is a variable bitrate stream, the STD specifications in ISO/IEC 13818-1 supersede the VBV, and the bitrate specified here is used to dimension the transport stream STD (2.4.2 in ITU-T Rec. xxx | ISO/IEC 13818-1), or the program stream STD (2.4.5 in ITU-T Rec. xxx | ISO/IEC 13818-1).

If the bitstream is not a constant rate bitstream the vbv_delay field shall have the value FFFF in hexadecimal.

Given the value encoded in the bitrate field, the bitstream shall be generated so that the video encoding and the worst case multiplex jitter do not cause STD buffer overflow or underflow.

marker_bit -- This is one bit that shall be set to "1". This bit prevents emulation of start codes.

vbv_buffer_size_value -- the lower 10 bits of vbv_buffer_size.

vbv_buffer_size -- vbv_buffer_size is an 18-bit integer. The lower 10 bits of the integer are in vbv_buffer_size_value and the upper 8 bits are in vbv_buffer_size_extension. The integer defines the size of the VBV (Video Buffering Verifier, see 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 Annex C).

constrained_parameters_flag -- This flag (used in ISO/IEC 11172-2) has no meaning in this specification and shall have the value “0”.

load_intra_quantiser_matrix -- See 6.3.7 “Quant matrix extension”

intra_quantiser_matrix -- See 6.3.7 “Quant matrix extension”

load_non_intra_quantiser_matrix -- See 6.3.7 “Quant matrix extension”

non_intra_quantiser_matrix -- See 6.3.7 “Quant matrix extension”

6.3.4 Extension and user data

extension_start_code -- The extension_start_code is the bit string 000001B5 in hexadecimal. It identifies the beginning of extensions beyond ISO/IEC 11172-2.

6.3.4.1 User data

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.

6.3.5 Sequence extension

extension_start_code_identifier -- This is an 4-bit integer which identifies the extension. See Table 6-2.

profile_and_level_indication – This is an 8-bit integer used to signal the profile and level identification. The meaning of the bits is given in clause 8.

Note In a scalable hierarchy the bitstreams of each layer may set profile_and_level_indication to a different value as specified in clause 8.

progressive_sequence – When set to “1” the coded video sequence contains only progressive frame-pictures. When progressive_sequence is set to “0” the coded video sequence may contain both frame-pictures and field-pictures, and frame-picture may be progressive or interlaced frames.

chroma_format - This is a two bit integer indicating the chrominance format as defined in the Table 6-5.

Table 6-5. Meaning of chroma_format

chroma_format	Meaning
00	reserved
01	4:2:0
10	4:2:2
11	4:4:4

horizontal_size_extension -- This word forms the 2 most significant bits from horizontal_size.

vertical_size_extension -- This word forms the 2 most significant bits from vertical_size.

bit_rate_extension -- This word forms the 12 most significant bits from bit_rate.

vbv_buffer_size_extension -- This word forms the 8 most significant bits from vbv_buffer_size.

low_delay - This flag, when set to “1”, indicates that the sequence does not contain any B-pictures, that the frame reordering delay is not present in the VBV description and that VBV buffer underflow may occur.

When set to “0”, it indicates that the sequence may contain B-pictures, that the frame reordering delay is present in the VBV description and that VBV buffer underflow shall not occur.

This flag is not used during the decoding process and therefore can be ignored by decoders, but it is necessary to define and verify the compliance of low-delay bitstreams.

frame_rate_extension_n -- See frame_rate_code.

frame_rate_extension_d -- See frame_rate_code.

6.3.6 Sequence display extension

This specification does not define the display process. The information in this extension does not affect the decoding process and may be ignored by decoders that conform to this specification.

video_format - This is a three bit integer indicating the representation of the pictures before being coded in accordance with this specification. Its meaning is defined in Table 6-6

Table 6-6. Meaning of video_format

video_format	Meaning
000	component
001	PAL
010	NTSC
011	SECAM
100	MAC
101	Unspecified video format
110	reserved
111	reserved

colour_description -- A flag which if set to “1” indicates the presence of colour_primaries, transfer_characteristics and matrix_coefficients in the bitstream.

colour_primaries -- This 8-bit integer describes the chromaticity coordinates of the source primaries, and is defined in Table 6-7.

Table 6-7. Colour Primaries

Value	Primaries
0	(forbidden)
1	ITU-R Recommendation 709 (1990) primary x y green 0,300 0,600 blue 0,150 0,060 red 0,640 0,330 white D65 0,3127 0,3290
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	ITU-R Recommendation 624-4 System M primary x y green 0,21 0,71 blue 0,14 0,08 red 0,67 0,33 white C 0,310 0,316
5	ITU-R Recommendation 624-4 System B, G primary x y green 0,29 0,60 blue 0,15 0,06 red 0,64 0,33 white D65 0,313 0,329
6	SMPTE 170M primary x y green 0,310 0,595 blue 0,155 0,070 red 0,630 0,340 white D65 0,3127 0,3290
7	SMPTE 240M (1987) primary x y green 0,310 0,595 blue 0,155 0,070 red 0,630 0,340 white D65 0,3127 0,3291
8-255	reserved

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the chromaticity is assumed to be that corresponding to colour_primaries having the value 1.

transfer_characteristics -- This 8-bit integer describes the opto-electronic transfer characteristic of the source picture, and is defined in Table 6-8.

Table 6-8. Transfer Characteristics

Value	Transfer Characteristic
0	(forbidden)
1	ITU-R Recommendation 709 (1990) $V = 1{,}099\ L_c^{0{,}45} - 0{,}099$ for $1 \geq L_c \geq 0{,}018$ $V = 4{,}500\ L_c$ for $0{,}018 > L_c \geq 0$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	ITU-R Recommendation 624-4 System M Assumed display gamma 2,2
5	ITU-R Recommendation 624-4 System B, G Assumed display gamma 2,8
6	SMPTE 170M $V = 1{,}099\ L_c^{0{,}45} - 0{,}099$ for $1 \geq L_c \geq 0{,}018$ $V = 4{,}500\ L_c$ for $0{,}018 > L_c \geq 0$
7	SMPTE 240M (1987) $V = 1{,}1115\ L_c^{0{,}45} - 0{,}1115$ for $L_c \geq 0{,}0228$ $V = 4{,}0\ L_c$ for $0{,}0228 > L_c$
8	Linear transfer characteristics i.e. $V = L_c$
9-255	reserved

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the transfer characteristics are assumed to be those corresponding to transfer_characteristics having the value 1.

matrix_coefficients -- This 8-bit integer describes the matrix coefficients used in deriving luminance and chrominance signals from the green, blue, and red primaries, and is defined in Table 6-9.

In this table;

E'_Y is analogue with values between the values 0 and 1

E'_{PB} and E'_{PR} are analogue between the values -0,5 and 0,5

E'_Y is analogue between the values 0 and 1

Y, Cb and Cr are related to E'_Y, E'_{PB} and E'_{PR} by the following formulae.

$Y = (219 * E'_Y) + 16.$

$Cb = (224 * E'_{PB}) + 128.$

$Cr = (224 * E'_{PR}) + 128.$

Note The decoding process given by this specification limits output sample values to the range [0:255]. Thus code words outside the range implied by the above equations may occasionally occur at the output of the decoding process. In particula the code words 0 and 255 may occur.

Table 6-9. Matrix Coefficients

Value	Matrix
0	(forbidden)
1	ITU-R Recommendation 709 (1990). $E'_Y = 0{,}7154\ E'_G + 0{,}0721\ E'_B + 0{,}2125\ E'_R$ $E'_{PB} = -0{,}386\ E'_G + 0{,}500\ E'_B - 0{,}115\ E'_R$ $E'_{PR} = -0{,}454\ E'_G - 0{,}046\ E'_B + 0{,}500\ E'_R$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	FCC $E'_Y = 0{,}59\ E'_G + 0{,}11\ E'_B + 0{,}30\ E'_R$ $E'_{PB} = -0{,}331\ E'_G + 0{,}500\ E'_B - 0{,}169\ E'_R$ $E'_{PR} = -0{,}421\ E'_G - 0{,}079\ E'_B + 0{,}500\ E'_R$
5	ITU-R Recommendation 624-4 System B, G $E'_Y = 0{,}587\ E'_G + 0{,}114\ E'_B + 0{,}299\ E'_R$ $E'_{PB} = -0{,}331\ E'_G + 0{,}500\ E'_B - 0{,}169\ E'_R$ $E'_{PR} = -0{,}419\ E'_G - 0{,}081\ E'_B + 0{,}500\ E'_R$
6	SMPTE 170M $E'_Y = 0{,}587\ E'_G + 0{,}114\ E'_B + 0{,}299\ E'_R$ $E'_{PB} = -0{,}331\ E'_G + 0{,}500\ E'_B - 0{,}169\ E'_R$ $E'_{PR} = -0{,}419\ E'_G - 0{,}081\ E'_B + 0{,}500\ E'_R$
7	SMPTE 240M (1987) $E'_Y = 0{,}701\ E'_G + 0{,}087\ E'_B + 0{,}212\ E'_R$ $E'_{PB} = -0{,}384\ E'_G + 0{,}500\ E'_B - 0{,}116\ E'_R$ $E'_{PR} = -0{,}445\ E'_G - 0{,}055\ E'_B + 0{,}500\ E'_R$
8-255	reserved

In the case that sequence_display_extension() is not present in the bitstream or colour_description is zero the matrix coefficients are assumed to be those corresponding to matrix_coefficients having the value 1.

display_horizontal_size - See display_vertical_size.

display_vertical_size - display_horizontal_size and display_vertical_size together define a rectangle which may be considered as the “intended display’s” active region. If this rectangle is smaller than the encoded frame size then the display process may be expected to display only a portion of the encoded frame. Conversely if the display rectangle is larger than the encoded frame size then the display process may be expected to display the reconstructed frames on a portion of the display device rather than on the whole display device.

display_horizontal_size shall be in the same units as horizontal_size (samples of the encoded frames).

display_vertical_size shall be in the same units as vertical_size (lines of the encoded frames).

display_horizontal_size and display_vertical_size do not affect the decoding process but may be used by the display process that is not standardised in this specification.

6.3.8 Sequence scalable extension

It is a semantic restriction that if a sequence_scalable_extension() is present in the bitstream following a given sequence_extension() then sequence_scalable_extension() shall follow every other occurrence of sequence_extension(). Thus a bitstream is either scalable or it is not scalable. It is not possible to mix scalable and non-scalable coding within a sequence.

scalable_mode - The scalable_mode indicates the type of scalability used in the video sequence. If no sequence_scalable_extension() is present in the bitstream then no scalability is used for that sequence. scalable_mode also indicates the macroblock_type tables to be used. However in the case of spatial scalability if no picture_spatial_scalable_extension() is present for a given picture then that picture shall be decoded in a non-scalable manner (i.e. as if sequence_scalable_extension() had not been present).

Table 6-10. Definition of scalable_mode

scalable_mode	Meaning	picture_spatial_scalable-_extension()	macroblock_type tables
sequence_scalable_extension() not present			B-2, B-3 and B-4
00	data partitioning		B-2, B-3 and B-4
01	spatial scalability	present	B-5, B-6 and B-7
		not present	B-2, B-3 and B-4
10	SNR scalability		B-8
11	temporal scalability		B-2, B-3 and B-4

layer_id - This is an integer which identifies the layers in a scalable hierarchy. The base layer always has layer_id = 0. However the base layer of a scalable hierarchy does not carry a sequence_scalable_extension() and hence layer_id, except in the case of data partitioning. Each successive layer has a layer_id which is one greater than the layer for which it is an enhancement.

In the case of data partitioning layer_id shall be zero for partition zero and layer_id shall be one for partition one.

lower_layer_prediction_horizontal_size -- this a 14-bit integer indicating the horizontal size of the lower layer frame which is used for prediction. This shall contain the value contained in horizontal_size (horizontal_size_value and horizontal_size_extension) in the lower layer bitstream.

lower_layer_prediction_vertical_size -- this a 14-bit integer indicating the vertical size of the lower layer frame which is used for prediction. This shall contain the value contained in vertical_size (vertical_size_value and vertical_size_extension) in the lower layer bitstream.

horizontal_subsampling_factor_m — This affects the spatial scalable upsampling process, as defined in 7.7.2. The value zero is forbidden.

horizontal_subsampling_factor_n — This affects the spatial scalable upsampling process, as defined in 7.7.2. The value zero is forbidden.

vertical_subsampling_factor_m -- This affects the spatial scalable upsampling process, as defined in 7.7.2. The value zero is forbidden.

vertical_subsampling_factor_n — This affects the spatial scalable upsampling process, as defined in 7.7.2. The value zero is forbidden.

picture_mux_enable -- If set to 1, picture_mux_order and picture_mux_factor are used for remultiplexing prior to display.

mux_to_progressive_sequence -- This flag when set to "1" indicates that the decoded pictures corresponding to the two layers shall be temporally multiplexed to generate a progressive sequence for display. When the temporal multiplexing is intended to generate an interlaced sequence this flag shall be "0".

picture_mux_order -- It denotes number of enhancement layer pictures prior to the first base layer picture. It thus assists remultiplexing of pictures prior to display as it contains information for inverting the demultiplexing performed at the encoder.

picture_mux_factor -- It denotes number of enhancement layer pictures between consecutive base layer pictures to allow correct remultiplexing of base and enhancement layers for display. It also assists in remultiplexing of pictures prior to display as it contains information for inverting the temporal demultiplexing performed at the encoder. The value of "000" is reserved.

6.3.9 Group of pictures header

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

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 as shown in Table 6-11. The fields correspond to the fields defined in the IEC standard for "time and control codes for video tape recorders" (see Bibliography, Annex G). The code refers to the first picture after the group of pictures header 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 frame 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.

Note The information carried by time_code plays no part in the decoding process.

Table 6-11 --- time_code

time_code	range of value	No. of bits	Mnemonic
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 indicates the nature of the predictions uses in the B-pictures (if any) immediately following the first coded I-frame following the group of picture header .

closed_gop is set to “1” to indicate that these B-pictures have been encoded using only backward prediction.

This bit is provided for use during any editing which occurs after encoding. If the previous pictures have been removed by editing, broken_link may be set to "1" so that a decoder may avoid displaying these B-Pictures following the first I-Picture following the group of picture header. However if the closed_gop bit is set to “1’, then the editor may choose not to set the broken_link bit as these B-Pictures can be correctly decoded.

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 first B-Pictures (if any) immediately following the first coded I-frame following the group of picture header may not be correctly decoded because the reference frame which is used for prediction is not available (because of the action of editing).

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

6.3.10 Picture header

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

temporal_reference -- The temporal_reference is a 10-bit unsigned integer associated with each input picture. It is incremented by one, modulo 1024, for each input frame. When a frame is coded as two fields the temporal reference in the picture header of both fields is the same.

Following a group start header the temporal reference of the earliest picture (in display order) shall be reset to zero.

picture_coding_type -- The picture_coding_type identifies whether a picture is an intra-coded picture(I), predictive-coded picture(P) or bidirectionally predictive-coded picture(B). The meaning of picture_coding_type is defined in Table 6-12.

Note Intra-coded pictures with only DC coefficients (D-pictures) that may be used in ISO/IEC 11172-2 are not supported by this specification.

Table 6-12 --- picture_coding_type

picture_coding_type	**coding method**
000	forbidden
001	intra-coded (I)
010	predictive-coded (P)
011	bidirectionally-predictive-coded (B)
100	shall not be used (dc intra-coded (D) in ISO/IEC11172-2)
101	reserved
110	reserved
111	reserved

vbv_delay -- The vbv_delay is a 16-bit unsigned integer. For constant bitrate operation, the vbv_delay is used to set the initial occupancy of the decoder's buffer at the start of 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 bitrate, 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 shall wait after receiving the final byte of the picture start code. It may be calculated from the state of the VBV as follows:

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

where:

$n > 0$

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

R = the actual bitrate (i.e. to full accuracy rather than the quantised value given by bit_rate in the sequence header.)

For non-constant bitrate operation vbv_delay shall have the value FFFF in hexadecimal. The bitstream shall be either a constant bitrate stream or a variable bitrate stream throughout the sequence.

full_pel_forward_vector -- This flag that is used in ISO/IEC 11172-2 is not used by this specification. It shall have the value zero.

forward_f_code -- This parameter that is used in ISO/IEC 11172-2 is not used by this specification. It shall have the value seven (all ones).

full_pel_backward_vector -- This flag that is used in ISO/IEC 11172-2 is not used by this specification. It shall have the value zero.

backward_f_code -- This parameter that is used in ISO/IEC 11172-2 is not used by this specification. It shall have the value seven (all ones).

extra_bit_picture -- A bit indicates the presence of the following extra information. If extra_bit_picture is set to "1", extra_information_picture will follow it. If it is set to "0", there are no data following it. extra_bit_picture shall be set to “0”, the value “1” is reserved for possible future extensions defined by ITU-T|ISO/IEC.

extra_information_picture -- Reserved. A decoder conforming to this specification that encounters extra_information_picture in a bitstream shall ignore it (i.e. parse from bitstream and discard). A bitstream conforming to this specification shall not contain this syntax element.

6.3.11 Picture coding extension

f_code[s][t] -- An unsigned integer taking values 1 through 9 (or 15 if unused). The value zero is forbidden. In an I-picture in which concealment_motion_vectors is zero (in which case the value of f_code[s][t] is not used in the decoding process) f_code[s][t] shall take the value 15 (all ones). Similarly, in an I-picture or a P-picture f_code[1][t] is not used in the decoding process (since it refers to backwards motion vectors) and shall take the value 15 (all ones). (See Table 7-7 for the meaning of the indices; s and t.)

intra_dc_precision - This is a 2-bit integer defined in the Table 6-13.

Table 6-13 Intra DC precision

intra_dc_precision	Precision (bits)
00	8
01	9
10	10
11	11

The inverse quantisation process for the Intra DC coefficients is modified by this parameter as explained in 7.4.1.

picture_structure - This is a 2-bit integer defined in the Table 6-14.

Table 6-14 Meaning of picture_structure

picture_structure	Meaning
00	reserved
01	Top Field
10	Bottom Field
11	Frame picture

When a frame is encoded in the form of two field pictures both fields must be of the same picture_coding_type, except where the first encoded field is an I-picture in which case the second may be either an I-picture or a P-picture.

The first encoded field of a frame may be a top-field or a bottom field, and the next field must be of opposite parity.

When a frame is encoded in the form of two field pictures the following syntax elements may be set independently in each field picture:

- f_code[0][0], f_code[0][1]
- f_code[1][0], f_code[1][1]
- intra_dc_precision, concealment_motion_vectors, q_scale_type
- intra_vlc_format, alternate_scan

top_field_first — The meaning of this element depends upon picture_structure, progressive_sequence and repeat_first_field.

If progressive_sequence is equal to 0, this flag indicates what field of a reconstructed frame is output first by the decoding process:

In a field picture top_field_first shall have the value “0”, and the only field output by the decoding process is the decoded field picture.

In a frame picture top_field_first being set to “1” indicates that the top field of the reconstructed frame is the first field output by the decoding process. top_field_first being set to “0” indicates that the bottom field of the reconstructed frame is the first field output by decoding process

If progressive_sequence is equal to 1, this flag, combined with repeat_first_field, indicates how many times (one, two or three) the reconstructed frame is output by the decoding process:

If repeat_first_field is set to 0, top_field_first shall be set to 0. In this case the output of the decoding process corresponding to this reconstructed frame consists of one progressive frame.

If top_field_first is set to 0 and repeat_first_field is set to 1, the output of the decoding process corresponding to this reconstructed frame consists of two progressive frames.

If top_field_first is set to 1 and repeat_first_field is set to 1, the output of the decoding process corresponding to this reconstructed frame consists of three progressive frames.

frame_pred_frame_dct - If this flag is set to “1” then only frame-DCT and frame prediction are used. In a field picture it shall be “0”. frame_pred_frame_dct shall be “1” if progressive_frame is “1”. This flag affects the syntax of the bitstream.

concealment_motion_vectors - This flag has the value “1” to indicate that motion vectors are coded for intra macroblocks.

q_scale_type - This flag affects the inverse quantisation process as described in 7.4.2.2.

intra_vlc_format -- This flag affects the decoding of transform coefficient data as described in 7.2.2.1.

alternate_scan -- This flag affects the decoding of transform coefficient data as described in 7.3.

repeat_first_field -- This flag is applicable only in a frame picture, in a field picture it shall be set to zero and does not affect the decoding process.

If progressive_sequence is equal to 0 and progressive_frame is equal to 0, repeat_first_field shall be zero, and the output of the decoding process corresponding to this reconstructed frame consists of two fields.

If progressive_sequence is equal to 0 and progressive_frame is equal to 1:

If this flag is set to 0, the output of the decoding process corresponding to this reconstructed frame consists of two fields. The first field (top or bottom field as identified by top_field_first) is followed by the other field.

If it is set to 1, the output of the decoding process corresponding to this reconstructed frame consists of three fields. The first field (top or bottom field as identified by top_field_first) is followed by the other field, then the first field is repeated.

If progressive_sequence is equal to 1:

If this flag is set to 0, the output of the decoding process corresponding to this reconstructed frame consists of one frame.

If it is set to 1, the output of the decoding process corresponding to this reconstructed frame consists of two or three frames, depending on the value of top_field_first.

chroma_420_type - If chroma_format is "4:2:0", the value of chroma_420_type shall be the same as progressive_frame; else chroma_420_type has no meaning and shall be equal to zero. This flag exists for historical reasons.

progressive_frame — If progressive_frame is set to 0 it indicates that the two fields of the frame are interlaced fields in which an interval of time of the field period exists between (corresponding spatial samples) of the two fields. In this case the following restriction applies:

- repeat_first_field shall be zero (two field duration).

If progressive_frame is set to 1 it indicates that the two fields (of the frame) are actually from the same time instant as one another. In this case a number of restrictions to other parameters and flags in the bitstream apply:

- picture_structure shall be "Frame"
- frame_pred_frame_dct shall be 1

This parameter is used when the video sequence is used as the lower layer of a spatial scalable sequence. Here it affects the up-sampling process used in forming a prediction in the enhancement layer from the lower layer.

composite_display_flag — This flag is set to 1 to indicate that the following fields that are of use when the input pictures have been coded as (analogue) composite video prior to encoding into a bitstream that complies with this specification. If it is set to 0 then these parameters do not occur in the bitstream.

The information relates to the picture that immediately follows the extension. In the case that this picture is a frame picture the information relates to the first field of that frame. The equivalent information for the second field may be derived (there is no way to represent it in the bitstream).

Note that repeat_first_field will cause a composite video field to be repeated out of colour field sequence. It is recommended that repeat_first_field and composite_display_flag are not both set simultaneously.

v_axis -- A 1-bit integer used only when the bitstream represents a signal that had previously been encoded according to PAL systems. v_axis is set to 1 on a positive sign, v_axis is set to 0 otherwise.

field_sequence -- A 3-bit integer which defines the number of the field in the eight field sequence used in PAL systems or the four field sequence used in NTSC systems as defined in the Table 6-15.

Table 6-15 Definition of field_sequence.

field sequence	frame	field
000	1	1
001	1	2
010	2	3
011	2	4
100	3	5
101	3	6
110	4	7
111	4	8

sub_carrier -- This is a 1-bit integer. Set to 0 means the sub-carrier/line frequency relationship is correct. When set to 1 the relationship is not correct.

burst_amplitude -- This is a 7-bit integer defining the burst amplitude (for PAL and NTSC only). The amplitude of the sub-carrier burst is quantised as a ITU-R Recommendation 601 luminance signal, with the MSB omitted.

sub_carrier_phase -- This is an 8-bit integer defining the phase of the reference sub-carrier at the field-synchronisation datum with respect, to field start as defined in ITU-R Recommendation 470. See Table 6-16.

Table 6-16 Definition of sub_carrier_phase.

sub_carrier_phase	Phase
0	$([360^{o} \div 256] * 0)$
1	$([360^{o} \div 256] * 1)$
...	...
255	$([360^{o} \div 256] * 255)$

6.3.7 Quant matrix extension

Each quantisation matrix has a default set of values. When a sequence_header_code is decoded all matrices shall be reset to their default values. User defined matrices may be downloaded and this can occur in a sequence_header() or in a quant_matrix_extension().

With 4:2:0 data only two matrices are used, one for intra blocks the other for non-intra blocks.

With 4:2:2 or 4:4:4 data four matrices are used. Both an intra and a non-intra matrix are provided for both luminance blocks and for chrominance blocks. Note however that it is possible to download the same user defined matrix into both the luminance and chrominance matrix at the same time.

The default matrix for intra blocks (both luminance and chrominance) is:

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

The default matrix for non-intra blocks (both luminance and chrominance) is:

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

load_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if intra_quantiser_matrix follows. If it is set to "0" then there is no change in the values that shall be used. Note that if this flag is zero in a sequence_header() the default values will be used because the sequence_header_code will reset the matrices to their defaults.

intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, encoded in the default zigzag scanning order as described in 7.3.1, replace the previous values. The first value shall always be 8. For all of the 8-bit unsigned integers, the value zero is forbidden. With 4:2:2 and 4:4:4 data the new values shall be used for both the luminance intra matrix and the chrominance intra matrix. However the chrominance intra matrix may subsequently be loaded with a different matrix.

load_non_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if non_intra_quantiser_matrix follows. If it is set to "0" then there is no change in the values that shall be used. Note that if this flag is zero in a sequence_header() the default values will be used because the sequence_header_code will reset the matrices to their defaults.

non_intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, encoded in the default zigzag scanning order as described in 7.3.1, replace the previous values. For the 8-bit unsigned integers, the value zero is forbidden. With 4:2:2 and 4:4:4 data the new values shall be used for both the luminance non-intra matrix and the chrominance non-intra matrix. However the chrominance non-intra matrix may subsequently be loaded with a different matrix.

load_chroma_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if chroma_intra_quantiser_matrix follows. If it is set to "0" then there is no change in the values that shall be used. If chroma_format is “4:2:0” this flag shall take the value “0”.

chroma_intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, encoded in the default zigzag scanning order as described in 7.3.1, replace the previous values. The first value shall always be 8. For all of the 8-bit unsigned integers, the value zero is forbidden.

load_chroma_non_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if chroma_non_intra_quantiser_matrix follows. If it is set to "0" then there is no change in the values that shall be used. If chroma_format is “4:2:0” this flag shall take the value “0”.

chroma_non_intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, encoded in the default zigzag scanning order as described in 7.3.1, replace the previous values. For the 8-bit unsigned integers, the value zero is forbidden.

6.3.12 Picture display extension

This specification does not define the display process. The information in this extension does not affect the decoding process and may be ignored by decoders that conform to this specification.

The picture display extension allows the position of the display rectangle whose size is specified in sequence_display_extension() to be moved on a picture-by-picture basis. One application for this is the implementation of pan-scan.

frame_centre_horizontal_offset — This is a 16-bit signed integer giving the horizontal offset in units of 1/16th sample. A positive value shall indicate that the centre of the reconstructed frame lies to the right of the centre of the display rectangle.

frame_centre_vertical_offset — This is a 16-bit signed integer giving the vertical offset in units of 1/16th sample. A positive value shall indicate that the centre of the reconstructed frame lies below the centre of the display rectangle.

The dimensions of the rectangular region are defined in the sequence_display_extension(). The coordinates of the region within the coded picture are defined in the picture_display_extension(). Since (in the case of an interlaced sequence) a coded picture may relate to one, two or three decoded fields the picture_display_extension() may contain up to three offsets.

The number of frame centre offsets in the picture_display_extension() shall be defined as follows:

```
if (( progressive_sequence == 1) || (picture_structure == “field”)) {
    number_of_frame_centre_offsets = 1
} else {
    if (repeat_first_field == “1” )
        number_of_frame_centre_offsets = 3
    else
        number_of_frame_centre_offsets = 2
}
```

A picture_display_extension() shall not occur unless a sequence_display_extension() followed the previous sequence_header().

In the case that a given picture does not have a picture_display_extension() then the most recently decoded frame centre offset shall be used. Note that each of the missing frame centre offsets have the same value (even if two or three frame centre offstes would have been contained in the picture_display_extension() had been present). Following a sequence_header() the value zero shall be used for all frame centre offsets until a picture_display_extension() defines non-zero values.

Figure 6-18 illustrates the picture display parameters. As shown the frame centre offsets contained in the picture_display_extension() shall specify the position of the centre of the reconstructed frame from the centre of the display rectangle.

Note The display rectangle may also be larger than the reconstructed frame.

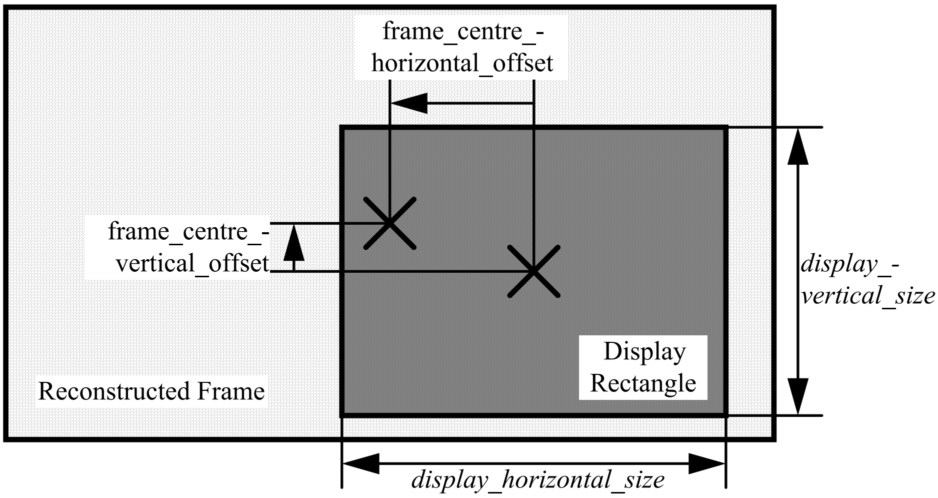


Figure 6-18. Frame centre offset parameters

Note Even in a field picture the frame_centre_vertical_offset still represents the offset of the centre of the frame in 1/16ths of a **frame** line (not a line in the field).

6.3.12.1 Pan-scan

The frame centre offsets may be used to implement pan-scan in which a rectangular region is defined which may be panned around the entire reconstructed frame.

By way of example only; this facility may be used to identify a 3/4 aspect ratio window in a 9/16 coded picture format. This would allow a decoder to produce usable pictures for a conventional definition television set from an encoded format intended for enhanced definition. The 3/4 aspect ratio region is intended to contain the “most interesting” region of the picture.

The 3/4 region is defined by display_horizontal_size and display_vertical_size. The 9/16 frame size is defined by horizontal_size and vertical_size.

6.3.14 Picture temporal scalable extension

reference_select_code -- This is a 2-bit code that identifies reference frames or reference fields for prediction depending on the picture type.

forward_temporal_reference -- An unsigned integer value which indicates temporal reference of the lower layer frame to be used to provide the forward prediction. If the lower layer indicates temporal reference with more than 10 bits, the least significant bits are encoded here. If the lower layer indicates temporal reference with fewer than 10 bits, all bits are encoded here and the more significant bits shall be set to zero.

backward_temporal_reference -- An unsigned integer value which indicates temporal reference of the lower layer frame to be used to provide the backward prediction. If the lower layer indicates temporal reference with more than 10 bits, the least significant bits are encoded here. If the lower layer indicates temporal reference with fewer than 10 bits, all bits are encoded here and the more significant bits shall be set to zero.

6.3.13 Picture spatial scalable extension

lower_layer_temporal_reference - An unsigned integer value which indicates temporal reference of the lower layer frame to be used to provide the prediction. If the lower layer indicates temporal reference with more than 10 bits, the least significant bits are encoded here. If the lower layer indicates temporal reference with fewer than 10 bits, all bits are encoded here and the more significant bits shall be set to zero.

lower_layer_horizontal_offset - This signed integer specifies the horizontal offset of the (top left hand corner) of the upsampled lower layer frame relative to the enhancement layer picture. It is expressed in units of the enhancement layer picture sample width. If the chrominance format is 4:2:0 or 4:2:2 then this parameter shall be an even number.

lower_layer_vertical_offset - This signed integer specifies the vertical offset of the (top left hand corner) of the upsampled lower layer picture relative to the enhancement layer picture. It is expressed in units of the enhancement layer picture sample height. If the chrominance format is 4:2:0 then this parameter shall be an even number.

spatial_temporal_weight_code_table_index -- This indicates which Table of spatial temporal weight codes is to be used as defined in 7.7. Permissible values of spatial_temporal_weight_code_table_index are defined in Table 7-20.

lower_layer_progressive_frame -- This flag shall be set to 0 if the lower layer frame is nterlaced and shall be set to "1" if the lower layer frame is progressive. The use of this flag in the spatial scalable upsampling process is defined in 7.7.

lower_layer_deinterlaced_field_select -- This affects the spatial scalable upsampling process, as defined in 7.7.

6.3.15 Slice

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

slice_vertical_position -- This is given by the last eight bits of the slice_start_code. It is an unsigned integer giving the vertical position in macroblock units of the first macroblock in the slice.

In large pictures (when the vertical size of the frame is greater than 2800 lines) the slice vertical position is extended by the **slice_vertical_position_extension**.

The macroblock row may be calculated as follows:

```
if ( vertical_size > 2800 )
    mb_row = (slice_vertical_position_extension << 7) + slice_vertical_position - 1;
else
    mb_row = slice_vertical_position - 1;
```

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 unless slice_vertical_position_extension is present in which case slice_vertical_position shall be in the range [1:128].

priority_breakpoint — This is a 7-bit integer that indicates the point in the syntax where the bitstream shall be partitioned. The allowed values and their semantic interpretation is given in Table 7-28. priority_breakpoint shall take the value zero in partition 1.

quantiser_scale_code -- An unsigned integer in the range 1 to 31 . The decoder shall use this value until another quantiser_scale_code is encountered either in slice() or macroblock(). The value zero is forbidden.

intra_slice_flag - This flag shall be set to “1” to indicate the presence of intra_slice and reserved_bits in the bitstream.

intra_slice - This shall be set to “0” if any of the macroblocks in the slice are non-intra macroblocks. If all of the macroblocks are intra macroblocks then intra_slice may be set to “1”. intra_slice may be omitted from the bitstream (by setting intra_slice_flag to “0”) in which case it shall be assumed to have the value zero.

intra_slice is not used by the decoding process. intra_slice is intended to aid a DSM application in performing FF/FR (see D.11).

reserved_bits - These seven bits shall have the value zero.

extra_bit_slice -- A bit indicates the presence of the following extra information. If extra_bit_slice is set to "1", extra_information_slice will follow it. If it is set to "0", there are no data following it. extra_bit_slice shall be set to “0”, the value “1” is reserved for possible future extensions defined by ITU-T|ISO/IEC.

extra_information_slice -- Reserved. A decoder conforming to this specification that encounters extra_information_slice in a bitstream shall ignore it (i.e. parse from bitstream and discard). A bitstream conforming to this specification shall not contain this syntax element.

6.3.16 Macroblock

Note “macroblock_stuffing” which is supported in ISO/IEC11172-2 shall not be used in a bitstream defined by this specification.

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 Annex B Table B-1 which indicates the difference between macroblock_address and previous_macroblock_address. The maximum value of macroblock_address_increment is 33. Values greater than this can be encoded using the macroblock_escape codeword.

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

The previous_macroblock_address is a variable defining the absolute position of the last non-skipped macroblock (see 7.6.6 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 = (mb_row * mb_width) -1

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

mb_column = macroblock_address % mb_width

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

Except at the start of a slice, if the value of macroblock_address recovered from macroblock_address_increment and the macroblock_escape codes (if any) differs from the previous_macroblock_address by more than one then some macroblocks have been skipped. It is a requirement that;

- There shall be no skipped macroblocks in I-pictures except when

  either picture_spatial_scalable_extension() follows the picture_header() of the current picture.

  or sequence_scalable_extension() is present in the bitstream and scalable_mode = “SNR scalability”.

- The first and last macroblock of a slice shall not be skipped.
- In a B-picture there shall be no skipped macroblocks immediately following a macroblock in which macroblock_intra is one.

6.3.16.1 Macroblock modes

macroblock_type -- Variable length coded indicator which indicates the method of coding and content of the macroblock according to the Tables B-2 through B-8, selected by picture_coding_type and scalable_mode.

macroblock_quant -- Derived from macroblock_type according to the Tables B-2 through B-8. This is set to 1 to indicate that quant_scale_code is present in the bitstream.

macroblock_motion_forward -- Derived from macroblock_type according to the Tables B-2 through B-8. This flag affects the bitstream syntax and is used by the decoding process.

macroblock_motion_backward -- Derived from macroblock_type according to the Tables B-2 through B-8. This flag affects the bitstream syntax and is used by the decoding process.

macroblock_pattern -- Derived from macroblock_type according to the Tables B-2 through B-8. This is set to 1 to indicate that coded_block_pattern() is present in the bitstream.

macroblock_intra -- Derived from macroblock_type according to the Tables B-2 through B-8. This flag affects the bitstream syntax and is used by the decoding process.

spatial_temporal_weight_code_flag -- Derived from the macroblock_type. This indicates whether the spatial_temporal_weight_code is present in the bitstream.

spatial_temporal_weight_code -- This is a two bit code which indicates, in the case of spatial scalability, how the spatial and temporal predictions shall be combined to form the prediction for the macroblock. A full description of how to form the spatial scalable prediction is given in 7.7.

frame_motion_type - This is a two bit code indicating the macroblock motion prediction, defined in Table 6-17.

If frame_pred_frame_dct is equal to 1 then frame_motion_type is omitted from the bitstream. In this case motion vector decoding and prediction formation shall be performed as if frame_motion_type had indicated “Frame-based prediction”.

In the case of intra macroblocks (in a frame picture) when concealment_motion_vectors is equal to 1 frame_motion_type is not present in the bitstream. In this case motion vector decoding and update of the motion vector predictors shall be performed as if frame_motion_type had indicated “Frame-based prediction”. See 7.6.3.9.

Table 6-17 Meaning of frame_motion_type

code	spatial_temporal _weight_class	prediction type	motion_vector _count	mv_format	dmv
00		reserved			
01	0,1	Field-based prediction	2	field	0
01	2,3	Field-based prediction	1	field	0
10	0,1,2,3	Frame-based prediction	1	frame	0
11	0,2,3	Dual-Prime	1	field	1

field_motion_type - This is a two bit code indicating the macroblock motion prediction, defined in Table 6-18.

In the case of intra macroblocks (in a field picture) when concealment_motion_vectors is equal to 1 field_motion_type is not present in the bitstream. In this case motion vector decoding and update of the motion vector predictors shall be performed shall be performed as if field_motion_type had indicated “Field-based prediction”. See 7.6.3.9.

Table 6-18 Meaning of field_motion_type

code	spatial_temporal _weight_class	prediction type	motion_vector _count	mv_format	dmv
00		reserved			
01	0,1	Field-based prediction	1	field	0
10	0,1	16x8 MC	2	field	0
11	0	Dual-Prime	1	field	1

decode_dct_type - This is a flag (derived from various bitstream elements) that determines whether the following syntax element dct_type is present in the bitstream. It is derived as follows:

```
if ( (picture_structure == “frame” ) && (frame_pred_frame_dct == 0) &&
        (macroblock_intra || macroblock_pattern) ) {
    decode_dct_type = 1;
} else
    decode_dct_type = 0;
```

dct_type - This is a flag indicating whether the macroblock is frame DCT coded or field DCT coded. If this is set to "1", the macroblock is field DCT coded. dct_type is only included in the bitstream if decode_dct_type is non-zero.

In the case that decode_dct_type is zero then dct_type (used in the remainder of the decoding process) shall be derived as shown in Table 6-19.

Table 6-19. Value of dct_type if dct_type is not in the bitstream.

Condition	dct_type
picture_structure == “field”	unused because there is no frame/field distinction in a field picture.
frame_pred_frame_dct == 1	0 (“frame”)
!(macroblock_intra || macroblock_pattern)	unused - macroblock is not coded
macroblock is skipped	unused - macroblock is not coded

6.3.16.2 Motion vectors

motion_vector_count is derived from field_motion_type or frame_motion_type as indicated in Table 6-17 and Table 6-18.

mv_format is derived from field_motion_type or frame_motion_type as indicated in the Table 6-17 and Table 6-18. mv_format indicates if the motion vector is a field-motion vector or a frame-motion vector. mv_format is used in the syntax of the motion vectors and in the process of motion vector prediction.

dmv is derived from field_motion_type or frame_motion_type as indicated in Table 6-17 and Table 6-18

motion_vertical_field_select[r][s] — This flag indicates which reference field shall be used to form the prediction. If motion_vertical_field_select[r][s] is zero then the top reference field shall be used, if it is one then the bottom reference field shall be used. (See Table 7-7 for the meaning of the indices; r and s.)

6.3.16.3 Motion vector

motion_code[r][s][t] — This is a variable length code which is used in motion vector decoding as described in 7.6.3.1. (See Table 7-7 for the meaning of the indices; r, s and t.)

motion_residual[r][s][t] — This is an integer which is used in motion vector decoding as described in 7.6.3.1. (See Table 7-7 for the meaning of the indices; r, s and t.) The number of bits in the bitstream for motion_residual[r][s][t], r_size, is derived from f_code[s][t] as follows;

$$r\_size = f\_code[s][t] - 1$$

Note: The number of bits for both motion_residual[0][s][t] and motion_residual[1][s][t] is denoted by f_code[s][t].

dmvector[t] — This is a variable length code which is used in motion vector decoding as described in 7.6.3.1. (See Table 7-7 for the meaning of the index; t.)

6.3.16.3 Coded block patern

coded_block_pattern_420 — A variable length code that is used to derive the variable cbp according to Table B-9.

coded_block_pattern_1 —

coded_block_pattern_2 — For 4:2:2 and 4:4:4 data the coded block pattern is extended by the addition of either a two bit or six bit fixed length code, coded_block_pattern_1 or coded_block_pattern_2. Then the pattern_code[i] is derived from cbp using the following:

```
for (i=0; i<12; i++)
    if (macroblock_intra != 0)
        pattern_code[i] = 1;
    else
        pattern_code[i] = 0;

if (macroblock_intra == 0) {
    for (i=0; i<6; i++)
        if ( cbp & (1<<(5-i)) ) pattern_code[i] = 1;
    if (chroma_format == "4:2:2")
        for (i=6; i<8; i++)
            if ( coded_block_pattern_1 & (1<<(7-i)) ) pattern_code[i] = 1;
    if (chroma_format == "4:4:4")
        for (i=6; i<12; i++)
            if ( coded_block_pattern_2 & (1<<(11-i)) ) pattern_code[i] = 1;
}
```

If pattern_code[i] equals to 1, i=0 to (block_count-1), then the block number i defined in Figures 6-8, 6-9 and 6-10 is contained in this macroblock.

The number “block_count” which determines the number of blocks in the macroblock is derived from the chrominance format as shown in Table 6-20.

Table 6-20 block_count as a function of chroma_format

chroma_format	block_count
4:2:0	6
4:2:2	8
4:4:4	12

6.3.17 Block

The semantics of block() are described in clause 7.

7 The video decoding process

This clause specifies the decoding process that a decoder shall perform to recover picture data from the coded bitstream.

With the exception of the Inverse Discrete Cosine Transform (IDCT) the decoding process is defined such that all decoders shall produce numerically identical results. Any decoding process that produces identical results to the process described here, by definition, complies with this specification.

The IDCT is defined statistically in order that different implementations for this function are allowed. The IDCT specification is given in Annex A.

In 7.1 through 7.6 the simplest decoding process is specified in which no scalability features are used. 7.7 to 7.11 specify the decoding process when scalable extensions are used.

Figure 7-1 is a diagram of the Video Decoding Process without any scaling. The diagram is simplified for clarity.

Note Throughout this specification two dimensional arrays are represented as *name*[*q*][*p*] where '*q*' is the index in the vertical dimension and '*p*' the index in the horizontal dimension.

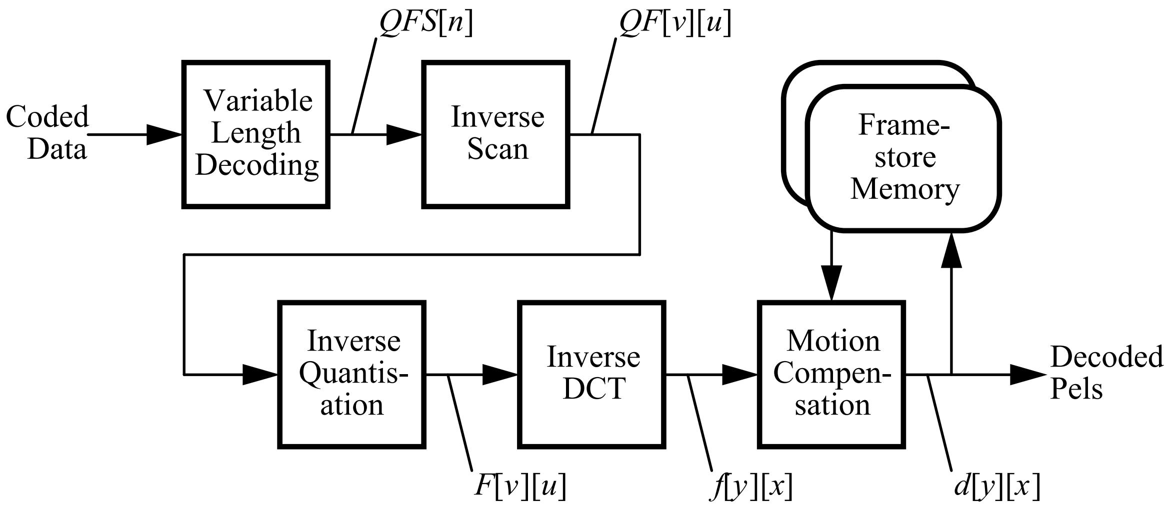


Figure 7-1. Simplified Video Decoding Process

7.1 Higher syntactic structures

The various parameters and flags in the bitstream for macroblock() and all syntactic structures above macroblock() shall be interpreted as indicated in clause 6. Many of these parameters and flags affect the decoding process described in the following clauses. Once all of the macroblocks in a given picture have been processed the entire picture will have been reconstructed.

Reconstructed field pictures shall be associated together in pairs to form reconstructed frames. (See "picture_structure" in 6.3.11.)

The sequence of reconstructed frames shall be reordered as described in 6.1.1.1.

If progressive_sequence == 1 the reconstructed frames shall be output from the decoding process at regular intervals of the frame period as shown in Figure 6-15.

If progressive_sequence == 0 the reconstructed frames shall be broken into a sequence of fields which shall be output from the decoding process at regular intervals of the field period as shown in Figure 6-16. In the case that a frame picture has repeat_first_field == 1 the first field of the frame shall be repeated after the second field. (See "repeat_first_field" in 6.3.11.)

7.2 Variable length decoding

7.2.1 specifies the decoding process used for the DC coefficient (*n*=0) in an intra coded block. (*n* is the index of the coefficient in the appropriate zig-zag scan order.) 7.2.2 specifies the decoding process for all other coefficients; AC coefficients (*n*≠0) and DC coefficients in non-intra coded blocks.

Let *cc* denote the colour component. It is related to the block number as specified in Table 7-1. Thus *cc* is zero for the Y component, one for the C_b component and two for the C_r component.

Table 7-1. Definition of *cc*, colour component index

Block Number	***cc***		
	4:2:0	**4:2:2**	**4:4:4**
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	1	1	1
5	2	2	2
6		1	1
7		2	2
8			1
9			2
10			1
11			2

7.2.1 DC coefficients in intra blocks

DC coefficients in blocks in intra macroblocks are encoded as a variable length code denoting dct_dc_size as defined in Table B-12 and B-13. If dct_dc_size is not equal to zero then this shall be followed by a fixed length code, dc_dct_differential, of dct_dc_size bits. A differential value is first recovered from the coded data which is added to a predictor in order to recover the final decoded coefficient.

If *cc* is zero then Table B-12 shall be used for dct_dc_size. If *cc* is non-zero then Table B-13 shall be used for dct_dc_size.

Three predictors are maintained, one for each of the colour components, *cc*. Each time a DC coefficient in a block in an intra macroblock is decoded the predictor is added to the differential to recover the actual coefficient. Then the predictor shall be set to the value of the coefficient just decoded. At various times, as described below, the predictors shall be reset. The reset value is derived from the parameter intra_dc_precision as specified in Table 7-2.

Table 7-2. Relation between intra_dc_precision and the predictor reset value

intra_dc_precision	Bits of precision	reset value
0	8	128
1	9	256
2	10	512
3	11	1024

The predictors shall be reset to the reset value at the following times:

- At the start of a slice.
- Whenever a non-intra macroblock is decoded.
- Whenever a macroblock is skipped. i.e. when macroblock_address_increment > 1.

The predictors are denoted *dc_dct_pred*[*cc*].

QFS[0] shall be calculated from dc_dct_size and dc_dct_differential by any process equivalent to:

```
if ( dc_dct_size == 0 ) {
    dct_diff = 0;
} else {
    half_range = 2 ^ ( dc_dct_size - 1 );                Note ^ denotes power (not XOR)
    if ( dc_dct_differential >= half_range )
        dct_diff = dc_dct_differential;
    else
        dct_diff = (dc_dct_differential + 1) - (2 * half_range);
}
QFS[0] = dc_dct_pred[cc] + dct_diff;
dc_dct_pred[cc] = QFS[0]
```

Note *dct_diff* and *half_range* are temporary variables which are not used elsewhere in this specification.

It is a requirement of the bitstream that QFS[0] shall lie in the range;

0 to ((2^(8 + intra_dc_precision))-1)

7.2.2 Other coefficients

All coefficients with the exception of the DC intra coefficients shall be encoded using Tables B-14, B-15 and B-16.

In all cases a variable length code shall first be decoded using either Table B-14 or Table B-15. The decoded value of this code denotes one of three courses of action:

1 End of Block. In this case there are no more coefficients in the block in which case the remainder of the coefficients in the block (those for which no value has yet been decoded) shall be set to zero. This is denoted by “End of block” in the syntax specification of 6.2.6.

2 A “normal” coefficient in which a value of *run* and *level* is decoded followed by a single bit, *s*, giving the sign of the coefficient *signed_level* is computed from *level* and *s* as shown below. *run* coefficients shall be set to zero and the subsequent coefficient shall have the value *signed_level.*

```
if (s ==0)
    signed_level = level;
else
    signed_level = (-level);
```

3 An “Escape” coded coefficient. In which the values of *run* and *signed_level* are fixed length coded as described in 7.2.2.3.

7.2.2.1 Table selection

Table 7-3 indicates which Table shall be used for decoding the DCT coefficients.

Table 7-3. Selection of DCT coefficient VLC tables

intra_vlc_format	**0**	**1**
intra blocks (macroblock_intra = 1)	B-14	B-15
non-intra blocks (macroblock_intra = 0)	B-14	B-14

7.2.2.2 First coefficient of a non-intra block

In the case of the first coefficient of a non-intra block (a block in a non-intra macroblock) Table B-14 is modified as indicated by “NOTE 2” and “NOTE 3” at the foot of that Table.

This modification only affects the entry that represents run = 0, level = ±1. Since it is not possible to encode an End of block as the first coefficient of a block (the block would be “not coded” in this case) no possibility for ambiguity exists.

The positions in the syntax that use this modified Table are denoted by “First DCT coefficient” in the syntax specification of 6.2.6. The remainder of the coefficients are denoted by “Subsequent DCT coefficients”.

Note In the case that Table B-14 is used for an intra block, the first coefficient shall be coded as specified in 7.2.1. Table B-14 shall therefore not be modified as the first coefficient that uses Table B-14 is the second coefficient in the block.

7.2.2.3 Escape coding

Many possible combinations of run and level have no variable length code to represent them. In order to encode these statistically rare combinations an Escape coding method is used.

Table B-16 defines the escape coding method. The Escape VLC is followed by a 6-bit fixed length code giving “*run*”. This is followed by a 12-bit fixed length code giving the values of “*signed_level*”.

Note Attention is drawn to the fact that the escape coding method used in this specification is different to that used in ISO/IEC 11172-2.

7.2.2.4 Summary

To summarise 7.2.2. The variable length decoding process shall be equivalent to the following. At the start of this process *n* shall take the value zero for non-intra blocks and one for intra blocks.

```
eob_not_read = 1;
while ( eob_not_read )
{
    <decode VLC, decode Escape coded coefficient if required>
    if ( <decoded VLC indicates End of block> ) {
        eob_not_read = 0;
        while ( n< 64) {
            QFS[n] = 0;
            n = n + 1;
        }
    } else {
        for ( m = 0; m < run; m++ ) {
            QFS[n] = 0;
            n = n + 1;
        }
        QFS[n] = signed_level
        n = n + 1;
    }
}
```

Note *eob_not_read* and *m* are temporary variables that are not used elsewhere in this specification.

7.3 Inverse scan

Let the data at the output of the variable length decoder be denoted by *QFS*[*n*]. *n* is in the range 0 to 63.

This clause specifies the way in which the one-dimensional data, *QFS*[*n*], is converted into a two-dimensional array of coefficients denoted by *QF*[*v*][*u*]. *u* and *v* both lie in the range 0 to 7.

Two scan patterns are defined. The scan that shall be used shall be determined by alternate_scan which is encoded in the picture header extension.

Figure 7-1 defines *scan*[alternate_scan][*v*][*u*] for the case that alternate_scan is zero. Figure 7-2 defines *scan*[alternate_scan][*v*][*u*] for the case that alternate_scan is one.

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

Figure 7-1. Definition of *scan*[0][*v*][*u*]

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

Figure 7-2. Definition of *scan*[1][*v*][*u*]

The inverse scan shall be any process equivalent to the following:

```
for (v=0; v<8; v++)
    for (u=0; u<8; u++)
        QF[v][u] = QFS[scan[alternate_scan][v][u]]
```

7.3.1 Inverse scan for matrix download

When the quantisation matrices are downloaded they are encoded in the bitstream in a scan order that is converted into the two-dimensional matrix used in the inverse quantiser in an identical manner to that used for coefficients.

For matrix download the scan defined by Figure 7-1 (i.e. scan zero) shall always be used.

Let $W[w][u][v]$ denote the weighting matrix in the inverse quantiser (see 7.4.2.1), and $W'[w][n]$ denote the matrix as it is encoded in the bitstream. The matrix download shall then be equivalent to the following:

```
for (v=0; v<8; v++)
    for (u=0; u<8; u++)
        W[w][v][u] = W'[w][scan[0][v][u]]
```

7.4 Inverse quantisation

The two-dimensional array of coefficients, $QF[v][u]$, is inverse quantised to produce the reconstructed DCT coefficients. This process is essentially a multiplication by the quantiser step size. The quantiser step size is modified by two mechanisms; a weighting matrix is used to modify the step size within a block and a scale factor is used in order that the step size can be modified at the cost of only a few bits (as compared to encoding an entire new weighting matrix).

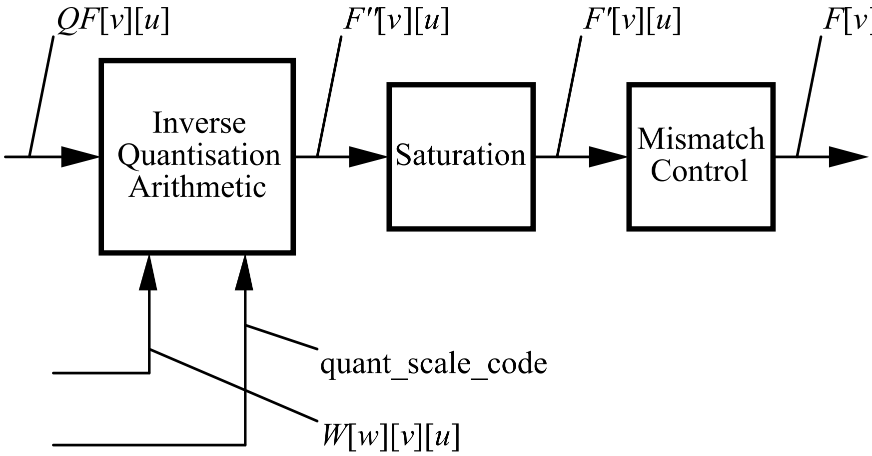


Figure 7-3. Inverse quantisation process

Figure 7-3 illustrates the overall inverse quantisation process. After the appropriate inverse quantisation arithmetic the resulting coefficients, $F''[v][u]$, are saturated to yield $F'[v][u]$ and then a mismatch control operation is performed to give the final reconstructed DCT coefficients, $F[v][u]$.

Note Attention is drawn to the fact that the method of achieving mismatch control in this specification is different to that employed by ISO/IEC 11172-2.

7.4.1 Intra DC coefficient

The DC coefficients of intra coded blocks shall be inverse quantised in a different manner to all other coefficients.

In intra blocks $F''[0][0]$ shall be obtained by multiplying $QF[0][0]$ by a constant multiplier, *intra_dc_mult*, (constant in the sense that it is not modified by either the weighting matrix or the scale factor). The multiplier is related to the parameter intra_dc_precision that is encoded in the picture coding extension. Table 7-4 specifies the relation between intra_dc_precision and *intra_dc_mult*.

Table 7-4. Relation between intra_dc_precision and *intra_dc_mult*

intra_dc_precision	Bits of precision	*intra_dc_mult*
0	8	8
1	9	4
2	10	2
3	11	1

Thus; $F''[0][0] = intra\_dc\_mult \times QF[0][0]$

7.4.2 Other coefficients

All coefficients other than the DC coefficient of an intra block shall be inverse quantised as specified in this clause.

7.4.2.1 Weighting matrices

When 4:2:0 data is used two weighting matrices are used. One shall be used for intra macroblocks and the other for non-intra macroblocks. When 4:2:2 or 4:4:4 data is used, four matrices are used allowing different matrices to be used for luminance and chrominance data. Each matrix has a default set of values which may be overwritten by down-loading a user defined matrix as explained in 6.2.3.2.

Let the weighting matrices be denoted by $W[w][v][u]$ where w takes the values 0 to 3 indicating which of the matrices is being used. Table 7-5 summarises the rules governing the selection of w.

Table 7-5. Selection of *w*

	4:2:0		**4:2:2 and 4:4:4**	
	luminance (cc = 0)	**chrominance (cc ≠ 0)**	**luminance (cc = 0)**	**chrominance (cc ≠ 0)**
intra blocks (macroblock_intra = 1)	0	0	0	2
non-intra blocks (macroblock_intra = 0)	1	1	1	3

7.4.2.2 Quantiser scale factor

The quantisation scale factor is encoded as a fixed length code, quantiser_scale_code. This indicates the appropriate *quantiser_scale* to apply in the inverse quantisation arithmetic.

q_scale_type (encoded in the picture coding extension) indicates which of two mappings between quantiser_scale_code and *quantiser_scale* shall apply. Table 7-6 shows the two mappings between quantiser_scale_code and *quantiser_scale*.

Table 7-6. Relation between *quantiser_scale* and quantiser_scale_code

quantiser_scale_code	quantiser_scale[q_scale_type]	
	q_scale_type = 0	q_scale_type = 1
0	(forbidden)	
1	2	1
2	4	2
3	6	3
4	8	4
5	10	5
6	12	6
7	14	7
8	16	8
9	18	10
10	20	12
11	22	14
12	24	16
13	26	18
14	28	20
15	30	22
16	32	24
17	34	28
18	36	32
19	38	36
20	40	40
21	42	44
22	44	48
23	46	52
24	48	56
25	50	64
26	52	72
27	54	80
28	56	88
29	58	96
30	60	104
31	62	112

7.4.2.3 Reconstruction formulae

The following equation specifies the arithmetic to reconstruct $F''[v][u]$ from $QF[v][u]$ (for all coefficients except intra DC coefficients).

$$F''[v][u] = ((2\,QF[v][u] + k) \times W[w][v][u] \times quantiser\_scale) / 32$$

where:

$$k = \begin{cases} 0 & \text{intra blocks} \\ Sign(QF[v][u]) & \text{non - intra blocks} \end{cases}$$

Note The above equation uses the “/” operator as defined in 4.1.

7.4.3 Saturation

The coefficients resulting from the Inverse Quantisation Arithmetic are saturated to lie in the range $[-2048{:}+2047]$. Thus;

$$F'[v][u] = \begin{cases} 2047 & F''[v][u] > 2047 \\ F''[u][v] & -2048 \le F''[v][u] \le 2047 \\ -2048 & F''[v][u] < -2048 \end{cases}$$

7.4.4 Mismatch control

Mismatch control shall be performed by any process equivalent to the following. Firstly all of the reconstructed, saturated coefficients, $F'[v][u]$ in the block shall be summed. This value is then tested to determine whether it is odd or even. If the sum is even then a correction shall be made to just one coefficient; $F[7][7]$. Thus;

$$sum = \sum_{v=0}^{v<8} \sum_{u=0}^{u<8} F'[v][u]$$

$$F[v][u] = F'[v][u] \text{ for all } u,\ v \text{ except } u = v = 7$$

$$F[7][7] = \begin{cases} F'[7][7] & \text{if } sum \text{ is odd} \\ \begin{cases} F'[7][7]-1 & \text{if } F'[7][7] \text{ is odd} \\ F'[7][7]+1 & \text{if } F'[7][7] \text{ is even} \end{cases} & \text{if } sum \text{ is even} \end{cases}$$

Note 1 It may be useful to note that the above correction for $F[7][7]$ may simply be implemented by toggling the least significant bit of the twos complement representation of the coefficient. Also since only the “oddness” or “evenness” of the *sum* is of interest an exclusive OR (of just the least significant bit) may be used to calculate “*sum*”.

Note 2 Warning. Small non-zero inputs to the IDCT may result in zero output for compliant IDCTs. If this occurs in an encoder, mismatch may occur in some pictures in a decoder that uses a different compliant IDCT. An encoder can avoid this problem by checking the output of its own IDCT.

7.4.5 Summary

In summary the inverse quantisation process is any process numerically equivalent to:

```
for (v=0; v<8;v++) {
    for (u=0; u<8;u++) {
        if ( (u==0) && (v==0) && (macroblock_intra) ) {
            F''[v][u] = intra_dc_mult * QF[v][u];
        } else {
            if ( macroblock_intra ) {
                F''[v][u] = ( QF[v][u] * W[w][v][u] * quantiser_scale * 2 ) / 32;
            } else {
                F''[v][u] = ( ( ( QF[v][u] *  2 ) + Sign(QF[v][u]) ) * W[w][v][u]
                                                    * quantiser_scale ) / 32;
            }
        }
    }
}
```

```
sum = 0;
for (v=0; v<8;v++) {
     for (u=0; u<8;u++) {
          if ( F''[v][u] > 2047 ) {
               F'[v][u] = 2047;
          } else {
               if ( F''[v][u] < -2048 ) {
                    F'[v][u] = -2048;
               } else {
                    F'[v][u] = F''[v][u];
               }
          }
     sum = sum + F'[v][u];
     F[v][u] = F'[v][u];
     }
}

if ((sum & 1) == 0) {
     if ((F[7][7] & 1) != 0) {
          F[7][7] = F'[7][7] - 1;
     } else {
          F[7][7] = F'[7][7] + 1;
     }
}
```

7.5 Inverse DCT

Once the DCT coefficients, $F[v][u]$, are reconstructed, the inverse DCT transform defined in Annex A shall be applied to obtain the inverse transformed values, $f[y][x]$. These values shall be saturated so that; $-256 \le f[y][x] \le 255$, for all x, y.

7.5.1 Non-coded blocks and skipped macroblocks

In a macroblock that is not skipped, if pattern_code[i] is one for a given block in the macroblock then coefficient data is included in the bitstream for that block. This is decoded using as specified in the preceding clauses.

However, if pattern_code[i] is zero, or if the macroblock is skipped, then that block contains no coefficient data. The sample domain coefficients $f[y][x]$ for such a block shall all take the value zero.

7.6 Motion compensation

The motion compensation process forms predictions from previously decoded pictures which are combined with the coefficient data (from the output of the IDCT) in order to recover the final decoded samples. Figure 7-4 shows a simplified diagram of this process.

In general up to four separate predictions are formed for each block which are combined together to form the final prediction block p[y][x].

In the case of intra coded macroblocks no prediction is formed so that p[y][x] will be zero. The saturation shown in Figure 7-4 is still required in order to remove negative values from f[y][x]. Intra coded macroblocks may carry motion vectors known as “concealment motion vectors”. Despite this no prediction is formed in the normal course of events. This motion vector information is intended for use in the case that bitstream errors preclude the decoding of coefficient information. The way in which a decoder shall use this information is not specified. The only requirement for these motion vectors is that they shall have the correct syntax for motion vectors. A description of the way in which these motion vectors may be used can be found in 7.6.3.9.

In the case where a block is not coded, either because the entire macroblock is skipped or the specific block is not coded there is no coefficient data. In this case f[y][x] is zero and the decoded samples are simply the prediction, p[y][x].

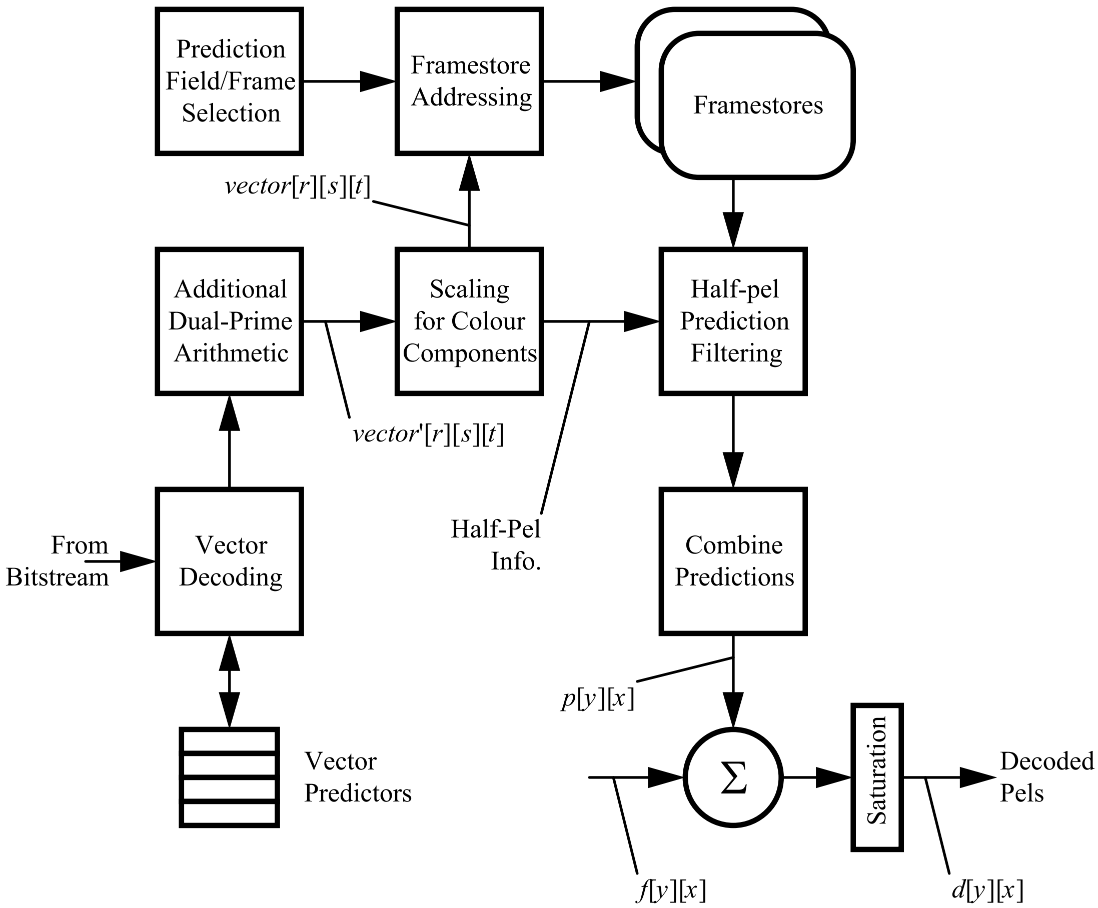


Figure 7-4. Simplified motion compensation process

7.6.1 Prediction modes

There are two major classifications of the prediction mode; field prediction and frame prediction.

In field prediction, predictions are made independently for each field by using data from one or more previously decoded fields. Frame prediction forms a prediction for the frame from one or more previously decoded frames. It must be understood that the fields and frames from which predictions are made may themselves have been decoded as either field pictures or frame pictures.

Within a field picture all predictions are field predictions. However in a frame picture either field predictions or frame predictions may be used (selected on a macroblock-by macroblock basis).

In addition to the major classification of field or frame prediction two special prediction modes are used:

- 16x8 motion compensation. In which two motion vectors are used for each macroblock. The first motion vector is used for the upper 16x8 region, the second for the lower 16x8 region. In the case of a bidirectionally predicted macroblock a total of four motion vectors will be used since there will be two for the forward prediction and two for the backward prediction. In this specification 16x8 motion compensation shall only be used with field pictures.
- Dual-prime. In which only one motion vector is encoded (in its full format) in the bitstream together with a small differential motion vector. In the case of field pictures two motion vectors are then derived from this information. These are used to form predictions from two reference fields (one top, one bottom) which are averaged to form the final prediction. In the case of frame pictures this process is repeated for the two fields so that a total of four field predictions are made. This mode shall only be used in P-pictures where there are no B-pictures between the predicted and reference fields or frames.

7.6.2 Prediction field and frame selection

The selection of which fields and frames shall be used to form predictions shall be made as detailed in this clause.

7.6.2.1 Field prediction

In P-pictures prediction shall be made from the two most recently decoded reference fields. The simplest case illustrated in Figure 7-5 shall be used when predicting the first picture of a coded frame or when using field prediction within a frame-picture. In these cases the two reference fields are part of the same reconstructed frame.

Note 1 The reference fields may themselves have been reconstructed by decoding two field-pictures or a single frame-picture.

Note 2 In the case of predicting a field picture, the field being predicted may be either the top field or the bottom field.

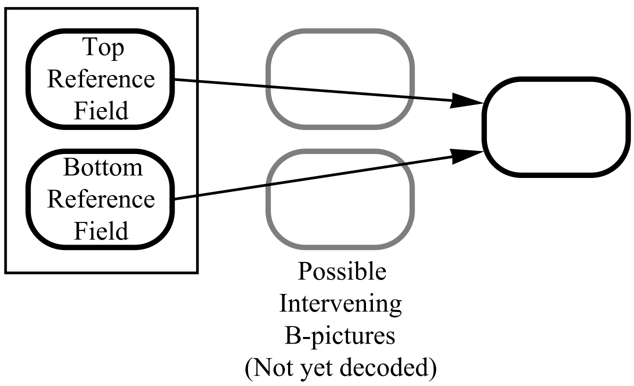


Figure 7-5. Prediction of the first field or field prediction in a frame-picture

The case when predicting the second field picture of a coded frame is more complicated because the two most recently decoded reference fields shall be used, and in this case, the most recent reference field was obtained from decoding the first field picture of the coded frame. Figure 7-6 illustrates the situation when this second picture is the bottom field. Figure 7-7 illustrates the situation when this second picture is the top field.

Note The earlier reference field may itself have been reconstructed by decoding a field picture or a frame picture.

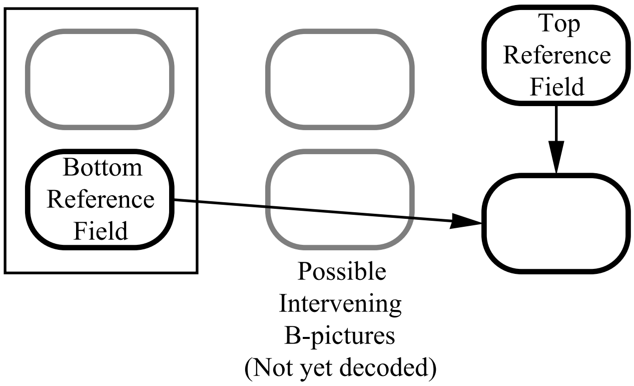


Figure 7-6. Prediction of the second field-picture when it is the bottom field

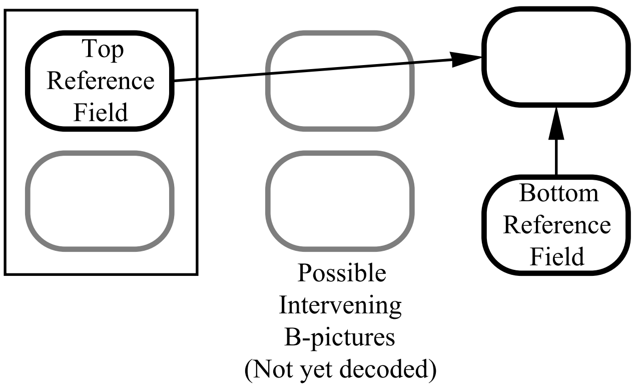


Figure 7-7. Prediction of the second field-picture when it is the top field

Field prediction in B-pictures shall be made from the two fields of the two most recently reconstructed reference frames. Figure 7-8 illustrates this situation.

Note The reference frames may themselves have been reconstructed from two field-pictures or a single frame-picture.

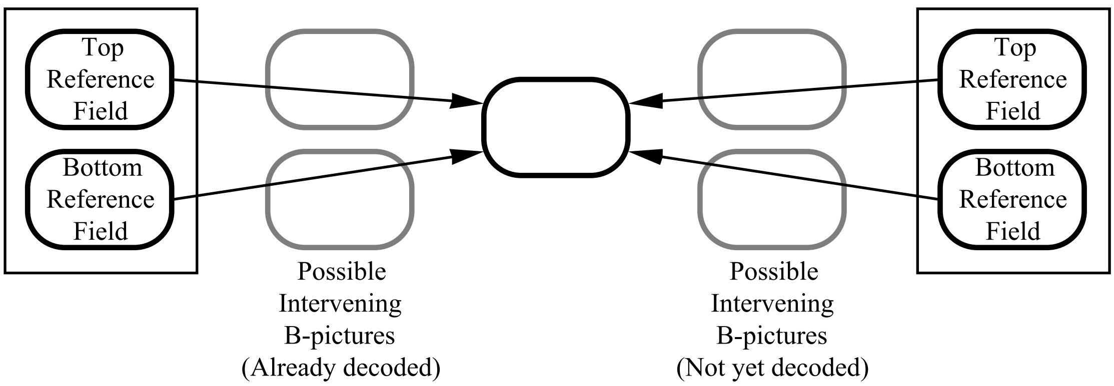


Figure 7-8. Field-prediction of B field pictures or B frame pictures

7.6.2.2 Frame prediction

In P-pictures prediction shall be made from the most recently reconstructed reference frame. This is illustrated in Figure 7-9.

Note The reference frame may itself have been coded as two field pictures or a single frame picture.

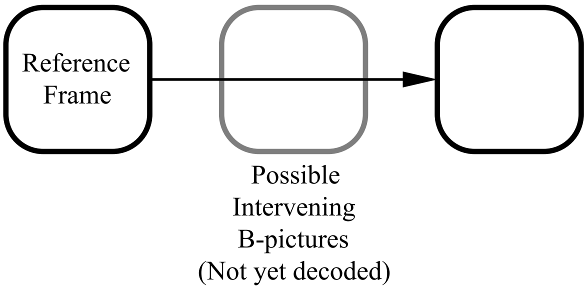


Figure 7-9. Frame-prediction for I-pictures and P-pictures

Similarly frame prediction in B-pictures shall be made from the two most recently reconstructed reference frames as illustrated in Figure 7-10.

Note The reference frames themselves may each have been coded as either two field pictures or a single frame picture.

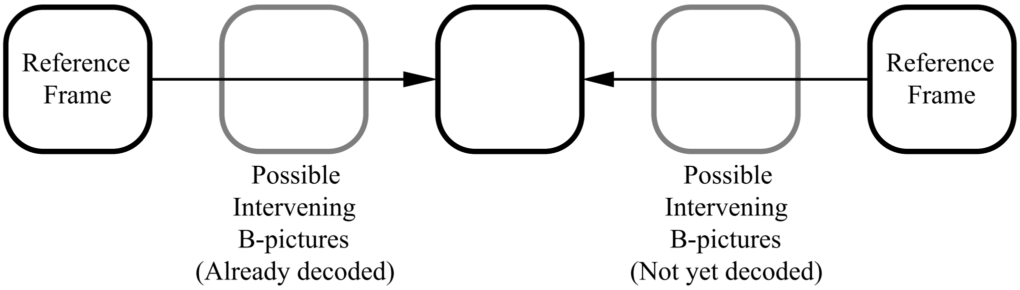


Figure 7-10. Frame-prediction for B-pictures

7.6.3 Motion vectors

Motion vectors are coded differentially with respect to previously decoded motion vectors in order to reduce the number of bits required to represent them. In order to decode the motion vectors the decoder shall maintain four motion vector predictors (each with a horizontal and vertical component) denoted *PMV*[*r*][*s*][*t*]. For each prediction, a motion vector, *vector'*[*r*][*s*][*t*] is first derived. This is then scaled depending on the sampling structure (4:2:0, 4:2:2 or 4:4:4) to give a motion vector, *vector*[*r*][*s*][*t*], for each colour component. The meanings associated with the dimensions in this array are defined in Table 7-7.

Table 7-7. Meaning of indices in *PMV*[*r*][*s*][*t*], *vector*[*r*][*s*][*t*] and *vector'*[*r*][*s*][*t*]

	0	**1**
r	First motion vector in Macroblock	Second motion vector in Macroblock
s	Forward motion Vector	Backwards motion Vector
t	Horizontal Component	Vertical Component
Note:	*r* also takes the values 2 and 3 for derived motion vectors used with dual-prime prediction. Since these motion vectors are derived they do not themselves have motion vector predictors.	

7.6.3.1 Decoding the motion vectors

Each motion vector component, *vector'*[*r*][*s*][*t*], shall be calculated by any process that is equivalent to the following one. Note that the motion vector predictors shall also be updated by this process.

```
r_size = f_code[s][t] - 1
f = 1 << r_size
high = ( 16 * f ) - 1;
low = ( (-16) * f );
range = ( 32 * f );

if ( (f == 1) || (motion_code[r][s][t] == 0) )
    delta = motion_code[r][s][t] ;
else {
    delta = ( ( Abs(motion_code[r][s][t]) - 1 ) * f ) + motion_residual[r][s][t] + 1;
    if (motion_code[r][s][t] < 0)
        delta = - delta;
}

prediction = PMV[r][s][t];
if ( (mv_format == "field") && (t==1) && (picture_structure == "Frame picture") )
    prediction = PMV[r][s][t] DIV 2;

vector'[r][s][t]= prediction + delta;
if (vector'[r][s][t] < low )
    vector'[r][s][t] = vector'[r][s][t] + range;
if (vector'[r][s][t] > high)
    vector'[r][s][t] = vector'[r][s][t] - range;

if ( (mv_format == "field") && (t==1) && (picture_structure == "Frame picture") )
    PMV[r][s][t] = vector'[r][s][t] * 2;
else
    PMV[r][s][t] = vector'[r][s][t];
```

The parameters in the bitstream shall be such that the reconstructed differential motion vector, *delta*, shall lie in the range [*low*:*high*]. In addition the reconstructed motion vector, *vector'*[*r*][*s*][*t*], and the updated value of the motion vector predictor *PMV*[*r*][*s*][*t*], shall also lie in the range [*low* : *high*].

r_size, *f*, *delta*, *high* , *low* and *range* are temporary variables that are not used in the remainder of this specification.

motion_code, **motion_residual** and **mv_format** are fields recovered from the bitstream.

r, *s* and *t* specify the particular motion vector component being processed as identified in Table 7-7.

vector'[*r*][*s*][*t*] is the final reconstructed motion vector for the luminance component of the macroblock.

7.6.3.2 Motion vector restrictions

In frame pictures, the vertical component of field motion vectors shall be restricted so that they only cover half the range that is supported by the f_code that relates to those motion vectors. This restriction ensures that the motion vector predictors will always have values that are appropriate for decoding subsequent frame motion vectors. Table 7-9 summarises the size of motion vectors that may be coded as a function of f_code.

Table 7-9. Allowable motion vector range as a function of *f_code*[*s*][*t*]

***f_code*[*s*][*t*]**	**Vertical components (*t*==1) of field vectors in frame pictures**	**All other cases**
0	(forbidden)	
1	[-4: +3,5]	[-8: +7,5]
2	[-8: +7,5]	[-16: +15,5]
3	[-16: +15,5]	[-32: +31,5]
4	[-32: +31,5]	[-64: +63,5]
5	[-64: +63,5]	[-128: +127,5]
6	[-128: +127,5]	[-256: +255,5]
7	[-256: +255,5]	[-512: +511,5]
8	[-512: +511,5]	[-1024: +1023,5]
9	[-1024: +1023,5]	[-2048: +2047,5]
10-15	(reserved)	

7.6.3.3 Updating motion vector predictors

Once all of the motion vectors present in the macroblock have been decoded using the process defined in the previous clause it is sometimes necessary to update other motion vector predictors. This is because in some prediction modes fewer than the maximum possible number of motion vectors are used. The remainder of the predictors that might be used in the picture must retain “sensible” values in case they are subsequently used.

The motion vector predictors shall be updated as specified in Table 7-10 and 7-11. The rules for updating motion vector predictors in the case of skipped macroblocks are specified in 7.6.6.

Note It is possible for an implementation to optimise the updating (and resetting) of motion vector predictors depending on the picture type. For example in a P-picture the predictors for backwards motion vectors are unused and need not be maintained.

Table 7-10. Updating of motion vector predictors in frame pictures

frame_motion_-type	macroblock_motion_-forward	macroblock_motion_-backward	macroblock_-intra	Predictors to Update
Frame-based‡	-	-	1	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]◊
Frame-based	1	1	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0] *PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Frame-based	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Frame-based	0	1	0	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Frame-based‡	0	0	0	*PMV*[*r*][*s*][*t*] = 0 §
Field-based	1	1	0	(none)
Field-based	1	0	0	(none)
Field-based	0	1	0	(none)
Dual prime	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]

Note: *PMV*[*r*][*s*][1:0] = *PMV*[*u*][*v*][1:0] means that;
PMV[*r*][*s*][1] = *PMV*[*u*][*v*][1] and *PMV*[*r*][*s*][0] = *PMV*[*u*][*v*][0]

◊ If **concealment_motion_vectors** is zero then *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*).

‡ **frame_motion_type** is not present in the bitstream but is assumed to be Frame-based

§ (Only occurs in P-picture) *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*). See 7.6.3.4

Table 7-11. Updating of motion vector predictors in field pictures

field_motion_-type	macroblock_motion_-forward	macroblock_motion_-backward	macroblock_-intra	Predictors to Update
Field-based‡	-	-	1	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]◊
Field-based	1	1	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0] *PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Field-based	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Field-based	0	1	0	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Field-based‡	0	0	0	*PMV*[*r*][*s*][*t*] = 0 §
16x8 MC	1	1	0	(none)
16x8 MC	1	0	0	(none)
16x8 MC	0	1	0	(none)
Dual prime	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]

Note: *PMV*[*r*][*s*][1:0] = *PMV*[*u*][*v*][1:0] means that;
PMV[*r*][*s*][1] = *PMV*[*u*][*v*][1] and *PMV*[*r*][*s*][0] = *PMV*[*u*][*v*][0]

◊ If **concealment_motion_vectors** is zero then *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*).

‡ **field_motion_type** is not present in the bitstream but is assumed to be Field-based

§ (Only occurs in P-picture) *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*). See 7.6.3.4

7.6.3.4 Resetting motion vector predictors

All motion vector predictors shall be reset to zero in the following cases:

- At the start of each slice.
- Whenever an intra macroblock is decoded which has no concealment motion vectors.
- In a P-picture when a non-intra macroblock is decoded in which macroblock_motion_forward is zero.
- In a P-picture when a macroblock is skipped.

7.6.3.5 Prediction in P-pictures

In P-pictures, in the case that *macroblock_motion_forward* is zero and *macroblock_intra* is also zero no motion vectors are encoded for the macroblock yet a prediction must be formed. If this occurs in a P field picture the following apply;

- The prediction mode shall be “Field-based”
- The (field) motion vector shall be zero (0;0)
- The motion vector predictors shall be reset to zero
- Predictions shall be made from the field of the same parity as the field being predicted.

If this occurs in a P frame picture the following apply;

- The prediction mode shall be “Frame-based”
- The (frame) motion vector shall be zero (0;0)
- The motion vector predictors shall be reset to zero

In the case that a P field picture is used as the second field of a frame in which the first field is an I field picture a series of semantic restrictions apply. These ensure that prediction is only made from the I field picture. These restrictions are;

- There shall be no macroblocks that are coded with *macroblock_motion_forward* zero and *macroblock_intra* zero.
- Dual prime prediction shall not be used.
- Field prediction in which **motion_vertical_field_select** indicates the same parity as the field being predicted shall not be used.
- There shall be no skipped macroblocks.

7.6.3.6 Dual prime additional arithmetic

In dual prime prediction one field motion vector (*vector'*[0][0][1:0]) will have been decoded by the process already described. This represents the motion vector used to form predictions from the reference field (or reference fields in a frame picture) of the same parity as the prediction being formed. Here the word “parity” is used to differentiate the two fields. The top field has parity zero, the bottom field has parity one.

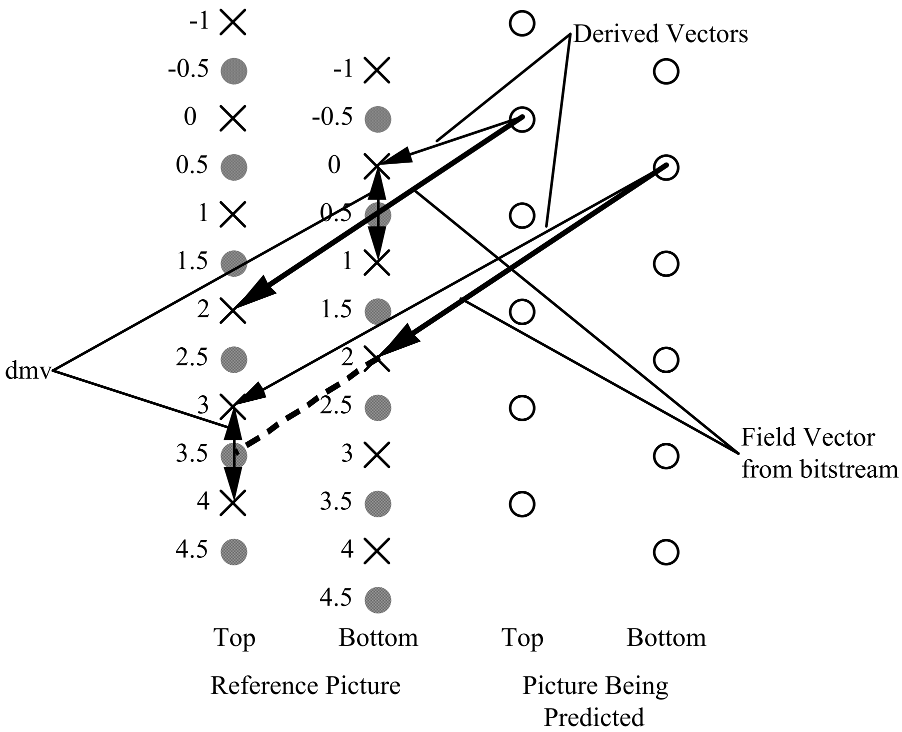


Figure 7-11. Scaling of motion vectors for dual prime prediction

In order to form a motion vector for the opposite parity (*vector'*[*r*][0][1:0]) the existing motion vector is scaled to reflect the different temporal distance between the fields. A correction is made to the vertical component (to reflect the vertical shift between the lines of top field and bottom field) and then a small differential motion vector is added. This process is illustrated in Figure 7-11 which shows the situation for a frame picture.

dmvector[0] is the horizontal component of the differential motion vector and *dmvector*[1] the vertical component. The two components of the differential motion vector shall be decoded directly using Table B-11 and shall take only one of the values -1, 0, +1.

m[*parity_ref*][*parity_pred*] is the field distance between the predicted field and the reference field as defined in Table 7-12. "*parity_ref*" is the parity of the reference field for which the new motion vector is being computed. "*parity_pred*" is the parity of the field that shall be predicted.

e[*parity_ref*][*parity_pred*] is the adjustment necessary to reflect the vertical shift between the lines of top field and bottom field as defined in Table 7-13.

Table 7-12. Definition of *m*[*parity_ref*][*parity_pred*]

picture_structure	top_field_first	m[*parity_ref*][*parity_pred*] m[1][0]	m[0][1]
11 (Frame)	1	1	3
11 (Frame)	0	3	1
01 (Top Field)	-	1	-
10 (Bottom Field)	-	-	1

Table 7-13. Definition of *e[parity_ref][parity_pred]*

parity_ref	*parity_pred*	*e[parity_ref][parity_pred]*
0	0	0
0	1	+1
1	0	-1
1	1	0

The motion vector (or motion vectors) used for predictions of opposite parity shall be computed as follows;

vector'[*r*][0][0] = ((*vector'*[0][0][0] * *m*[*parity_ref*][*parity_pred*])//2) + *dmvector*[0];
vector'[*r*][0][1] = ((*vector'*[0][0][1] * *m*[*parity_ref*][*parity_pred*])//2)
+ *e*[*parity_ref*][*parity_pred*] + *dmvector*[1];

In the case of field pictures only one such motion vector is required and here r=2. Thus the (encoded) motion vector used for the same parity prediction is *vector'*[0][0][1:0] and the motion vector used for the opposite parity prediction is *vector'*[2][0][1:0].

In the case of frame pictures two such motion vectors are required. Both fields use the encoded motion vector (*vector'*[0][0][1:0]) for predictions of the same parity. The top field shall use *vector'*[2][0][1:0] for opposite parity prediction and the bottom field shall use *vector'*[3][0][1:0] for opposite parity prediction.

7.6.3.7 Motion vectors for chrominance components

The motion vectors calculated in the previous clauses refer to the luminance component where;

vector[*r*][*s*][*t*] = *vector'*[*r*][*s*][*t*] (for all r, s and t)

For each of the two chrominance components the motion vectors shall be scaled as follows:

4:2:0 Both the horizontal and vertical components of the motion vector are scaled by dividing by two:

vector[*r*][*s*][0] = *vector'*[*r*][*s*][0] / 2;
vector[*r*][*s*][1] = *vector'*[*r*][*s*][1] / 2;

4:2:2 The horizontal component of the motion vector is scaled by dividing by two, the vertical component is not altered:

vector[*r*][*s*][0] = *vector'*[*r*][*s*][0] / 2;
vector[*r*][*s*][1] = *vector'*[r][s][1];

4:4:4 The motion vector is unmodified:

vector[*r*][*s*][0] = *vector'*[*r*][*s*][0];
vector[*r*][*s*][1] = *vector'*[*r*][*s*][1];

7.6.3.8 Semantic restrictions concerning predictions

It is a requirement on the bitstream that it shall only demand of a decoder that predictions shall be made from slices actually encoded in a reference frame or reference field. This rule applies even for skipped macroblocks and macroblocks in P-pictures in which a zero motion vector is assumed (as explained in 7.6.3.5).

Note As explained in 6.1.3 it is, in general, not necessary for the slices to cover the entire picture. However in many defined levels of defined profiles the "restricted slice structure" is used in which case the slices do cover the entire picture. In this case the semantic rule may be more simply stated; "it is a restriction on the bitstream that reconstructed motion vectors shall not refer to samples outside the boundary of the coded picture."

7.6.3.9 Concealment motion vectors

Concealment motion vectors are motion vectors that may be carried by intra macroblocks for the purpose of concealing errors should data errors preclude decoding the coefficient data. A concealment motion vector shall be present for all intra macroblocks if (and only if) concealment_motion_vectors (in the picture_coding_extension()) has the value one.

In the normal course of events no prediction shall be formed for such macroblocks (as would be expected since macroblock_intra = 1). This specification does not specify how error recovery shall be performed. However it is a recommendation that concealment motion vectors are suitable for use by a decoder that performs concealment by forming predictions as if the following flags that control the formation of predictions have the indicated values;

- In a field picture; field_motion_type = "Field-based prediction"
- In a frame picture; frame_motion_type = "Frame-based prediction"

Note If concealment is used in an I-picture then the decoder should perform prediction in a similar way to a P-picture.

Concealment motion vectors are intended for use in the case that a data error results in information being lost. There is therefore little point in encoding the concealment motion vector in the macroblock for which it is intended to be used since if the data error results in the need for error recovery it is very likely that the concealment motion vector itself would be lost or corrupted As a result the following semantic rules are appropriate.

- For all macroblocks except those in the bottom row of macroblocks concealment motion vectors should be appropriate for use in the macroblock that lies vertically below the macroblock in which the motion vector occurs.
- When the motion vector is used with respect to the macroblock identified in the previous rule a decoder must assume that the motion vector may refer to samples outside of the slices encoded in the reference frame or reference field.

  For all macroblocks in the bottom row of macroblocks the reconstructed concealment motion vectors will not be used. Therefore the motion vector (0;0) may be used to reduce unnecessary overhead.

7.6.4 Forming predictions

Predictions are formed by reading prediction samples from the reference fields or frames. A given sample is predicted by reading the corresponding sample in the reference field or frame offset by the motion vector.

A positive value of the horizontal component of a motion vector indicates that the prediction is made from samples (in the reference field/frame) that lie to the right of the samples being predicted.

A positive value of the vertical component of a motion vector indicates that the prediction is made from samples (in the reference field/frame) that lie the below the samples being predicted.

All motion vectors are specified to an accuracy of one half sample. Thus if a component of the motion vector is odd, the samples will be read from mid-way between the actual samples in the reference field/frame. These half-samples are calculated by simple linear interpolation from the actual samples.

In the case of field-based predictions it is necessary to determine which of the two available fields to use to form the prediction. In the case of dual-prime this is specified in that a motion vector is derived for both of the fields and a prediction is formed from each. In the case of field-based prediction and 16x8 MC an additional bit, motion_vertical_field_select, is encoded to indicate which field to use.

If motion_vertical_field_select is zero then the prediction is taken from the top reference field.

If motion_vertical_field_select is one then the prediction is taken from the bottom reference field.

For each prediction block the integer sample motion vectors int_vec[t] and the half sample flags half_flag[t] shall be formed as follows;

```
for (t=0; t<2; t++) {
    int_vec[t] = vector[r][s][t] DIV 2;
    if ((vector[r][s][t] - (2 * int_vec[t]) != 0)
        half_flag[t] = 1;
    else
        half_flag[t] = 0;
}
```

Then for each sample in the prediction block the samples are read and the half sample prediction applied as follows;

```
    if ( (! half_flag[0] )&& (! half_flag[1]) )
        pel_pred[y][x] = pel_ref[y + int_vec[1]][x + int_vec[0]] ;

    if ( (! half_flag[0] )&& half_flag[1] )
        pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
                           pel_ref[y + int_vec[1]+1][x + int_vec[0]] ) // 2;

    if ( half_flag[0]&& (! half_flag[1]) )
        pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
                           pel_ref[y + int_vec[1]][x + int_vec[0]+1] ) // 2;

    if ( half_flag[0]&& half_flag[1] )
        pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
                           pel_ref[y + int_vec[1]][x + int_vec[0]+1] +
                             pel_ref[y + int_vec[1]+1][x + int_vec[0]] +
                               pel_ref[y + int_vec[1]+1][x + int_vec[0]+1] ) // 4;
```

where *pel_pred*[*y*][*x*] is the prediction sample being formed and *pel_ref*[*y*][*x*] are samples in the reference field or frame.

7.6.5 Motion vector selection

Table 7-14 shows the prediction modes used in field pictures and Table 7-15 shows the predictions used in frame pictures. In each table the motion vectors that are present in the bitstream are listed in the order in which they appear in the bitstream.

Table 7-14. Predictions and motion vectors in field pictures

field_ motion_ type	**macroblock_motion_- forward**	**macroblock_motion_- backward**	**macro- block_- intra**	**Motion vector**	**Prediction formed for**
Field-based‡	-	-	1	*vector'*[0][0][1:0]◊	None (motion vector is for concealment)
Field-based	1	1	0	*vector'*[0][0][1:0] *vector'*[0][1][1:0]	whole field, forward whole field, backward
Field-based	1	0	0	*vector'*[0][0][1:0]	whole field, forward
Field-based	0	1	0	*vector'*[0][1][1:0]	whole field, backward
Field-based‡	0	0	0	*vector'*[0][0][1:0]*§	whole field, forward
16x8 MC	1	1	0	*vector'*[0][0][1:0] *vector'*[1][0][1:0] *vector'*[0][1][1:0] *vector'*[1][1][1:0]	upper 16x8 field, forward lower 16x8 field, forward upper 16x8 field, backward lower 16x8 field, backward
16x8 MC	1	0	0	*vector'*[0][0][1:0] *vector'*[1][0][1:0]	upper 16x8 field, forward lower 16x8 field, forward
16x8 MC	0	1	0	*vector'*[0][1][1:0] *vector'*[1][1][1:0]	upper 16x8 field, backward lower 16x8 field, backward
Dual prime	1	0	0	*vector'*[0][0][1:0] *vector'*[2][0][1:0]*†	whole field, from same parity, forward whole field, from opposite parity, forward

Note: Motion vectors are listed in the order they appear in the bitstream

◊ the motion vector is only present if **concealment_motion_vectors** is one

‡ **field_motion_type** is not present in the bitstream but is assumed to be Field-based

* These motion vectors are not present in the bitstream

† These motion vectors are derived from *vector'*[0][0][1:0] as described in 7.6.3.6

§ The motion vector is taken to be (zero, zero) as explained in 7.6.3.5.

Table 7-15. Predictions and motion vectors in frame pictures

frame_-motion_-type	**macroblock_motion_-** **forward**	**macroblock_motion_-** **backward**	**macro-block_-intra**	**Motion vector**	**Prediction formed for**
Frame-based‡	-	-	1	*vector'*[0][0][1:0]◊	None (motion vector is for concealment)
Frame-based	1	1	0	*vector'*[0][0][1:0] *vector'*[0][1][1:0]	frame, forward frame, backward
Frame-based	1	0	0	*vector'*[0][0][1:0]	frame, forward
Frame-based	0	1	0	*vector'*[0][1][1:0]	frame, backward
Frame-based‡	0	0	0	*vector'*[0][0][1:0]*§	frame, forward
Field-based	1	1	0	*vector'*[0][0][1:0] *vector'*[1][0][1:0] *vector'*[0][1][1:0] *vector'*[1][1][1:0]	top field, forward bottom field, forward top field, backward bottom field, backward
Field-based	1	0	0	*vector'*[0][0][1:0] *vector'*[1][0][1:0]	top field, forward bottom field, forward
Field-based	0	1	0	*vector'*[0][1][1:0] *vector'*[1][1][1:0]	top field, backward bottom field, backward
Dual prime	1	0	0	*vector'*[0][0][1:0] *vector'*[0][0][1:0] *vector'*[2][0][1:0]*† *vector'*[3][0][1:0]*†	top field, from same parity, forward bottom field, from same parity, forward top field, from opposite parity, forward bottom field, from opposite parity, forward

Note: Motion vectors are listed in the order they appear in the bitstream

◊ the motion vector is only present if **concealment_motion_vectors** is one

‡ **frame_motion_type** is not present in the bitstream but is assumed to be Frame-based

* These motion vectors are not present in the bitstream

† These motion vectors are derived from *vector'*[0][0][1:0] as described in 7.6.3.6

§ The motion vector is taken to be (zero, zero) as explained in 7.6.3.5

7.6.6 Skipped Macroblocks

In skipped macroblocks (where macroblock_address_increment is greater than 1) the decoder has neither DCT coefficient information nor motion vector information. The decoder shall form a prediction for such macroblocks which shall then be used as the final decoded sample values.

The handling of skipped macroblocks is different between P-pictures and B-pictures. In addition the process differs between field pictures and frame pictures.

There shall be no skipped macroblocks in I-pictures except when;

either picture_spatial_scalable_extension() follows the picture_header() of the current picture.

or sequence_scalable_extension() is present in the bitstream and scalable_mode = “SNR scalability”.

7.6.6.1 P field picture

- The prediction shall be made as if field_motion_type is “Field-based”
- The prediction shall be made from the field of the same parity as the field being predicted.
- Motion vector predictors shall be reset to zero.
- The motion vector shall be zero.

7.6.6.2 P frame picture

- The prediction shall be made as if frame_motion_type is “Frame-based”
- Motion vector predictors shall be reset to zero.
- The motion vector shall be zero.

7.6.6.3 B field picture

- The prediction shall be made as if field_motion_type is “Field-based”
- The prediction shall be made from the field of the same parity as the field being predicted.
- The direction of the prediction forward/backward/bi-directional shall be the same as the previous macroblock.
- Motion vector predictors are unaffected.
- The motion vectors are taken from the appropriate motion vector predictors. Scaling of the motion vectors for colour components shall be performed as described in 7.6.3.7.

7.6.6.4 B frame picture

- The prediction shall be made as if frame_motion_type is “Frame-based”
- The direction of the prediction forward/backward/bi-directional shall be the same as the previous macroblock.
- Motion vector predictors are unaffected.
- The motion vectors are taken directly from the appropriate motion vector predictors. Scaling of the motion vectors for colour components shall be performed as described in 7.6.3.7.

7.6.7 Combining predictions

The final stage is to combine the various predictions together in order to form the final prediction blocks.

It is also necessary to organise the data into blocks that are either field organised or frame organised in order to be added directly to the decoded coefficients.

The transform data is either field organised or frame organised as specified by *dct_type*.

7.6.7.1 Simple frame predictions

In the case of simple frame predictions the only further processing that may be required is to average forward and backward predictions in B-pictures. If *pel_pred_forward*[*y*][*x*] is the forwards prediction sample and *pel_pred_backward*[*y*][*x*] is the corresponding backward prediction then the final prediction sample shall be formed as;

pel_pred[*y*][*x*] = (*pel_pred_forward*[*y*][*x*] + *pel_pred_backward*[*y*][*x*])//2;

The predictions for chrominance components of 4:2:0, 4:2:2 and 4:4:4 formats shall be of size 8 samples by 8 lines, 8 samples by 16 lines and 16 samples by 16 lines respectively.

7.6.7.2 Simple field predictions

In the case of simple field predictions (i.e. neither 16x8 or dual prime) the only further processing that may be required is to average forward and backward predictions in B-pictures. This shall be performed as specified for “Frame predictions” in the previous clause.

The predictions for chrominance components of 4:2:0, 4:2:2 and 4:4:4 formats for each field shall be of size 8 samples by 4 lines, 8 samples by 8 lines and 16 samples by 8 lines respectively.

7.6.7.3 16x8 Motion compensation

In this prediction mode separate predictions are formed for the upper 16x8 region of the macroblock and the lower 16x8 region of the macroblock.

The predictions for chrominance components, for each 16x8 region, of 4:2:0, 4:2:2 and 4:4:4 formats shall be of size 8 samples by 4 lines, 8 samples by 8 lines and 16 samples by 8 lines respectively.

7.6.7.4 Dual prime

In dual prime mode two predictions are formed for each field in an analogous manner to the backward and forward predictions in B-pictures. If *pel_pred_same_parity*[*y*][*x*] is the prediction sample from the same parity field and *pel_pred_opposite_parity*[*y*][*x*] is the corresponding sample from the opposite prediction then the final prediction sample shall be formed as;

pel_pred[*y*][*x*] = (*pel_pred_same_parity*[*y*][*x*] + *pel_pred_opposite_parity*[*y*][*x*])//2;

In the case of dual prime prediction in a frame picture, the predictions for chrominance components of each field of 4:2:0, 4:2:2 and 4:4:4 formats shall be of size 8 samples by 4 lines, 8 samples by 8 lines and 16 samples by 8 lines respectively.

In the case of dual prime prediction in a field picture, the predictions for chrominance components of 4:2:0, 4:2:2 and 4:4:4 formats shall be of size 8 samples by 8 lines, 8 samples by 16 lines and 16 samples by 16 lines respectively.

7.6.8 Adding prediction and coefficient data

The prediction blocks have been formed and reorganised into blocks of prediction samples *p*[*y*][*x*] which match the field/frame structure used by the transform data blocks.

The transform data *f*[*y*][*x*] shall be added to the prediction data and saturated to form the final decoded samples *d*[*y*][*x*] as follows;

```
for (y=0; y<8; y++) {
    for (x=0; x<8; x++) {
        d[y][x] = f[y][x]+p[y][x];
        if (d[y][x] < 0) d[y][x] = 0;
        if (d[y][x] > 255) d[y][x] = 255;
    }
}
```

7.7 Spatial scalability

This clause specifies the additional decoding process required for the spatial scalable extensions.

Both the lower layer and the enhancement layer shall use the “restricted slice structure” (no gaps between slices).

Figure 7-12 is a diagram of the video decoding process with spatial scalability The diagram is simplified for clarity.

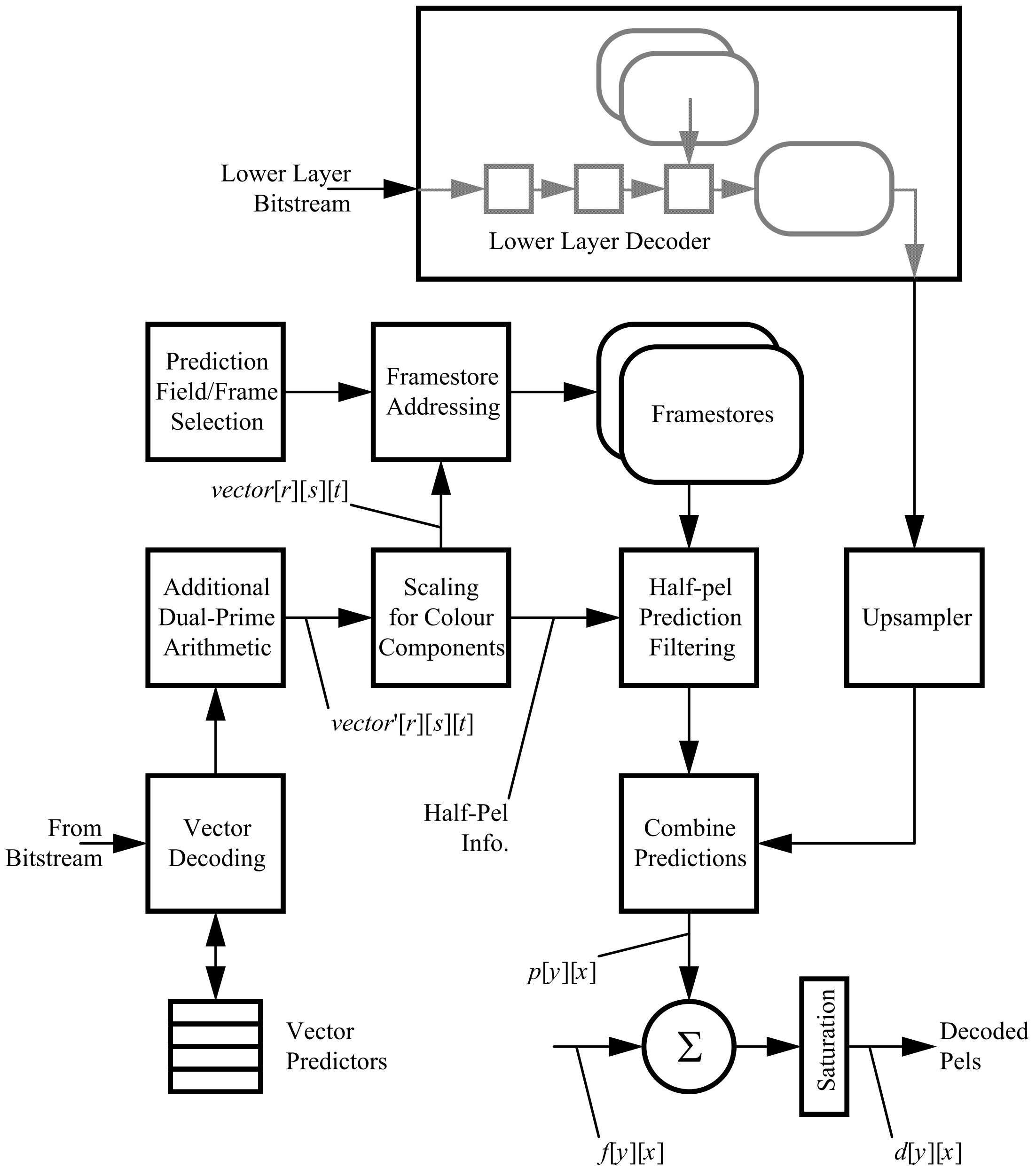


Figure 7-12. Simplified motion compensation process for spatial scalability

7.7.1 Higher syntactic structures

In general the base layer of a spatial scalable hierarchy can conform to any coding standard including ITU-T Rec. H.261, ISO/IEC11172-2 and ITU-T Rec. xxxx | ISO/IEC13818-2. Note however, that within this specification the decodability of a spatial scalable hierarchy is only considered in the case that the base layer conforms to this specification or ISO/IEC11172-2.

Due to the "loose coupling" of layers only one syntactic restriction is needed in the enhancement layer if both lower and enhancement layer are interlaced. In that case picture_structure has to take the same value as in the reference frame used for prediction from the lower layer. See 7.7.3.1 for how to identify this reference frame.

7.7.2 Prediction in the enhancement layer

A motion compensated 'temporal' prediction is made from previously decoded pictures in the enhancement layer as described in 7.6. In addition, a 'spatial' prediction is formed, which is an upsampled version of a lower layer decoded frame, as described in 7.7.3. These predictions are selected individually or combined to form the actual prediction.

In general up to four separate predictions are formed for each macroblock which are combined together to form the final prediction macroblock p[y][x].

In the case that a macroblock is not coded, either because the entire macroblock is skipped or the specific macroblock is not coded there is no coefficient data. In this case f[y][x] is zero and the decoded samples are simply the prediction, p[y][x].

7.7.3 Formation of spatial prediction

Forming the spatial prediction requires identification of the correct reference frame and definition of the spatial resampling process, which is done in the following clauses.

The resampling process is defined for a whole frame, however, for decoding of a macroblock, only the 16x16 region in the upsampled frame, which corresponds to the position of this macroblock, is needed.

7.7.3.1 Selection of reference frame

The spatial prediction is made from the decoded frame of the lower layer referenced by the lower_layer_temporal_reference. However, if base and enhancement layer bitstreams are embedded in an ITU-T Rec. xxxx | ISO/IEC13818-1 (Systems) multiplex, this information is overriden by the timing information given by the decoding time stamps (DTS) in the PES headers.

Note: If group_of_pictures_header() occurs often in the lower layer bitstream then the temporal reference in the lower layer may be ambiguous (because temporal_reference is reset after a group_of_pictures_header()).

The picture from which the spatial prediction is made shall be the coincident or most recently decoded lower layer picture as indicated by the DTS values.

Note furthermore that spatial scalability will only work efficiently when predictions are formed from frames in the lower layer which are also coincident (or very close) in display time with the predicted frame in the enhancement layer.

7.7.3.2 Resampling process

The spatial prediction is made by resampling the lower layer frame to the same sample grid as the enhancement layer. This grid is defined in terms of frame coordinates, even if a lower-layer interlaced frame was actually coded as a field structured frame.

This resampling process is illustrated in Figure 7-13.

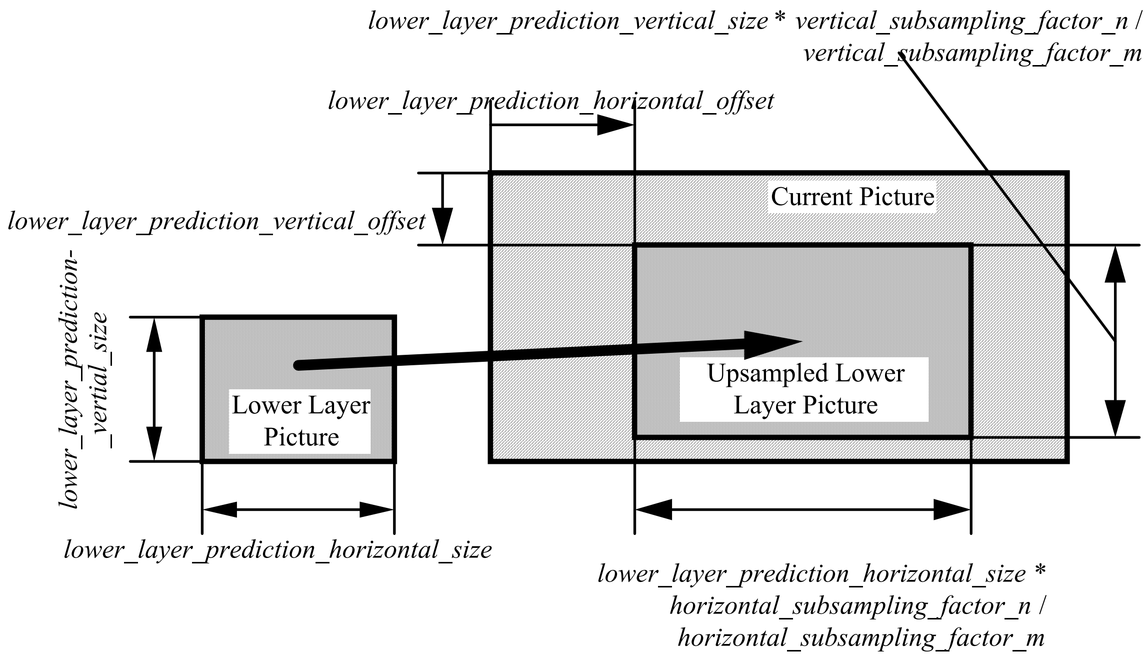


Figure 7-13. Formation of the 'spatial' prediction by interpolation of the lower layer picture

Spatial predictions shall only be made for macroblocks in the enhancement layer that lie wholly within the upsampled lower layer reconstructed frame.

The upsampling process depends on whether the lower layer reconstructed frame is interlaced or progressive, as indicated by lower_layer_progressive_frame and whether the enhancement layer frame is interlaced or progressive, as indicated by progressive_frame.

When lower_layer_progressive_frame is "1", the lower layer reconstructed frame (renamed to input_prog_field) is resampled vertically as described in 7.7.3.5. The resulting frame is considered to be progressive if progressive_frame is "1" and interlaced if progressive_frame is "0". The resulting frame is resampled horizontally as described in 7.7.3.6. lower_layer_deinterlaced_field_select shall have the value "1".

When lower_layer_progressive_frame is "0" and progressive_frame is "0", each lower layer reconstructued field is deinterlaced as described in 7.7.3.4, to produce a progressive field (input_prog_field). This field is resampled vertically, taking into account the lower_layer_vertical_offset, as described in 7.7.3.5. The resulting field is resampled horizontally as described in 7.7.3.6. Finally the resulting field is subsampled, again taking into account the lower_layer_vertical_offset, to produce an interlaced field. lower_layer_deinterlaced_field_select shall have the value "1".

When lower_layer_progressive_frame is "0" and progressive_frame is "1", each lower layer reconstructed field is deinterlaced as described in 7.7.3.4, to produce a progressive field (input_prog_field). Only one of these fields is required. When lower_layer_deinterlaced_field_select is "0" the top field is used, otherwise the bottom field is used. The one that is used is resampled vertically as described in 7.7.3.5. The resulting frame is resampled horizontally as described in 7.7.3.6.

For interlaced frames, if the current (and implicitly the lower-layer) frame are encoded as field pictures, the deinterlacing process described in 7.7.3.5 is done within the field.

The lower layer offsets are limited to even values when the chrominance in the enhancement layer is subsampled in that dimension in order to align the chrominance samples between the two layers.

The upsampling process is summarised Table 7-16.

Table 7-16 Upsampling process

lower_layer_ deinterlaced_ field_select	lower_layer_ progressive_frame	progressive_ frame	Apply deinterlace process	Entity used for prediction
0	0	1	yes	top field
1	0	1	yes	bottom field
1	1	1	no	frame
1	1	0	no	frame
1	0	0	yes	both fields

7.7.3.3 Colour component processing

Due to the different sampling grids of luminance and chrominance components, some variables used in 7.7.3.4 to 7.7.3.6 take different values for luminance and chrominance resampling. Furthermore it is permissible for the chrominance formats in the lower layer and the enhancement layer to be different from one another.

The following table defines the values for the variables used in 7.7.3.4 to 7.7.3.6

Table 7-17 Local variables used in 7.7.3.3 to 7.7.3.5

variable	value for luminance processing	value for chrominance processing
ll_h_size	lower_layer_prediction_horizontal_size	lower_layer_prediction_horizontal_size / chroma_ratio_horizontal
ll_v_size	lower_layer_prediction_vertical_size	lower_layer_prediction_vertical_size / chroma_ratio_vertical
ll_h_offset	lower_layer_horizontal_offset	lower_layer_horizontal_offset / chroma_ratio_horizontal
ll_v_offset	lower_layer_vertical_offset	lower_layer_vertical_offset / chroma_ratio_vertical
h_subs_m	horizontal_subsampling_factor_m	horizontal_subsampling_factor_m * format_ratio_horizontal
h_subs_n	horizontal_subsampling_factor_n	horizontal_subsampling_factor_n
v_subs_m	vertical_subsampling_factor_m	vertical_subsampling_factor_m * format_ratio_vertical
v_subs_n	vertical_subsampling_factor_n	vertical_subsampling_factor_n

Tables 7-17a and 7-17b give additional definitions.

Table 7-17a chrominance subsampling ratios

chrominance format lower layer	chroma_ratio_ horizontal	chroma_ratio_ vertical
4:2:0	2	2
4:2:2	2	1
4:4:4	1	1

Table 7-17b chrominance format ratios

chrominance format lower layer	chrominance format enhancement layer	format_ratio_ horizontal	format_ratio_ vertical
4:2:0	4:2:0	1	1
4:2:0	4:2:2	1	2
4:2:0	4:4:4	2	2
4:2:2	4:2:2	1	1
4:2:2	4:4:4	2	1
4:4:4	4:4:4	1	1

7.7.3.4 Deinterlacing

First, each lower layer field is padded with zeros to form a progressive grid at a frame rate equal to the field rate of the lower layer, and with the same number of lines and samples per line as the lower layer frame. Table 7-18 specifies the filters to be applied next. The luminance component is filtered using the relevant two field aperture filter if picture_structure == "Frame-Picture" or else using the one field aperture filter . The chrominance component is filtered using the one field aperture filter.

The temporal and vertical columns of the table indicate the relative spatial and temporal coordinates of the samples to which the filter taps defined in the other two columns apply. An intermediate sum is formed by adding the multiplied coefficients together.

Table 7-18. Deinterlacing Filter

		two field aperture		one field aperture
Temporal	**Vertical**	**Filter for first field**	**Filter for second field**	**Filter (both fields)**
-1	-2	0	-1	0
-1	0	0	2	0
-1	2	0	-1	0
0	-1	8	8	8
0	0	16	16	16
0	1	8	8	8
1	-2	-1	0	0
1	0	2	0	0
1	+2	-1	0	0

The output of the filter (sum) is then scaled according to the following formula:

$$prog\_field[y][x] = sum // 16$$

and saturated to lie in the range [0:255].

The filter aperture can extend outside the coded picture size. In this case the samples of the lines outside the active picture shall take the value of the closest neighbouring existing sample (below or above) of the same field as defined below.

For all samples [y][x]:

```
if (y<0 && (y&1 == 1))
      y=1
if (y<0 && (y&1 == 0))
      y=0
if (y >= ll_v_size &&
                  ( (y-ll_v_size)&1 == 1))
      y = ll_v_size - 1
if (y >= ll_v_size &&
                  ((y-ll_v_size)&1 == 0))
      y = ll_v_size - 2
```

7.7.3.5 Vertical resampling

The frame subject to vertical resampling, input_prog_field, is resampled to the enhancement layer vertical sampling grid using linear interpolation between the sample sites according to the following formula, where mid_field is the resulting field:

$$\text{mid\_field}[y_h + \text{ll\_v\_offset}][x] = (16 - \text{phase})*\text{input\_prog\_field}[y1][x] + \text{phase} * \text{input\_prog\_field}[y2][x]$$

where y_h+ ll_v_offset = output sample co-ordinate in mid_field

y1 = $(y_h * \text{v\_subs\_m}) / \text{v\_subs\_n}$

y2 = y1 + 1 if y1 < ll_v_size - 1

y1 otherwise

phase = $(16 * ((y_h * \text{v\_subs\_m}) \% \text{v\_subs\_n}))$ // v_subs_n

Samples which lie outside the lower layer picture which are required for upsampling are obtained by border extension of the lower layer picture.

Note: The calculation of phase assumes that the sample position in the enhancement layer at $y_h = 0$ is spatially coincident with the first sample position of the lower layer. It is recognised that this is an approximation for the chrominance component if the chroma_format == 4:2:0.

7.7.3.6 Horizontal resampling

The frame subject to horizontal resampling, mid_field, is resampled to the enhancement layer horizontal sampling grid using linear interpolation between the sample sites according to the following formula, where output_field is the resulting field:

output_field[y][x_h + ll_h_offset] = ((16 - phase)*mid_field[y][x1] + phase * mid_field[y][x2]) // 256

where x_h+ ll_h_offset = output sample co-ordinate in output_field

x1 = $(x_h * \text{h\_subs\_m}) / \text{h\_subs\_n}$

x2 = x1 + 1 if x1 < ll_h_size - 1

x1 otherwise

phase = $(16 * ((x_h * \text{h\_subs\_m}) \% \text{h\_subs\_n}))$ // h_subs_n

Samples which lie outside the lower layer picture which are required for upsampling are obtained by border extension of the lower layer picture.

7.7.4 Selection and combination of spatial and temporal predictions

The spatial and temporal predictions can be selected or combined to form the actual prediction. The macroblock_type (Tables B-5, B-6 and B-7) indicates, by use of the spatial_temporal_weight_class, which can take the values 0, 1, 2, 3, 4, whether the prediction is temporal-only, spatial-only or a weighted combination of temporal and spatial predictions. A fuller description of spatial_temporal_weight_class is given in 7.7.5.

In intra pictures, if spatial_temporal_weight_class is 0, normal intra coding is performed, otherwise the prediction is spatial-only. In predicted and interpolated pictures, if the spatial_temporal_weight_class is 0, prediction is temporal-only, if the spatial_temporal_weight_class is 4, prediction is spatial-only, otherwise one or a pair of prediction weights is used to combine the spatial and temporal predictions.

The possible spatial_temporal_weights are given in a weight table which is selected in the picture scalable extension. Up to four different weight tables are available for use depending on whether the current and lower layers are interlaced or progressive. In macroblock_modes(), a two bit code, spatial_temporal_weight_code, is used to describe the prediction for each field (or frame), as shown in the Table 7-19. In this table spatial_temporal_integer_weight identifies those spatial_temporal_weight_codes that can also be used with dual prime prediction (see tables 7-21, 7-23).

Table 7-19 spatial_temporal_weights and spatial_temporal_weight_classes for the spatial_temporal_weight_code_table_index and spatial_temporal_weight_codes

spatial_temporal_ weight_code_table_ index	spatial_ temporal_ weight_code	spatial_ temporal_ weight (s)	spatial_ temporal_ weight class	spatial_ temporal_ integer_weight
00*	-	(0,5)	1	0
01	00	(0; 1)	3	1
	01	(0; 0,5)	1	0
	10	(0,5; 1)	3	0
	11	(0,5; 0,5)	1	0
10	00	(1; 0)	2	1
	01	(0,5; 0)	1	0
	10	(1; 0,5)	2	0
	11	(0,5; 0,5)	1	0
11	00	(1; 0)	2	1
	01	(1; 0,5)	2	0
	10	(0,5; 1)	3	0
	11	(0,5; 0,5)	1	0

* For spatial_temporal_weight_code_table_index == 0 no spatial_temporal_weight_code is transmitted.

Note: Spatial-only prediction (weight_class == 4) is signalled by different values of macroblock_type (see tables B-5 to B-7).

When the spatial_temporal_weight combination is given in the form (a; b), "a" gives the proportion of the prediction for the top field which is derived from the spatial prediction and "b" gives the proportion of the prediction for the bottom field which is derived from the spatial prediction for that field.

When the spatial_temporal_weight is given in the form (a), "a" gives the proportion of the prediction for the picture which is derived from the spatial prediction for that picture.

The precise method for predictor calculation is as follows:

If pel_pred_temp[y][x] is used to denote the temporal prediction (formed within the enhancement layer) as defined for pel_pred[y][x] in 7.6. pel_pred_lower[y][x] is used to denote the prediction formed from the lower layer, then:

If the spatial_temporal_weight is zero then no prediction is made from the lower layer. Therefore;

pel_pred[y][x] = pel_pred_temp[y][x];

If the spatial_temporal_weight is one then no prediction is made from the enhancement layer. Therefore;

pel_pred[y][x] = pel_pred_lower[y][x];

If the weight is one half then the prediction is the average of the temporal and spatial predictions. Therefore;

pel_pred[y][x] = (pel_pred_temp[y][x] + pel_pred_lower[y][x])//2;

When chroma_format == 4:2:0, and chroma_420_type == 0, chrominance is treated as interlaced, that is, the first weight is used for the top field chrominance lines and the second weight is used for the bottom field chrominance lines.

It is intended that the different weight code tables are used in the following circumstances (all other allowed values are given in brackets)

Table 7-20. Intended (allowed) spatial_temporal_weight_code_table_index values

Lower layer format	Enhancement layer format	spatial_temporal_weight_code_table_index
Progressive or interlaced	Progressive	00
Progressive coincident with enhancement layer top fields	Interlaced	10 (00,01,11)
Progressive coincident with enhancement layer from bottom fields	Interlaced	01 (00,10,11)
Interlaced (picture_structure == Frame-Picture)	Interlaced	00 or 11 (01, 10)
Interlaced (picture_structure != Frame-Picture)	Interlaced	00

7.7.5 Updating motion vector predictors and Motion vector selection

In frame pictures where field prediction is used the possibility exists that one of the fields is predicted using spatial-only prediction. In this case no motion vector is present in the bitstream for the field which has spatial-only prediction. For the case where both fields of a frame have spatial-only prediction, the macroblock_type is such that no motion vectors are present in the bitstream for that macroblock.

The class also indicates the number of motion vectors which are present in the coded bitstream and how the prediction vectors are updated as defined in Table 7-21 and Table 7-22. Classes are defined in the following way:

Class 0 indicates temporal-only prediction

Class 1 indicates that neither field has spatial-only prediction

Class 2 indicates that the top field is spatial-only prediction

Class 3 indicates that the bottom field is spatial-only prediction

Class 4 indicates spatial-only prediction.

Table 7-21. Updating of motion vector predictors in Field Pictures

frame_motion_type	**macroblock_motion_forward**	**macroblock_motion_backward**	**macroblock_intra**	**spatial_temporal_weight_class**	**Predictors to update**
Field-based‡	-	-	1	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]◊
Field-based	1	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0] *PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Field-based	1	0	0	0,1	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Field-based	0	1	0	0,1	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Field-based‡	0	0	0	0,1,4	*PMV*[*r*][*s*][*t*] = 0 §
16x8 MC	1	1	0	0	(none)
16x8 MC	1	0	0	0,1	(none)
16x8 MC	0	1	0	0,1	(none)
Dual prime	1	0	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]

Note: *PMV*[*r*][*s*][1:0] = *PMV*[*u*][*v*][1:0] means that;

PMV[*r*][*s*][1] = *PMV*[*u*][*v*][1] and *PMV*[*r*][*s*][0] = *PMV*[*u*][*v*][0]

◊ If **concealment_motion_vectors** is zero then *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*).

‡ **field_motion_type** is not present in the bitstream but is assumed to be Field-based

§ *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*). See 7.6.3.4.

Table 7-22. Updating of motion vector predictors in Frame Pictures

frame_motion_type	macroblock_motion_forward	macroblock_motion_backward	macroblock_intra	spatial_temporal_weight_class	Predictors to update
Frame-based‡	-	-	1	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]◊
Frame-based	1	1	0	0	*PMV*[1][0][1:0] = *PMV*[0][0][1:0] *PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Frame-based	1	0	0	0,1,2,3	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Frame-based	0	1	0	0,1,2,3	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Frame-based‡	0	0	0	0,1,2,3,4	*PMV*[r][s][t] = 0§
Field-based	1	1	0	0	(none)
Field-based	1	0	0	0,1	(none)
Field-based	1	0	0	2	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Field-based	1	0	0	3	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]
Field-based	0	1	0	0,1	(none)
Field-based	0	1	0	2	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Field-based	0	1	0	3	*PMV*[1][1][1:0] = *PMV*[0][1][1:0]
Dual prime@	1	0	0	0,2,3	*PMV*[1][0][1:0] = *PMV*[0][0][1:0]

Note: *PMV*[*r*][*s*][1:0] = *PMV*[*u*][*v*][1:0] means that;

PMV[*r*][*s*][1] = *PMV*[*u*][*v*][1] and *PMV*[*r*][*s*][0] = *PMV*[*u*][*v*][0]

◊ If **concealment_motion_vectors** is zero then *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*).

‡ **frame_motion_type** is not present in the bitstream but is assumed to be Frame-based

§ *PMV*[*r*][*s*][*t*] is set to zero (for all *r*, *s* and *t*). See 7.6.3.4.

@ Dual prime can not be used when spatial_temporal_integer_weight = "0".

7.7.5.1 Resetting motion vector predictors

In addition to the cases identified in 7.6.3.4 the motion vector predictors shall be reset in the following cases;

- In a P-picture when a macroblock is purely spatially predicted (spatial_temporal_weight_class == 4)
- In a B-picture when a macroblock is purely spatially predicted (spatial_temporal_weight_class == 4)

Note: In case of spatial_temporal_weight_class == 2 in a frame picture when field-based prediction is used, the transmitted vector is applied for the *bottom* field (see Table 7-24). However this vector[0][s][1:0] is predicted from PMV[0][s][1:0] . PMV[1][s][1:0] is then updated as shown in Table 7-22.

Table 7-23. Predictions and motion vectors in field pictures

field_motion_type	**macroblock_motion_forward**	**macroblock_motion_backward**	**macroblock_intra**	**spatial_temporal_weight_class**	**Motion vector**	**Prediction formed for**
Field-based‡	-	-	1	0	*vector'*[0][0][1:0]◊	None (motion vector is for concealment)
Field-based	1	1	0	0	*vector'*[0][0][1:0]	whole field, forward
					vector'[0][1][1:0]	whole field, backward
Field-based	1	0	0	0,1	*vector'*[0][0][1:0]	whole field, forward
Field-based	0	1	0	0,1	*vector'*[0][1][1:0]	whole field, backward
Field-based‡	0	0	0	0,1,4	*vector'*[0][0][1:0]*§	whole field, forward
16x8 MC	1	1	0	0	*vector'*[0][0][1:0]	upper 16x8 field, forward
					vector'[1][0][1:0]	lower 16x8 field, forward
					vector'[0][1][1:0]	upper 16x8 field, backward
					vector'[1][1][1:0]	lower 16x8 field, backward
16x8 MC	1	0	0	0,1	*vector'*[0][0][1:0]	upper 16x8 field, forward
					vector'[1][0][1:0]	lower 16x8 field, forward
16x8 MC	0	1	0	0,1	*vector'*[0][1][1:0]	upper 16x8 field, backward
					vector'[1][1][1:0]	lower 16x8 field, backward
Dual prime	1	0	0	0	*vector'*[0][0][1:0]	whole field, same parity, forward
					vector'[2][0][1:0]*†	whole field, opposite parity, forward

Note: Motion vectors are listed in the order they appear in the bitstream

◊ the motion vector is only present if **concealment_motion_vectors** is one

‡ **field_motion_type** is not present in the bitstream but is assumed to be Field-based

* These motion vectors are not present in the bitstream

† These motion vectors are derived from *vector'*[0][0][1:0] as described in 7.6.3.6

§ The motion vector is taken to be (zero, zero) as explained in 7.6.3.5

1 **Table 7-24. Predictions and motion vectors in frame pictures**

frame_motion_type	macroblock_motion_forward	macroblock_motion_backward	macroblock_intra	spatial_temporal_weight_class	Motion vector	Prediction formed for
Frame-based‡	-	-	1	0	*vector′*[0][0][1:0]◊	None (motion vector is for concealment)
Frame-based	1	1	0	0	*vector′*[0][0][1:0]	frame, forward
					vector′[0][1][1:0]	frame, backward
Frame-based	1	0	0	0,1,2,3	*vector′*[0][0][1:0]	frame, forward
Frame-based	0	1	0	0,1,2,3	*vector′*[0][1][1:0]	frame, backward
Frame-based‡	0	0	0	0,1,2,3,4	*vector′*[0][0][1:0]*§	frame, forward
Field-based	1	1	0	0	*vector′*[0][0][1:0]	top field, forward
					vector′[1][0][1:0]	bottom field, forward
					vector′[0][1][1:0]	top field, backward
					vector′[1][1][1:0]	bottom field, backward
Field-based	1	0	0	0,1	*vector′*[0][0][1:0]	top field, forward
					vector′[1][0][1:0]	bottom field, forward
Field-based	1	0	0	2		top field, spatial
					vector′[0][0][1:0]	bottom field, forward
Field-based	1	0	0	3	*vector′*[0][0][1:0]	top field, forward
						bottom field, spatial
Field-based	0	1	0	0,1	*vector′*[0][1][1:0]	top field, backward
					vector′[1][1][1:0]	bottom field, backward
Field-based	0	1	0	2		top field, spatial
					vector′[0][1][1:0]	bottom field, backward
Field-based	0	1	0	3	*vector′*[0][1][1:0]	top field, backward
						bottom field, spatial
Dual prime@	1	0	0	0,2,3	*vector′*[0][0][1:0]	top field, same parity, forward
					vector′[0][0][1:0]*	bottom field, same parity, forward
					vector′[2][0][1:0]*†	top field, opposite parity, forward
					vector′[3][0][1:0]*†	bottom fld., opposite parity, forward

Note: Motion vectors are listed in the order they appear in the bitstream

◊ the motion vector is only present if **concealment_motion_vectors** is one

‡ **frame_motion_type** is not present in the bitstream but is assumed to be Frame-based

* These motion vectors are not present in the bitstream

† These motion vectors are derived from *vector′*[0][0][1:0] as described in 7.6.3.6

§ The motion vector is taken to be (zero, zero) as explained in 7.6.3.5

@ Dual prime can not be used when spatial_temporal_integer_weight = "0".

7.7.6 Skipped macroblocks

In all cases, a skipped macroblock is the result of a prediction only, and all the DCT coefficients are considered to be zero.

If sequence_scalable_extension is present and scalable_mode = “spatial scalability”, the following rules apply in addition to those given in 7.6.6.

In I-pictures, skipped macroblocks are allowed. These are defined as spatial-only predicted.

In P-pictures and B-pictures, the skipped macroblock is temporal-only predicted.

In B-pictures a skipped macroblock shall not follow a spatial-only predicted macroblock.

7.7.7 VBV buffer underflow in the lower layer

In the case of spatial scalability, VBV buffer underflow in the lower layer may cause problems. This is because of possible uncertainty in precisely which frames will be repeated by a particular decoder.

Hence in the lower layer bitstream VBV buffer underflow shall not occur.

7.8 SNR scalability

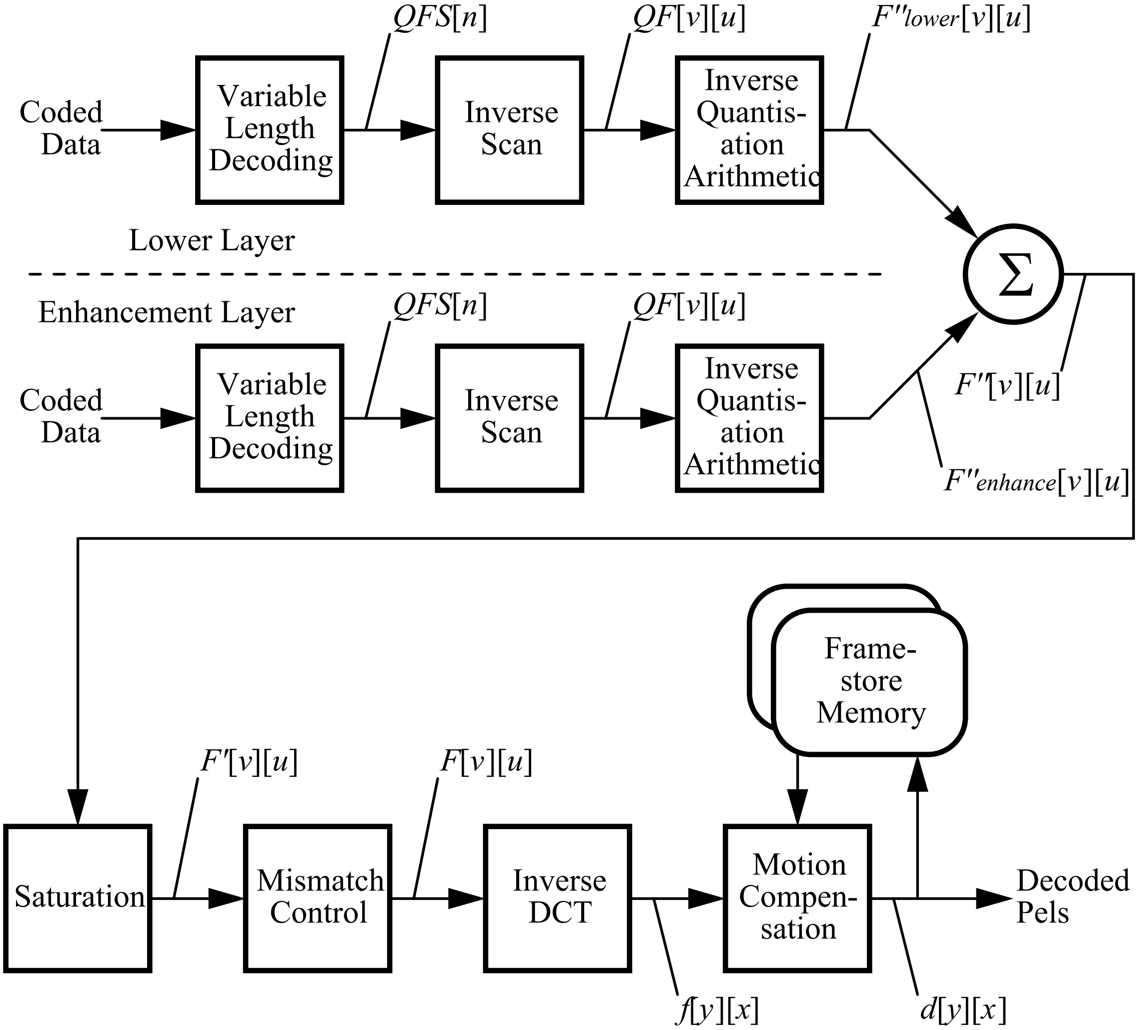


Figure 7-14. Illustration of decoding process for SNR scalability

This clause describes the additional decoding process required for the SNR scalable extensions.

SNR scalability defines a mechanism to refine the DCT coefficients encoded in another layer of a stream. As illustrated in Figure 7-14 data from two bitstreams is combined after the inverse quantisation processes by adding the DCT coefficients. Until the data is combined the decoding process of the two layers is independent of one another.

7.8.1 defines how to identify these bitstreams in a multilayer set of streams, however they can be classified as follows.

The "lower layer", derived from the first bitstream, can itself be either non-scalable, or require the spatial or temporal scalability decoding process to be applied.

The "enhancement" layer, derived from the second bitstream, contains mainly coded DCT coefficients and a small overhead. The decoding process for this layer and the combination of the two layers are described in this clause.

Note All information regarding prediction is contained in the lower layer only. Therefore it is not possible to reconstruct an enhancement layer without decoding the lower layer data in parallel.

Furthermore prediction and reconstruction of the pictures as described in 7.6, 7.7 and 7.9 for the combined lower and enhancement layer is identical to the respective steps for decoding of the lower layer only.

Semantics and decoding process described in this clause include a mechanism for "chroma simulcast". This may be used (for instance) to enhance a 4:2:0 signal in the lower layer to a 4:2:2 signal after processing the enhancement layer data. While the luminance data is processed as described before, in this case the chrominance information retrieved from the lower layer (with exception of intra-DC values, see 7.8.3.4) shall be discarded and replaced by the new information with higher chrominance resolution decoded from the enhancement layer.

It is inherent in SNR scalability that the two layers are very tightly coupled to one another. It is a requirement that corresponding pictures in each layer shall be decoded at the same time as one another.

In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then two different IDCT mismatch control schemes are being used in decoding. Care must be taken in the encoder to take account of this.

7.8.1 Higher syntactic structures

The two bitstreams layers in this clause are identified by their layer_id, decoded from the sequence_scalable_extension.

The two bitstreams shall have consecutive layer ids, with enhancement layer having layer_id = $id_{enhance}$ and the lower layer having layer_id = $id_{enhance}$-1.

The syntax and semantics of the enhancement layer are as defined in 6.2 and 6.3, respectively.

In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then both this lower and the enhancement layer shall use the "restricted slice structure" defined in this specification.

Semantic restrictions apply to several values in the headers and extensions of the enhancement layer as follows:

Sequence header

This header shall be identical to the one in the lower layer except for the values of bit_rate, vbv_buffer_size, load_intra_quantiser_matrix, intra_quantiser_matrix, load_non_intra_quantiser_matrix and non_intra_quantiser_matrix. These can be selected independently except for load_intra_quantiser_matrix which shall be zero.

Sequence extension

This extension shall be identical to the one in the lower layer except for the values of profile_and_level_indication, chroma_format, bit_rate_extension and vbv_buffer_size_extension. Those can be selected independently.

A different value of chroma_format in each layer will cause the chroma_simulcast flag to be set as specified by Table 7-24.

The chroma_format of the enhancement layer shall be higher or equal to the chroma_format of the lower layer.

Table 7-24 chroma_simulcast flag

chroma_format (lower layer)	chroma_format (enhancement layer)	chroma_simulcast
4:2:0	4:2:0	0
4:2:0	4:2:2	1
4:2:0	4:4:4	1
4:2:2	4:2:2	0
4:2:2	4:4:4	1
4:4:4	4:4:4	0

In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification), sequence_extension() is not present in the lower layer, and the following values shall be assumed for the decoding process.

progressive_sequence = 1
chroma_format = "4:2:0"
horizontal_size_extension = 0
vertical_size_extension = 0
bit_rate_extension = 0
vbv_buffer_size_extension = 0
low_delay = 0
frame_rate_extension_n = 0
frame_rate_extension_d = 0

The sequence_extension() in the enhancement layer shall have the values shown above.

Sequence display extension

This extension shall not be present as there is no separate display process for the enhancement layer.

Sequence scalable extension

This extension shall be present with scalable_mode = "SNR scalability".

GOP header

This header (if present) shall be identical to the one in the lower layer.

Picture header

This header shall be identical to the one in the lower layer except for the value of vbv_delay. This can be selected independently.

Picture coding extension

This extension shall be identical to the one in the lower layer except for the value of q_scale_type and alternate_scan. These can be selected independently.

chroma_420_type shall be set to "0" if chroma_simulcast is set. Else it shall have the same value as in the lower layer.

In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then picture_coding_extension() is not present in the lower layer and the following values shall be assumed for the decoding process:

f_code[0][0]	=	forward_f_code in the lower layer or 15
f_code[0][1]	=	forward_f_code in the lower layer or 15
f_code[1][0]	=	backward_f_code in the lower layer or 15
f_code[1][1]	=	backward_f_code in the lower layer or 15
intra_dc_precision	=	0
picture_structure	=	“Frame Picture”
top_field_first	=	0
frame_pred_frame_dct	=	1
concealment_motion_vectors	=	0
intra_vlc_format	=	0
repeat_first_field	=	0
chroma_420_type	=	1
progressive_frame	=	1
composite_display_flag	=	0

The picture_coding_extension() in the enhancement layer shall have the values shown above.

For the lower layer q_scale_type and alternate_scan shall be assumed to have the value zero.

Note q_scale_type and alternate_scan can be set independently in the enhancement layer.

Quant matrix extension

This extension is optional. Semantics are described in 6.3.7.

load_intra_quantiser_matrix and load_chroma_intra_quantiser_matrix shall both be zero.

Note Only the non-intra matrices will be used in the subsequent decoding process.

Pan-scan extension

This extension shall not be present.

Note There is no separate display process for the enhancement layer. If pan-scan functionality is desired it can be accomplished already by using the information conveyed by the pan-scan extension of the lower layer.

Slice header

Slices shall be coincident with those in the lower layer. The value of quantiser_scale_code can be set independently from the lower layer.

7.8.2 Macroblock

Subsequently the "current macroblock" denotes the currently processed macroblock. The "current macroblock of the lower layer" denotes the macroblock identified by having the same macroblock_address as the current macroblock.

The decoding of the macroblock header information is done according to semantics in 6.3.16.

Note Table B-8 which is used if scalable_mode == "SNR scalability" will never set the macroblock_intra, macroblock_motion_forward or macroblock_motion_backward flags, since a macroblock in the enhancement layer contains only refinement data for the current macroblock of the lower layer.

However the corresponding syntax elements and flags of the current macroblock in the lower layer are relevant for the combined decoding process of lower and enhancement layer following the inverse DCT as described in 7.8.3.5.

7.8.2.1 dct_type

The syntax element dct_type may be present in none, one or both of the lower and enhancement layer macroblock_modes(), as indicated by the semantics in 6.3.16.

If dct_type is present in the macroblock_modes() in both layers it shall have identical values.

7.8.2.2 Skipped Macroblocks

Macroblocks can be skipped in the enhancement layer, meaning that no coefficient enhancement is done. In that case the decoding process is exactly as specified in 7.6.6.

Macroblocks can also be skipped in the lower layer, while still being coded in the enhancement layer. In that case the decoding process detailed in the following has to be applied.

7.8.3 Block

The first part of the decoding process of the enhancement layer block is independent from the lower layer.

The second part of the decoding process of the enhancement layer block has to be done jointly with the decoding process of the coincident lower layer block.

Two sets of inverse quantised coefficients F''_{lower} and $F''_{enhance}$ are added to form F" (see Figure 7-14).

F''_{lower} is derived from the lower layer exactly as defined in 7.1 to 7.4.2.3.

$F''_{enhance}$ is derived as is defined in the clauses below.

The resulting signal F" is further processed, starting with saturation, as defined in 7.4.4 to 7.6 (7.7, 7.9)

7.8.3.1 Variable length decoding

In an enhancement layer block the VLC decoding shall be performed according to 7.2., as for a non-intra block (as indicated by macroblock_intra = 0).

7.8.3.2 Inverse scan

Inverse scan shall be done exactly as defined in 7.3

7.8.3.3 Inverse quantisation

In an enhancement layer block the inverse quantisation shall be performed according to 7.4.2 as for a non-intra block.

In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then the “inverse quantisation arithmetic” used to derive $F''_{lower}[v][u]$ (see Figure 7-14) shall include the IDCT mismatch control (oddification) and saturation specified in ISO/IEC 11172-2.

7.8.3.4 Addition of coefficients from the two layers

Corresponding coefficients from the blocks of each layer shall be added together to form the signal F'' (see Figure 7-14).

$$F''[v][u] = F''_{lower}[v][u] + F''_{enhance}[v][u], \text{ for all } u, v$$

If chroma_simulcast = 1 is set only the luminance blocks are treated as described above.

For chrominance blocks the DC coefficient of the base layer is used as a prediction of the DC coefficient in the coincident block in the enhancement layer, whereas the AC coefficients of the base layer are discarded and AC coefficients of the enhancement layer form the signal F" in Figure 7-14 according to the following formulae:

$$F''[0][0] = F''_{lower}[0][0] + F''_{enhance}[0][0]$$

$$F''[v][u] = F''_{enhance}[v][u], \text{ for all } u, v \text{ except } u = v = 0$$

Note Chroma simulcast blocks are inverse quantised like non-intra blocks and use the chrominance non-intra matrix.

Table 7-25 gives the index of the chrominance block whose DC coefficient ($F''_{lower}[0][0]$) is to be used to predict the DC coefficient in the coincident chrominance block of the enhancement layer ($F''_{enhance}[0][0]$).

Table 7-25. block index used to predict DC coefficient

	block index							
chroma_format	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
base: 4:2:0 upper: 4:2:2	4	5	4	5				
base: 4:2:0 upper: 4:4:4	4	5	4	5	4	5	4	5
base: 4:2:2 upper: 4:4:4	4	5	6	7	4	5	6	7

7.8.3.5 Remaining macroblock decoding steps

After addition of coefficients from the two layers, the remainder of the macroblock decoding steps is exactly as described in 7.4.4 to 7.6 (7.7, 7.9, if applicable), since there is now only one data stream F"[v][u] to be processed.

In this process, the spatio/temporal prediction signal p[y][x] is derived according to the macroblock type syntax elements and flags for the current macroblock known from the lower layer.

7.9 Temporal scalability

Temporal scalability involves two layers, a lower layer and an enhancement layer. Both the lower and the enhancement layers process the same spatial resolution. The enhancement layer enhances the temporal resolution of the lower layer and if temporally remultiplexed with the lower layer signal provides full temporal rate. This is the frame rate indicated in the enhancement layer. The decoding process for enhancement layer pictures is similar to the normal decoding process described in 7.1 to 7.6. The only difference is in the "Prediction field and frame selection" described in 7.6.2.

The reference pictures for prediction are selected by reference_select_code as described in Tables 7-26 and 7-27. In P pictures, the forward reference picture can be one of the following three: most recent enhancement picture, most recent lower layer frame, or next lower layer frame in display order. Note that in the latter case, the reference frame in lower layer used for prediction is backward in time.

In B-pictures, the forward reference can be one of the following two: most recent the enhancement pictures or most recent (or temporally coincident) lower layer frame whereas the backward reference can be one of the following two: most recent lower layer picture including temporally coincident picture in display order or next lower layer frame in display order. Note that in this case, the backward reference frame in lower layer used for prediction is forward in time.

Backward prediction cannot be made from a picture in the enhancement layer. This avoids the need for frame reordering in the enhancement layer. Motion compensation process forms predictions using lower layer decoded pictures and/or previous temporal prediction from the enhancement layer.

The enhancement layer can contain I-pictures, P-pictures or B-pictures, but B-pictures in enhancement layer behave more like P-pictures in the sense that a decoded B-picture can be used to predict the following P-pictures or B-pictures in the enhancement layer.

When the most recent frame in the lower layer is used as the reference, this includes the frame that is temporally coincident with the frame or the first field (in case of field pictures) in the enhancement layer. The prediction references used for P-picture and B-pictures are shown in Table 7-26 and Table 7-27 respectively.

The lower and enhancement layers shall use the restricted slice structure.

Table 7-26 Prediction references selection in P-pictures

reference_select_code	forward prediction reference
00	Most recent decoded enhancement picture(s)
01	Most recent lower layer frame in display order
10	Next lower layer frame in display order
11	forbidden

Table 7-27 Prediction references selection in B-pictures

reference_ select_ code	forward prediction reference	backward prediction reference
00	forbidden	forbidden
01	Most recent decoded enhancement picture(s)	Most recent lower layer picture in display order
10	Most recent decoded enhancement picture(s)	Next lower layer picture in display order
11	Most recent lower layer picture in display order	Next lower layer picture in display order

Figure 7-15 shows a simplified diagram of the motion compensation process for the enhancement layer using temporal scalability.

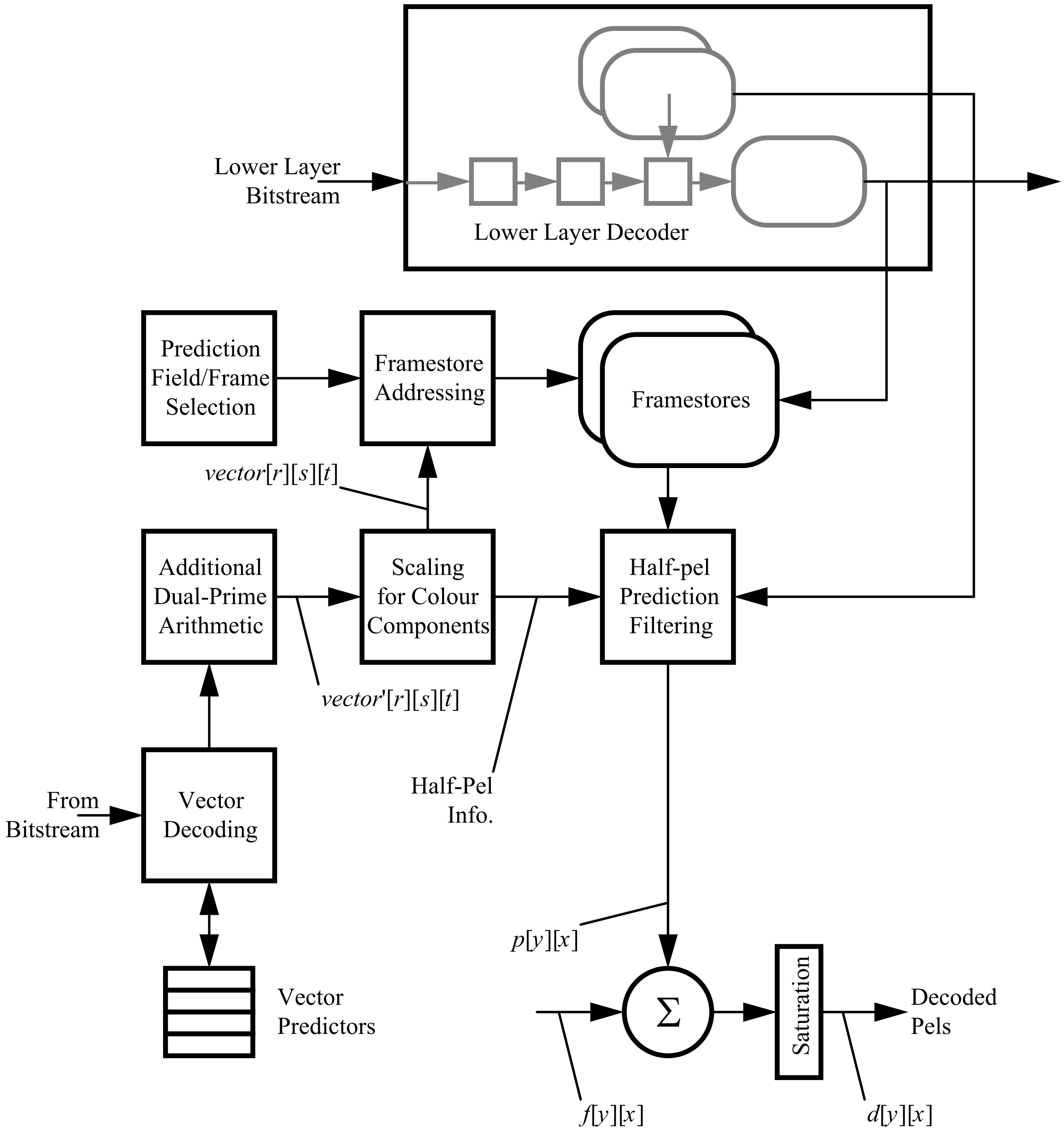


Figure 7-15 Simplified motion compensation process for the enhancement layer using temporal scalability.

I-pictures do not use prediction references; to indicate this, the reference_select_code for I-pictures shall be '11' .

Depending on picture_type, when forward_temporal_reference or backward_temporal_reference do not imply references to be used for prediction, they shall take the value 0.

7.9.1 Higher syntactic structures

The two bitstreams layers in this Clause are identified by their layer_id, decoded from the sequence_scalable_extension.

The two bitstreams shall have consecutive layer ids, with enhancement layer having layer_id=$id_{enhance}$ and the lower layer having layer_id=$id_{enhance}$-1.

The syntax and semantics of enhancement layers are as defined in Clauses 6.2 and 6.3 respectively.

Semantic restrictions apply to several values in the headers and extensions of the enhancement layer as follows.

The lower layer shall conform to this specification (and not to ISO/IEC 11172-2).

Sequence header

The values in this header can be different from the lower layer except for horizontal_size_value, vertical_size_value and aspect_ratio_information.

Sequence extension

This extension shall be identical to the one in the lower layer except for values of profile_and_level_indication, bit_rate_extension, vbv_buffer_size_extension, low_delay, frame_rate_extension_n and frame_rate_extension_d. These can be selected independently. Note that progressive_sequence indicates the scanning format of the enhancement layer frames only rather than of the output frames after multiplexing. The latter is indicated by mux_to_progressive_sequence (see sequence scalable extension).

Sequence display extension

This extension shall not be present as there is no separate display process for the enhancement layer.

Sequence scalable extension

This extension shall be present with scalable_mode = “Temporal scalability”.

When progressive_sequence=0 and mux_to_progressive_sequence=0, top_field_first and picture_mux_factor can be selected.

When progressive_sequence=0 and mux_to_progressive_sequence=1, top_field_first shall contain a complement of the value of top_field_first of the lower layer but picture_mux_factor shall be 1.

When progressive_sequence=1 and mux_to_progressive_sequence=1, top_field_first shall be zero but picture_mux_factor can be selected.

The combination of progressive_sequence=1 and mux_to_progressive_sequence=0 shall not occur.

GOP header

There is no restriction on GOP header (if present) to be the same as that for lower layer

Picture header

There is no restriction on picture headers to be the same as in the lower layer.

Picture coding extension

The values in this extension can be different from the lower layer except for top_field_first, concealment_motion_vectors, and chroma_420_type and progressive_frame. The top_field_first shall be based on progressive_sequence and mux_to_progressive_sequence (see sequence_scalable_extension above) and concealment_motion_vectors shall be 0. Chroma_420_type shall be identical to the lower layer. Progressive_frame shall always have the same value as progressive_sequence.

Picture temporal scalable extension

This extension shall be present for each picture.

Quant matrix extension

This extension may be present in the enhancement layer.

7.9.2 Restrictions on Temporal Prediction

Although temporal predictions can be made from decoded pictures referenced by forward_temporal_reference or both forward_temporal_reference and backward_temporal_references, temporal scalability is efficient if predictions are formed using decoded picture/pictures from lower layer and enhancement layer that are very close in time to the enhancement picture being predicted. It is a requirement on the bitstreams that P- pictures and B- pictures shall form predictions from most recent or next pictures as illustrated by Tables 7-26 and 7-27.

In case group_of_pictures_header occurs very often in lower_layer, ambiguity can occur due to possibility of nonuniqueness of temporal references (which are reset at each group_of_pictures_header). This ambiguity shall be resolved with help of systems layer timing information.

7.10 Data partitioning

Data partitioning is a technique that splits a video bitstream into two layers, called partitions. A priority breakpoint indicates which syntax elements are placed in partition 0, which is the base partition (also called high priority partition). The remainder of the bitstream is placed in partition 1 (which is also called low priority partition). Sequence, GOP, and picture headers are redundantly copied in partition 1 to facilitate error recovery. The sequence_end_code is also redundantly copied into partition one. All fields in the redundant headers must be identical to the original ones. The only extensions allowed (and required) in partition 1 are sequence_extension(), picture_coding_extension() and sequence_scalable_extension().

Note The slice() syntax given in 6.2.4 is followed in both partitions up to (an including) the syntax element extra_bit_slice.

The interpretation of priority_breakpoint is given in Table 7-28.

Table 7-28 Priority breakpoint values and associated semantics

priority_breakpoint	Syntax elements included in partition zero
0	This value is reserved for partition 1. All slices in partition 1 shall have a priority_breakpoint equal to 0.
1	All data at the sequence, GOP, picture and slice() down to **extra_bit_slice** in slice().
2	All data included above, plus macroblock syntax elements up to and including **macroblock_address_increment**.
3	All data included above, plus macroblock syntax elements up to but not including coded_block_pattern().
4 ... 63	Reserved.
64	All syntax elements up to and including coded_block_pattern() or DC coefficient (**dct_dc_differential**), and the first (run, level) DCT coefficient pair (or EOB).[†]
65	All syntax elements above, plus up to 2 (run, level) DCT coefficient pairs.
...	
63+*j*	All syntax elements above, plus up to *j* (run, level) DCT coefficient pairs.
...	
127	All syntax elements above, plus up to 64 (run, level) DCT coefficient pairs.

[†] Note that a priority_breakpoint immediately following the DC coefficient is disallowed since it might cause start code emulation.

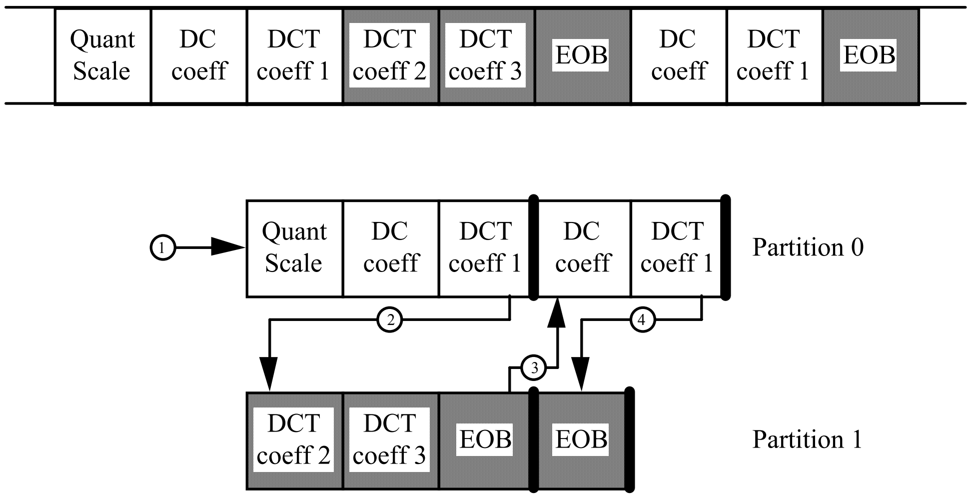


Figure 7-16 A segment from a bitstream with two partitions, with priority_breakpoint set to 64 (one (run, level) pair). The two partitions are shown, with arrows indicating how the decoder needs to switch between partitions.

Semantics of VBV remains unchanged, i.e. the VBV refers to the sum of two partitions, not any single one.

The bitstream parameters bit_rate (bit_rate_value and bit_rate_extension), vbv_buffer_size (vbv_buffer_size_value and vbv_buffer_size_extension) and vbv_buffer_delay shall take the same value in the two partitions. These parameters refer to the characteristics of the entire bitstream formed from the two partitions.

The decoding process is modified in the following manner:

Set *current_partition* to 0, and start decoding from bitstream that contains the sequence_scalable_extension (partition 0).

If *current_partition* = 0, check to see if the current point in the bitstream is a priority breakpoint.

If yes, set *current_partition* to 1. Next item will be decoded from partition. 1

Otherwise, continue decoding from partition 0. Remove sequence, GOP, and picture headers from both partitions.

If *current_partition = 1*, check the priority breakpoint to see if the next item to be decoded is expected in partition 0.

If yes, set *current_partition* to 0. Next item will be decoded from partition 0.

Otherwise, continue decoding from partition 1.

An example is shown in Figure 7-16 where the priority breakpoint is set at 64 (one (run, level) pair).

7.11 Hybrid scalability

Hybrid scalability is the combination of two different types of scalability. The types of scalability that can be combined are SNR scalability, spatial scalability and temporal scalability. When two types of scalability are combined, there are three bitstreams that have to be decoded. The layers to which these bitstreams belong are named in Table 7-29.

Table 7-29 Names of layers

layer_id	name
0	base layer
1	enhancement layer 1
2	enhancement layer 2
...	...

For the scalability between the enhancement layers 1 and 2, the enhancement layer 1 is its lower layer, and the enhancement layer 2 is its enhancement layer. No layer can be omitted from the hierarchical ladder. E.g., if there is SNR scalability between enhancement layer 1 and enhancement layer 2, the prediction types in enhancement layer 1 are also valid for the combined decoding process for enhancement layers 1 and 2.

The coupling of layers is more loose with spatial and temporal scalability than with SNR scalability. Therefore, in these kinds of scalability, first the base layer has to be decoded and upconverted before it can be used in the enhancement layer. In SNR scalability, both layers are decoded simultaneously. The decoding order can be summarised as follows :

case 1 :

base layer

<*spatial or temporal scalability*>

enhancement layer 1

<*SNR scalability*>

enhancement layer 2

First decode the base layer, and then decode both enhancement layers simultaneously.

case 2 :

base layer

<*SNR scalability*>

enhancement layer 1

<*spatial or temporal scalability*>

enhancement layer 2

First decode the base layer and the enhancement layer 1 simultaneously, and then decode the enhancement layer 2.

case 3 :

base layer

<spatial or temporal scalability>

enhancement layer 1

<spatial or temporal scalability>

enhancement layer 2

First decode the base layer, then decode the enhancement layer 1, and finally decode enhancement layer 2.

7.12 Output of the decoding process

This section describes the output of the theoretical model of the decoding process that decodes bitstreams conforming to this specification.

The decoding process input is video data, consisting of one or more layers. The video layers are generally multiplexed by the means of a system stream that also contains timing information.

The output of the decoding process is a series of fields or frames that are normally the input of a display process. The order in which fields or frames are output by the decoding process is called the display order, and may be different from the decoding order (when B-pictures are used). The display process is responsible for the action of displaying the decoded fields or frames on a display device. If the display device cannot display at the frame rate indicated in the bitstream, the display process may perform frame rate conversion. This specification does not describe a theoretical model of display process nor the operation of the display process.

Since some of the syntax elements, such as progressive_frame, may be needed by the display process, in this theoretical model of the decoding process, all the syntactic elements that are decoded by the decoding process are output by the decoding process and may be accessed by the display process.

Also the position of the samples in a frame (both temporally and spatially) are not used by the decoding process but must be known by the display process.

When the a progressive sequence is decoded (progressive_sequence is equal to 1), the luminance and chrominance samples of the reconstructed frames are output by decoding process in the form of progressive frames and the output rate is the frame rate. Figure 6-15 illustrates this in the case of chroma_format equals to 4:2:0.

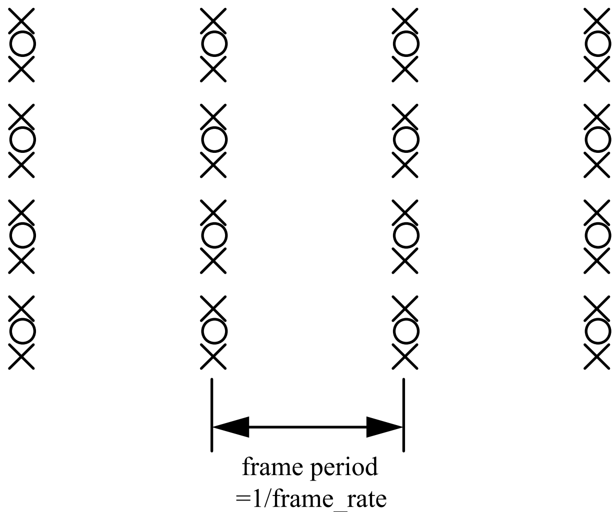


Figure 7-15. progressive_sequence == 1

The same reconstructed frame is output one time if repeat_first_field is equal to 0, and two or three consecutive times if repeat_first_field is equal to 1, depending on the value of top_field_first. Figure 6-15 illustrates this in the case of chroma_format equals to 4:2:0 and repeat_first_field equals 1.

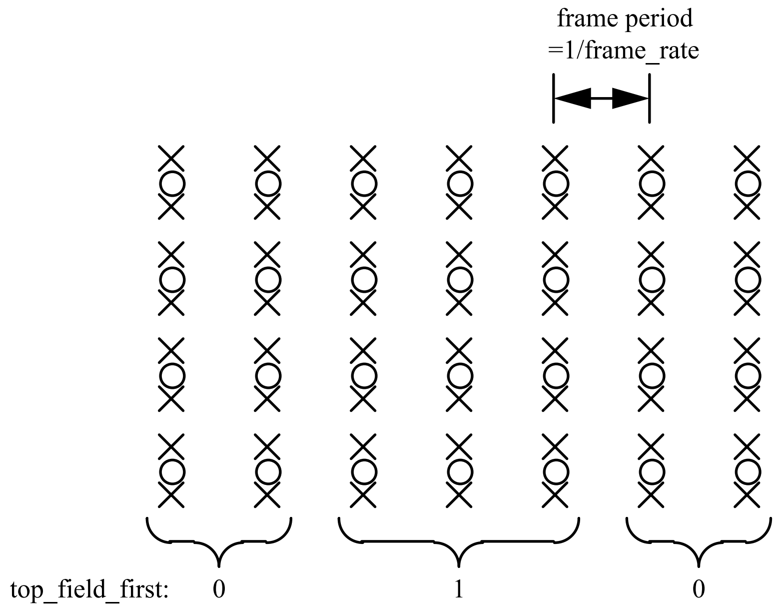


Figure 7-15. progressive_sequence == 1, repeat_first_feld = 1

When decoding an interlaced sequence (progressive_sequence is equal to 0), the luminance samples of the reconstructed frames are output by the decoding process in the form of interlaced fields at a rate that is twice the frame rate. Figure 6-16 illustrates this.

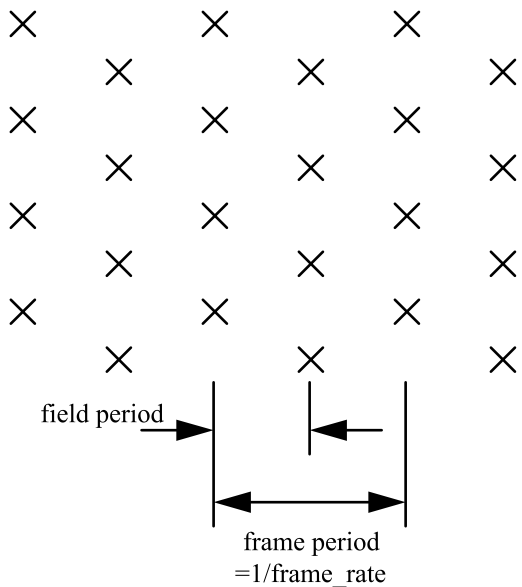


Figure 7-16. progressive_sequence == 0

It is a requirement on the bitstream that the fields at the output of the decoding process shall always be alternately top and bottom (note that the very first field of a sequence may be either top or bottom).

If the reconstructed frame is interlaced (progressive_frame is equal to 0), the luminance samples and chrominance samples are output by the decoding process in the form of two consecutive fields. The first field output by the decoding process is the top field or the bottom field of the reconstructed frame, depending on the value of top_field_first.

Although all the samples of progressive frames represent the same instant in time, all the samples are not output at the same time by the decoding process when the sequence is interlaced.

If the reconstructed frame is progressive (progressive_frame is equal to 1), the luminance samples are output by the decoding process in the form of two or three consecutive fields, depending on the value of repeat_first_field.

Note The information that these fields originate from the same progressive frame in the bitstream is conveyed to the display process.

All of the chrominance samples of the reconstructed progressive frame are output by the decoding process at the same time as the first field of luminance samples. This is illustrated in Figures 6-17 and 6-18.

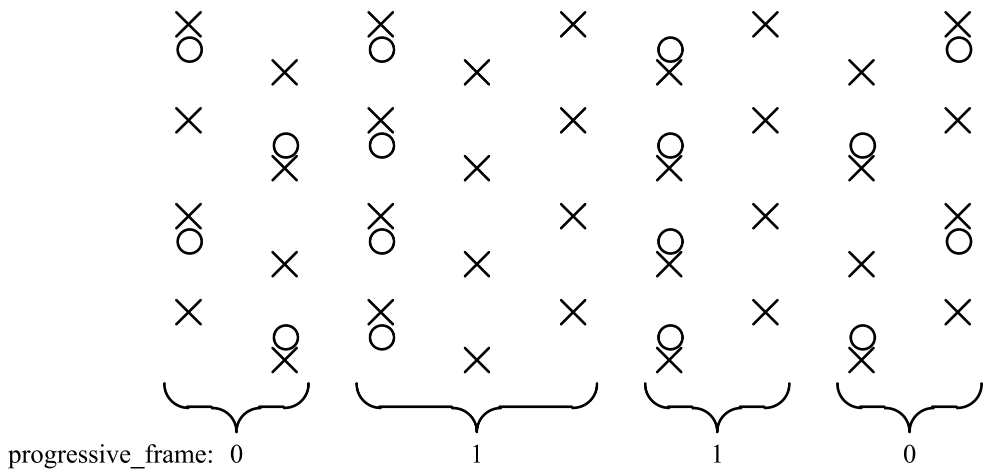


Figure 7-17. progressive_sequence == 0 with 4:2:0 chrominance.

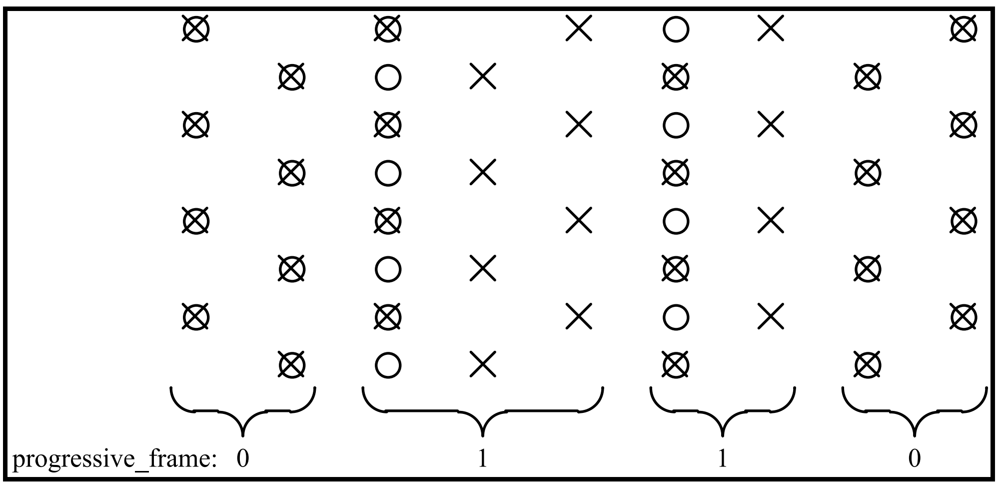


Figure 7-18. progressive_sequence == 0 with 4:2:2 or 4:4:4 chrominance.

8 Profiles and levels

Note In this specification the word “profile” is used as defined below. It should not be confused with other definitions of “profile” and in particular it does not have the meaning that is defined by JTC1/SGFS.

Profiles and levels provide a means of defining subsets of the syntax and semantics of this specification and thereby the decoder capabilities required to decode a particular bitstream. A profile is a defined sub-set of the entire bitstream syntax that is defined by this specification. A level is a defined set of constraints imposed on parameters in the bitstream. Conformance tests will be carried out against defined profiles at defined levels.

The purpose of defining conformance points in the form of profiles and levels is to facilitate bitstream interchange among different applications. Implementers of this specification are encouraged to produce decoders and bitstreams which correspond to those defined conformance regions. The discretely defined profiles and levels are the means of bitstream interchange between applications of this specification.

In this clause the constrained parts of the defined profiles and levels are described. All syntactic elements and parameter values which are not explicitly constrained may take any of the possible values that are allowed by this specification. A decoder shall be deemed to be conformant to a given profile at a given level if it is able to properly decode all allowed values of all syntactic elements as specified by that profile at that level. A bitstream shall be deemed to be conformant if it does not exceed the allowed range of allowed values and does not include disallowed syntactic elements.

Attention is drawn to 5.4 which defines the convention for specifying a range of numbers. This is used throughout to specify the range of values and parameters.

The profile_and_level_indication in the sequence_extension indicates the profile and level to which the bitstream complies. The meaning of the bits in this parameter is defined in Table 8-1.

Table 8-1. Meaning of bits in profile_and_level_indication.

Bits	Field Size (bits)	Meaning
[7:7]	1	Escape bit
[6:4]	3	Profile identification
[3:0]	4	Level identification

Table 8-2 specifies the profile identification codes and Table 8-3 the level identification codes. When the escape bit equals zero a profile with a numerically larger identification value will be a subset of a profile with a numerically smaller identification value. Similarly, whenever the escape bit equals zero, a level with a numerically larger identification value will be a subset of a level with a numerically smaller identification value.

Table 8-2. Profile identification.

Profile identification	Profile
110 to 111	(reserved)
101	Simple
100	Main
011	SNR Scalable
010	Spatially Scalable
001	High
000	(reserved)

Table 8-3. Level identification.

Level identification	Level
1011 to 1111	(reserved)
1010	Low
1001	(reserved)
1000	Main
0111	(reserved)
0110	High 1440
0101	(reserved)
0100	High
0000 to 0011	(reserved)

Table 8-4 describes profiles and levels when the escape bit equals 1. For these profiles and levels there is no implied hierarchy from the assignment of profile_and_level_indication and profiles and levels are not necessarily subsets of others.

Table 8-4. Escape profile_and_level_indication identification.

profile_and_level_indication	Name
10000000 to 11111111	(reserved)

Attention is drawn to Annex E, which describes in detail those parts of ISO/IEC 13818-2 that are used for a given profile and level.

8.1 ISO/IEC 11172-2 compatibility

ISO/IEC 11172-2 “constrained parameter” bitstreams shall be decodable by Simple, Main, SNR Scalable, Spatially Scalable and High profile decoders at all levels. When a bitstream conforming to ISO/IEC 11172-2 constrained parameter coding is generated, the **constrained_parameters_flag** shall be set.

Additionally Simple, Main, SNR Scalable, Spatially Scalable and High profile decoders shall be able to decode D-pictures-only bitstreams of ISO/IEC 11172-2 which are within the level constraints of the decoder.

8.2 Relationship between defined profiles

The Simple, Main, SNR Scalable, Spatially Scalable and High profiles have a hierarchical relationship. Therefore the syntax supported by a ‘higher’ profile includes all the syntactic elements of ‘lower’ profiles (e.g., for a given level, a Main profile decoder shall be able to decode a bitstream conforming to Simple profile restrictions). For a given profile, the same syntax set is supported regardless of level. The order of hierarchy is given in Table 8-2.

The syntactic differences between profiles are given in Table 8-5:

Table 8-5. Syntactic constraints of profiles

Syntactic Element	Profile				
	Simple	**Main**	**SNR**	**Spatial**	**High**
chroma_format	4:2:0	4:2:0	4:2:0	4:2:0	4:2:2 or 4:2:0
frame_rate_extension_n	0	0	0	0	0
frame_rate_extension_d	0	0	0	0	0
picture_coding_type	I, P	I, P, B	I, P, B	I, P, B	I, P, B
repeat_first_field	Constrained		Unconstrained		
sequence_scalable_extension()	No	No	Yes	Yes	Yes
scalable_mode	-	-	SNR	SNR or Spatial	SNR or Spatial
picture_spatial_scalable_extension()	No	No	No	Yes	Yes
intra_dc_precision	8, 9, 10	8, 9, 10	8, 9, 10	8, 9, 10	8, 9, 10, 11
Slice structure	Restricted See 6.1.3.2				

For all defined profiles, there is a semantic restriction on the bitstream that all of the data for the macroblock shall be represented with not more than the number of bits indicated by Table 8-6. However, any two macroblocks in each horizontal row of macroblocks may exceed this limitation.

In this context a macroblock is deemed to start with the first bit of the macroblock_address_increment (or macroblock_escape, if any) and continue until the last bit of the “End of block” symbol of the last coded block (or the last bit of the coded_block_pattern() if there are no coded blocks). The bits required to represent any slice_header() that precedes (or follows) the macroblock are not counted as part of the macroblock.

Table 8-6 — Maximum number of bits in a macroblock

chroma_format	Maximum number of bits
4:2:0	4608
4:2:2	6144
4:4:4	9216

The use of **repeat_first_field** in Simple and Main profile bitstreams is constrained as specified in Table 8-7.

Table 8-7. Constraints on use of repeat_first_field

		progressive_ sequence==0	progressive_ sequence==1
frame_rate_code	**frame_rate_value**	**repeat_first_field**	**repeat_first_field**
0000	forbidden		
0001	24 000 / 1001 (23,976)	0	0
0010	24	0	0
0011	25	0 or 1	0
0100	30 000 / 1001 (29,97)	0 or 1	0
0101	30	0 or 1	0
0110	50	0 or 1	0
0111	60 000 / 1001 (59,94)	0 or 1	0 or 1
1000	60	0 or 1	0 or 1
...	reserved		
1111	reserved		

The High profile is also distinguished by having different constraints on luminance sample rate, maximum bit rate, and VBV buffer size. Refer to tables 8-11, 8-12 and 8-13.

Decoders that are Simple profile @ Main level compliant shall be capable of decoding Main profile @ Low level bitstreams.

8.3 Relationship between defined levels

The Low, Main, High-1440 and High levels have a hierarchical relationship. Therefore the parameter constraints of a 'higher' level equal or exceed the constraints of 'lower' levels (e.g., for a given profile, a Main level decoder shall be able to decode a bitstream conforming to Low level restrictions). The order of hierarchy is given in Table 8-3.

The different parameter constraints for levels are given in Table 8-6:

Table 8-8. Parameter constraints for levels

Syntactic Element	Level			
	Low	Main	High-1440	High
f_code[0][0] (forward horizontal)	[1:7]	[1:8]	[1:9]	[1:9]
f_code[0][1] (forward vertical)	[1:4]	[1:5]	[1:5]	[1:5]
f_code[1][0]*(backward horizontal)	[1:7]	[1:8]	[1:9]	[1:9]
f_code[1][1]*(backward vertical)	[1:4]	[1:5]	[1:5]	[1:5]
vertical vector range Frame Picture†	[-64:63,5]	[-128:127,5]	[-128:127,5]	[-128:127,5]
vertical vector range Field Picture†	[-32:31,5]	[-64:63,5]	[-64:63,5]	[-64:63,5]
frame_rate_code	[1:5]	[1:5]	[1:8]	[1:8]
Sample Density	See Table 8-10			
Luminance Sample Rate	See Table 8-11			
Maximum Bit Rate	See Table 8-12			
Buffer Size	See Table 8-13			

* For Simple profile bitstreams which do not include B-pictures, **backward_horizontal_f_code** and **backward_vertical_f_code** shall be set to 15 (not used).

† This restriction applies to the final reconstructed motion vector. In the case of dual prime motion vectors it applies before scaling is performed, after scaling is performed and after the small differential motion vector has been added.

8.4 Scalable layers

The SNR Scalable, Spatial Scalable and High profiles may use more than one bitstream to code the image. These different bitstreams represent layers of coding, which when combined create a higher quality image than that obtainable from one layer alone (see annex D). The maximum number of layers for a given profile is specified in table 8-7. The scalable layers are named according to Table 7-29. The syntactic and parameter contraints for these profile / level combinations when coded using the maximum permitted number of layers are given in tables 8-10, 8-11, 8-12 and 8-13. When the number of layers is less than the maximum permitted, reference should also be made to tables 8-15 to 8-17 as appropriate.

It should be noted that the base layer of an SNR profile bitstream can always be decoded by a Main profile decoder of equivalent level. Conversely, a Main profile bitstream shall be decodable by an SNR profile decoder of equivalent level.

Table 8-9. Upper bounds for scalable layers in SNR, Spatial and High profiles

Level	Maximum Number of	Profile		
		SNR	Spatial	High
High	All layers (base + enh.)			3
	Spatial enhancement layers			1
	SNR enhancement layers			1
High-1440	All layers (base + enh.)		3	3
	Spatial enhancement layers		1	1
	SNR enhancement layers		1	1
Main	All layers (base + enh.)	2		3
	Spatial enhancement layers	0		1
	SNR enhancement layers	1		1
Low	All layers (base + enh.)	2		
	Spatial enhancement layers	0		
	SNR enhancement layers	1		

8.4.1 Permissible layer combinations

There are many possible combinations of layers in the SNR, Spatial and High profiles. In order to maximise interoperabilty, only a subset of all permutations are permitted, and for certain combinations the parameter constraints are more restrictive than indicated by tables 8-10, 8-11, 8-12 and 8-13. These additional restrictions are to ensure base layer bitstream decoding may be performed by a decoder of a defined lower profile / level.

The following table is a summary of the permitted combinations, and is subject to the following rules:

- SNR profile maximum of 2 layers; Spatial & High profile - maximum of 3 layers. (See Table 8-9)
- Only one SNR and one Spatial scale allowed in 3-layer combinations, either SNR/Spatial or Spatial/SNR order is permitted. (See Table 8-9)
- Adding 4:2:2 chroma format to a 4:2:0 lower layer is considered an SNR scale.
- A 4:2:0 layer is not permitted if the lower layer is 4:2:2. (See 7.8.1)

	Scalable mode			**Profile / level of simplest base layer decoder**
Profile	**Base layer**	**Enh. layer 1**	**Enh. layer 2**	**(level ref. top layer) ***
SNR	4:2:0	-	-	MP@same level
SNR	4:2:0	SNR	-	MP@same level
Spatial	4:2:0	-	-	MP@same level
Spatial	4:2:0	SNR, 4:2:0	-	MP@same level
Spatial	4:2:0	Spatial, 4:2:0	-	MP@(level - 1)
Spatial	4:2:0	SNR, 4:2:0	Spatial, 4:2:0	MP@(level - 1)
Spatial	4:2:0	Spatial, 4:2:0	SNR, 4:2:0	MP@(level - 1)
High	4:2:0	-	-	HP@same level
High	4:2:2	-	-	HP@same level
High	4:2:0	SNR, 4:2:0	-	HP@same level
High	4:2:0	SNR, 4:2:2	-	HP@same level
High	4:2:2	SNR, 4:2:2	-	HP@same level
High	4:2:0	Spatial, 4:2:0	-	HP@(level - 1)
High	4:2:2	Spatial, 4:2:2	-	HP@(level - 1)
High	4:2:0	SNR, 4:2:0	Spatial, 4:2:0	HP@(level - 1)
High	4:2:0	SNR, 4:2:2	Spatial, 4:2:2	HP@(level - 1)
High	4:2:2	SNR, 4:2:2	Spatial, 4:2:2	HP@(level - 1)
High	4:2:0	Spatial, 4:2:0	SNR, 4:2:0	HP@(level - 1)
High	4:2:0	Spatial, 4:2:0	SNR, 4:2:2	HP@(level - 1)
High	4:2:2	Spatial, 4:2:2	SNR, 4:2:2	HP@(level - 1)

* The simplest compliant decoder to decode the base layer is specified, assuming that bitstream may contain any syntax permitted for the stated profile, except scalability. Note that for High profile @ Main level spatially scaled bitstreams, 'HP@(level - 1)' becomes 'MP@(level - 1)'.

1 8.5 Parameter values for defined profiles, levels and layers

2 **Table 8-10. Upper bounds for sampling density**

Level	Spatial resolution layer		Profile: Simple	Main	SNR	Spatial	High
High	Enhancement	samples/line		1920			1920
		lines/frame		1152			1152
		frames/sec		60			60
	Lower	samples/line					960
		lines/frame		-			576
		frames/sec					30
High-1440	Enhancement	samples/line		1440		1440	1440
		lines/frame		1152		1152	1152
		frames/sec		60		60	60
	Lower	samples/line				720	720
		lines/frame		-		576	576
		frames/sec				30	30
Main	Enhancement	samples/line	720	720	720		720
		lines/frame	576	576	576		576
		frames/sec	30	30	30		30
	Lower	samples/line					352
		lines/frame	-	-	-		288
		frames/sec					30
Low	Enhancement	samples/line		352	352		
		lines/frame		288	288		
		frames/sec		30	30		
	Lower	samples/line					
		lines/frame		-	-		
		frames/sec					

Note: In the case of single layer or SNR scaled coding, the limits specified by ‘Enhancement layer’ apply

3 The syntactic elements referenced by this table are as follows:

4 samples/line : **horizontal_size_value**
5 lines/frame : **vertical_size_value**
6 frames/sec : **frame_rate_value**

Table 8-11. Upper bounds for luminance sample rate (samples/sec)

Level	Spatial resolution layer	Profile				
		Simple	**Main**	**SNR**	**Spatial**	**High**
High	Enhancement		62 668 800			62 668 800 (4:2:2) 83 558 400 (4:2:0)
	Lower		-			14 745 600 (4:2:2) 19 660 800 (4:2:0)
High-1440	Enhancement		47 001 600		47 001 600	47 001 600 (4:2:2) 62 668 800 (4:2:0)
	Lower		-		10 368 000	11 059 200 (4:2:2) 14 745 600 (4:2:0)
Main	Enhancement	10 368 000	10 368 000	10 368 000		11 059 200 (4:2:2) 14 745 600 (4:2:0)
	Lower	-	-	-		- 3 041 280 (4:2:0)
Low	Enhancement		3 041 280	3 041 280		
	Lower		-	-		
Note: In the case of single layer or SNR scaled coding, the limits specified by 'Enhancement layer' apply						

The luminance sample rate, P is defined as follows:

P = **horizontal_size_value** x **vertical_size_value** x **frame_rate_value**

Table 8-12. Upper bounds for bit rates (Mbit/s)

Level	Profile				
	Simple	**Main**	**SNR**	**Spatial**	**High**
High		80			100 all layers 80 middle + base layer 25 base layer
High-1440		60		60 all layers 40 middle + base layers 15 base layer	80 all layers 60 middle + base layers 20 base layer
Main	15	15	- 15 both layers 10 base layer		20 all layers 15 middle + base layer 4 base layer
Low		4	- 4 both layers 3 base layer		

Note 1 This table defines the maximum coded data rate for fixed bit rate operation and the maximum elementary stream rate Res(max) for variable bit rate operation, which are indicated by **bit_rate** (see 6.3.3). See also 2.4.2 of ISO/IEC 13818-1.

Note 2 This table defines the maximum permissible data rate for all layers up to and including the stated layer. For multi-layer coding applications, the data rate apportioned between layers is constrained only by the maximum rate permitted for a given layer as stated in this table.

Note 3 1 Mbit = 1 000 000 bits

Table 8-13. VBV Buffer size requirements (bits)

Level	Layer	Profile				
		Simple	**Main**	**SNR**	**Spatial**	**High**
High	Enh. 2					12 222 464
	Enh. 1					9 781 248
	Base		9 781 248			3 047 424
High-1440	Enh. 2				7 340 032	9 781 248
	Enh. 1				4 882 432	7 340 032
	Base		7 340 032		1 835 008	2 441 216
Main	Enh. 2			-		2 441 216
	Enh. 1			1 835 008		1 835 008
	Base	1 835 008	1 835 008	1 212 416		475 136
Low	Enh. 2			-		
	Enh. 1			475 136		
	Base		475 136	360 448		

Note 1 The buffer size is calculated to be proportional to the maximum allowable bit rate, *rounded down* to the nearest multiple of 16 x 1024 bits. The reference value for scaling is the Main profile, Main level buffer size.

Note 2 This table defines the *total* decoder buffer size required to decode all layers up to and including the stated layer. For multi-layer coding applications, the allocation of buffer memory between layers is constrained only by the maximum size permitted for a given layer as stated in this table.

Note 3 The syntactic element corresponding to this table is **vbv_buffer_size** (see 6.3.3).

The following tables indicate the parameter limits that apply to each layer of a bitstream, and the minimum profile / level of a compliant decoder capable of fully decoding each layer. Each table describes the limits of a single complinace point in the profile / level matrix.

Table 8-14. Detailed specification of layered profiles

Level	Profile		
	SNR	**Spatial**	**High**
High	-	-	See annex E
High-1440	-	Table 8-17	See annex E
Main	Table 8-16	-	See annex E
Low	Table 8-15	-	-

Note: The full specification of High profiles has yet to be determined. The tables found in Annex E are for guidance purposes.

In the following tables, the following notation has been adopted:

<profile abbreviation>@<level abbreviation>

The abbreviations are defined in table 8-14a.

Table 8-14a. Abbreviations for profile and level names

Profile	<profile abbreviation>	Level	<level abbreviation>
Simple	SP	Low	LL
Main	MP	Main	ML
SNR Scalable	SNR	High-1440	H-14
Spatially Scalable	Spatial	High	HP
High	HP		

Table 8-15. SNR profile @ Low level

# Layers	layer_id	Scalable mode	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	MP@LL	MP@LL	4	475 136	MP@LL
2	0	-	MP@LL	MP@LL	3	360 448	MP@LL
2	1	SNR	MP@LL	MP@LL	4	475 136	SNR@LL

Table 8-16. SNR profile @ Main level

# Layers	layer_id	Scalable mode	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	MP@ML	MP@ML	15	1 835 008	MP@ML
2	0	-	MP@ML	MP@ML	10	1 212 416	MP@ML
2	1	SNR	MP@ML	MP@ML	15	1 835 008	SNR@ML

Table 8-17. Spatial profile @ High-1440 level

# Layers	layer_id	Scalable mode	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	MP@H-14	MP@H-14	60	7 340 032	MP@H-14
2	0	-	MP@H-14	MP@H-14	40	4 882 432	MP@H-14
2	1	SNR	MP@H-14	MP@H-14	60	7 340 032	Spatial@H-14
2	0	-	MP@ML	MP@ML	15	1 835 008	MP@ML
2	1	Spatial	MP@H-14	MP@H-14	60	7 340 032	Spatial@H-14
3	0	-	MP@ML	MP@ML	10	1 212 416	MP@ML
3	1	SNR	MP@ML	MP@ML	15	1 835 008	SNR@ML
3	2	Spatial	MP@H-14	MP@H-14	60	7 340 032	Spatial@H-14
3	0	-	MP@ML	MP@ML	15	1 835 008	MP@ML
3	1	Spatial	MP@H-14	MP@H-14	40	4 882 432	Spatial@H-14
3	2	SNR	MP@H-14	MP@H-14	60	7340 032	Spatial@H-14

Table 8-18. Forward compatibility between different profiles and levels

	Decoder										
bitstream	**HP @ HL**	**HP @ H-14**	**HP @ ML**	**Spatial @ H-14**	**SNR @ ML**	**SNR @ LL**	**MP @ HL**	**MP @ H-14**	**MP @ ML**	**MP @ LL**	**SP @ ML**
HP@HL	X										
HP@H-14	X	X									
HP@ML	X	X	X								
Spatial@H-14	X	X		X							
Base layer	X	X	X	X			X	X			
SNR @ML	X	X	X	X	X						
Base layer	X	X	X	X	X		X	X	X		
SNR @LL	X	X	X	X	X	X					
Base layer	X	X	X	X	X	X	X	X	X	X	
MP@HL	X						X				
MP@H-14	X	X		X			X	X			
MP@ML	X	X	X	X	X		X	X	X		
MP@LL	X	X	X	X	X	X	X	X	X	X	X*
SP@ML	X	X	X	X	X		X	X	X		X
ISO/IEC 11172	X	X	X	X	X	X	X	X	X	X	X

X indicates the decoder shall be able to decode the bitstream.

* Note that SP@ML decoders are required to decode MP@LL bitstreams.

Note: For Profiles and Levels which obey a hierarchical structure, it is recommended that each layer of the bitstream should contain the **profile_and_level_indication** of the "simplest" decoder which is capable of successfully decoding that layer of the bitstream. In the case where the **profile_and_level_indication** Escape bit = 0, this will be the numerically largest of the possible valid values of **profile_and_level_indication**.

1

Annex A
Discrete cosine transform

(This annex forms an integral part of this Recommendation | International Standard)

The NxN two dimensional DCT is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

with u, v, x, y = 0, 1, 2, ... N-1

where x, y are spatial coordinates in the sample domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

The input to the forward transform and output from the inverse transform is represented with 9 bits. The coefficients are represented in 12 bits. The dynamic range of the DCT coefficients is [-2048:+2047].

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990. Note that clause 2.3 Std 1180-1990 "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 1180-1990 for visual telephony according to ITU-T Recommendation H.261.

(see Annex G).

Annex B
Variable length code tables

(This annex forms an integral part of this Recommendation | International Standard)

B.1 Macroblock addressing

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

Note: The “macroblock stuffing” entry that is available in ISO/IEC11172-2 is not available in this specification.

B.2 Macroblock type

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

Table B-2 — Variable length codes for macroblock_type in I-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
1	0	0	0	0	1	0	Intra	0
01	1	0	0	0	1	0	Intra, Quant	0

Table B-3 — Variable length codes for macroblock_type in P-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
1	0	1	0	1	0	0	MC, Coded	0
01	0	0	0	1	0	0	No MC, Coded	0
001	0	1	0	0	0	0	MC, Not Coded	0
0001 1	0	0	0	0	1	0	Intra	0
0001 0	1	1	0	1	0	0	MC, Coded, Quant	0
0000 1	1	0	0	1	0	0	No MC, Coded, Quant	0
0000 01	1	0	0	0	1	0	Intra, Quant	0

Table B-4 — Variable length codes for macroblock_type in B-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
10	0	1	1	0	0	0	Interp, Not Coded	0
11	0	1	1	1	0	0	Interp, Coded	0
010	0	0	1	0	0	0	Bwd, Not Coded	0
011	0	0	1	1	0	0	Bwd, Coded	0
0010	0	1	0	0	0	0	Fwd, Not Coded	0
0011	0	1	0	1	0	0	Fwd, Coded	0
0001 1	0	0	0	0	1	0	Intra	0
0001 0	1	1	1	1	0	0	Interp, Coded, Quant	0
0000 11	1	1	0	1	0	0	Fwd, Coded, Quant	0
0000 10	1	0	1	1	0	0	Bwd, Coded, Quant	0
0000 01	1	0	0	0	1	0	Intra, Quant	0

Table B-5 — Variable length codes for macroblock_type in I-pictures with spatial scalability.

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
1	0	0	0	1	0	0	Coded, Compatible	4
01	1	0	0	1	0	0	Coded, Compatible, Quant	4
0011	0	0	0	0	1	0	Intra	0
0010	1	0	0	0	1	0	Intra, Quant	0
0001	0	0	0	0	0	0	Not Coded, Compatible	4

Table B-6 — Variable length codes for macroblock_type in P-pictures with spatial scalability.

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
10	0	1	0	1	0	0	MC, Coded	0
011	0	1	0	1	0	1	MC, Coded, Compatible	1,2,3
0000 100	0	0	0	1	0	0	No MC, Coded	0
0001 11	0	0	0	1	0	1	No MC, Coded, Compatible	1,2,3
0010	0	1	0	0	0	0	MC, Not Coded	0
0000 111	0	0	0	0	1	0	Intra	0
0011	0	1	0	0	0	1	MC, Not coded, Compatible	1,2,3
010	1	1	0	1	0	0	MC, Coded, Quant	0
0001 00	1	0	0	1	0	0	No MC, Coded, Quant	0
0000 110	1	0	0	0	1	0	Intra, Quant	0
11	1	1	0	1	0	1	MC, Coded, Compatible, Quant	1,2,3
0001 01	1	0	0	1	0	1	No MC, Coded, Compatible, Quant	1,2,3
0001 10	0	0	0	0	0	1	No MC, Not Coded, Compatible	1,2,3
0000 101	0	0	0	1	0	0	Coded, Compatible	4
0000 010	1	0	0	1	0	0	Coded, Compatible, Quant	4
0000 011	0	0	0	0	0	0	Not Coded, Compatible	4

Table B-7 — Variable length codes for macroblock_type in B-pictures with spatial scalability.

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
10	0	1	1	0	0	0	Interp, Not coded	0
11	0	1	1	1	0	0	Interp, Coded	0
010	0	0	1	0	0	0	Back, Not coded	0
011	0	0	1	1	0	0	Back, Coded	0
0010	0	1	0	0	0	0	For, Not coded	0
0011	0	1	0	1	0	0	For, Coded	0
0001 10	0	0	1	0	0	1	Back, Not Coded, Compatible	1,2,3
0001 11	0	0	1	1	0	1	Back, Coded, Compatible	1,2,3
0001 00	0	1	0	0	0	1	For, Not Coded, Compatible	1,2,3
0001 01	0	1	0	1	0	1	For, Coded, Compatible	1,2,3
0000 110	0	0	0	0	1	0	Intra	0
0000 111	1	1	1	1	0	0	Interp, Coded, Quant	0
0000 100	1	1	0	1	0	0	For, Coded, Quant	0
0000 101	1	0	1	1	0	0	Back, Coded, Quant	0
0000 0100	1	0	0	0	1	0	Intra, Quant	0
0000 0101	1	1	0	1	0	1	For, Coded, Compatible, Quant	1,2,3
0000 0110 0	1	0	1	1	0	1	Back, Coded, Compatible, Quant	1,2,3
0000 0111 0	0	0	0	0	0	0	Not Coded, Compatible	4
0000 0110 1	1	0	0	1	0	0	Coded, Compatible, Quant	4
0000 0111 1	0	0	0	1	0	0	Coded, Compatible	4

Table B-8 — Variable length codes for macroblock_type in I-pictures, P-pictures and B-pictures with SNR scalability.

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	spatial_temporal_weight_code_flag	Description	permitted spatial_temporal_weight_classes
1	0	0	0	1	0	0	Coded	0
01	1	0	0	1	0	0	Coded, Quant	0
001	0	0	0	0	0	0	Not Coded	0

Note There is no differentiation between picture types, since macroblocks are processed identically in I, P and B-pictures. The "Not coded" type is needed, since skipped macroblocks are not allowed at beginning and end of a slice.

1 ## B.3 Macroblock pattern

2 **Table B-9 --- 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	0000 0000 1	0 (NOTE)
NOTE — This entry shall not be used with 4:2:0 chrominance structure			

3

B.4 Motion vectors

Table B-10 --- Variable length codes for motion_code

Variable length code	motion_code[r][s][t]
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

Table B-11 — Variable length codes for dmvector[t]

code	value
11	-1
0	0
10	1

B.5 DCT coefficients

Table B-12 --- Variable length codes for dct_dc_size_luminance

Variable length code	dct_dc_size_luminance
100	0
00	1
01	2
101	3
110	4
1110	5
1111 0	6
1111 10	7
1111 110	8
1111 1110	9
1111 1111 0	10
1111 1111 1	11

Table B-13 --- Variable length codes for dct_dc_size_chrominance

Variable length code	dct_dc_size_chrominance
00	0
01	1
10	2
110	3
1110	4
1111 0	5
1111 10	6
1111 110	7
1111 1110	8
1111 1111 0	9
1111 1111 10	10
1111 1111 11	11

1 **Table B-14 --- DCT coefficients Table zero**

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 the first (DC) coefficient in the block

NOTE3 - This code shall be used for all other coefficients

2

1 **Table B-14 --- DCT coefficients Table zero (continued)**

Variable length code (NOTE)	run	level
0000 0001 1101 s	0	8
0000 0001 1000 s	0	9
0000 0001 0011 s	0	10
0000 0001 0000 s	0	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

Table B-14 --- DCT coefficients Table zero (continued)

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 B-14 --- DCT coefficients Table zero (concluded)

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.		

1 **Table B-15 --- DCT coefficients Table one**

Variable length code (NOTE)	**run**	**level**
0110	End of Block	
10s	0	1
010 s	1	1
110 s	0	2
0010 1 s	2	1
0111 s	0	3
0011 1 s	3	1
0001 10 s	4	1
0011 0 s	1	2
0001 11 s	5	1
0000 110 s	6	1
0000 100 s	7	1
1110 0 s	0	4
0000 111 s	2	2
0000 101 s	8	1
1111 000 s	9	1
0000 01	Escape	
1110 1 s	0	5
0001 01 s	0	6
1111 001 s	1	3
0010 0110 s	3	2
1111 010 s	10	1
0010 0001 s	11	1
0010 0101 s	12	1
0010 0100 s	13	1
0001 00 s	0	7
0010 0111 s	1	4
1111 1100 s	2	3
1111 1101 s	4	2
0000 0010 0 s	5	2
0000 0010 1 s	14	1
0000 0011 1 s	15	1
0000 0011 01 s	16	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.		

2

1 **Table B-15 --- DCT coefficients Table one (continued)**

Variable length code (NOTE)	**run**	**level**
1111 011 s	0	8
1111 100 s	0	9
0010 0011 s	0	10
0010 0010 s	0	11
0010 0000 s	1	5
0000 0011 00 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
1111 1010 s	0	12
1111 1011 s	0	13
1111 1110 s	0	14
1111 1111 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

Table B-15 --- DCT coefficients Table one (continued)

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.		

1

2 **Table B-15 --- DCT coefficients Table one (concluded)**

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.		

3

4 **Table B-16 --- Encoding of run and level following an ESCAPE code**

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

fixed length code	signed_level
1000 0000 0001	-2047
1000 0000 0010	-2046
...	...
1111 1111 1111	-1
0000 0000 0000	forbidden
0000 0000 0001	+1
...	...
0111 1111 1111	+2047

5

Annex C
Video buffering verifier

(This annex forms an integral part of this Recommendation | International Standard)

Constant rate coded video bitstreams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in this clause. In variable bit-rate operation this annex is superseded by the STD models defined in ISO/IEC 13818-1. If the system coding layer defined is ISO/IEC 13818-1 is present, the STD model supersedes the VBV model.

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 bitrate that is being used. Coded data is removed from the buffer as defined below. It is required that a bitstream that conforms to this specification shall not cause the VBV to overflow. When low_delay equals 0, the bitstream shall not cause the VBV buffer to underflow. When low_delay is 1, VBV buffer may underflow, as specified in C.7 and C.8.

All the arithmetic in Annex C are done with real-value, so that no rounding error can propagate. For example, the number of bits in the VBV buffer is not necessarily integer.

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

C.2 The VBV has a input buffer of size B, where B is the vbv_buffer_size coded in the sequence header and sequence extension if any.

C.3 The VBV is initially empty. After filling the input buffer with all the data that precedes the first picture start code of the sequence and the picture start code itself, the input buffer is filled from the bitstream for the time specified by the vbv_delay field in the picture header.

C.4 Starting at this time, the VBV buffer is examined at successive times defined in C.9 to C.12. C.5 to C.8 defines the actions to be taken at each time the VBV is examined.

C.5 This clause defines a requirement on all video bitstreams.

At the time the VBV buffer is examined *before* removing any picture data and immediately *after* this picture data is removed, when this applies (cf. C.6 and C.7), the number of bits in the buffer shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by vbv_buffer_size.

For the purpose of this annex, picture data includes picture_data() as well as any header(s), user data and stuffing that immediately precedes it, and any trailing stuffing bits or bytes immediately following picture_data() if they are present in the VBV buffer at the time of examination. For the first coded picture in the video sequence, any zero bit or byte stuffing immediately preceding the sequence_header() is also included in the picture data.

To meet this requirement the following inequality must be satisfied:

$$d_{n+1} > B_n + (t_{n+2} - t_n) * R - B$$

Real-valued arithmetic is used in this inequality.

where:

d_n is the picture data of n’th coded picture, measured in bits

B_n is the buffer occupancy (measured in bits) just *after* the n’th coded picture has been removed from the buffer

R = is the bitrate measured in bits/s. The full precision of the bitrate rather than the rounded value encoded by the bit_rate and bit_rate_extension fields shall be used by the encoder in the VBV model.

t_n is the time when the n'th coded picture is removed from the VBV buffer (measured in seconds, with full precision)

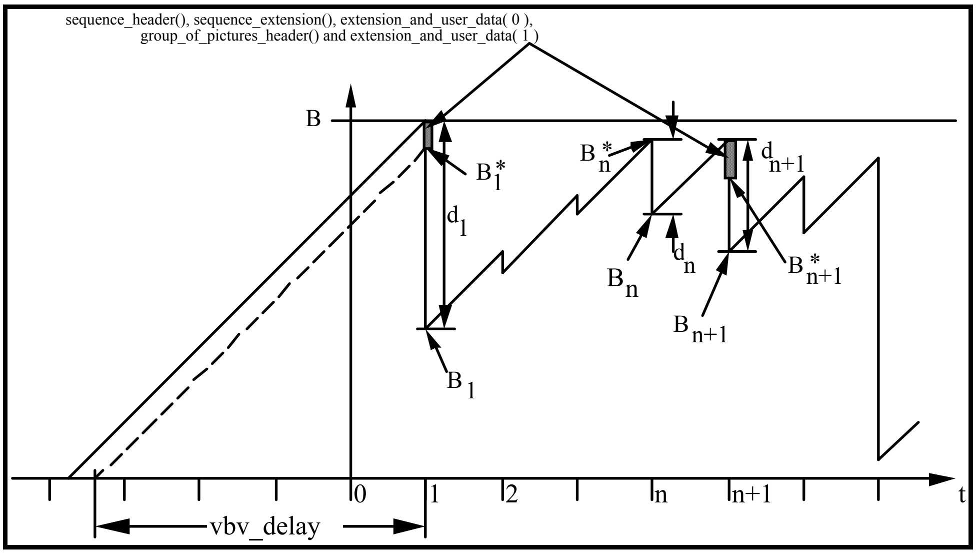


Figure C-1. VBV Buffer Occupancy

C.6 This clause defines a requirement on the video bitstreams when the low_delay flag is equal to 0.

At each time the VBV buffer is examined and before any bits are removed, all of the data for the picture which (at that time) has been in the buffer longest shall be present in the buffer.

To meet this requirement the following inequality must be satisfied:

$$d_n <= B^*_n$$

where:

d_n is the picture data of n'th coded picture, measured in bits

B^*_n is the buffer occupancy (measured in bits) just *before* the n'th coded picture has been removed from the buffer

VBV buffer underflow (that would happen if all the data of the picture were not present) shall not occur when the low_delay flag is equal to 0.

C.7 This clause only applies when the low_delay flag is equal to 1.

This clause describes the VBV action in case underflow would occur, i.e. when the VBV buffer does not contain a complete picture at the time it is examined.

This is the case when the following inequality is satisfied:

$$d_n > B^*_n$$

where:

d_n is the picture data of n'th coded picture, measured in bits

B^*_n is the buffer occupancy (measured in bits) just *before* the n'th coded picture has been removed from the buffer

The following procedure applies in this case:

The buffer is re-examined at intervals of 2 field-periods, until the complete picture data is present in the VBV input buffer. When this is the case, the number of bits in the buffer must be less than B. At that point normal operation of the VBV resumes and C.5 applies.

The value of temporal_reference is not affected when this applies. For example, when no B frame picture are present, temporal_reference is incremented by 1 for each frame whether VBV buffer would underflow or not.

C.8 This clause is informative only.

The situation where VBV buffer would underflow (see C.7) can happen when low-delay applications must transmit occasionally large pictures, for example in case of scene-cuts.

Decoding such bitstreams will cause the display process associated with a decoder to repeat a previously decoded field or frame until normal operation of the VBV can resume. This process is sometimes referred to as the occurrence of "skipped pictures". Note that his situation should normally not occur except occasionally and only in low-delay bitstreams (i.e., when low_delay is equal to 1).

C.9 This clause defines the time intervals between successive examination of the VBV buffer in the case where progressive_sequence equals to 1 and low_delay equals to 0. In this case, the frame reordering delay always exists and B pictures can occur.

The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a multiple of T, where T is the inverse of the frame rate.

If the n'th picture is a B-picture with repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to T.

If the n'th picture is a B-Picture with repeat_first_field equals to 1 and top_field_first equals 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a B-Picture with repeat_first_field equals to 1 and top_field_first equals 1, then $t_{n+1} - t_n$ is equal to 3*T.

If the n'th picture is a P-Picture or I-Picture and if the previous P-Picture or I-Picture has repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to T.

If the n'th picture is a P-Picture or I-Picture and if the previous P-Picture or I-Picture has repeat_first_field equals to 1 and top_field_first equal to 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a P-Picture or I-Picture and if the previous P-Picture or I-Picture has repeat_first_field equals to 1 and top_field_first equal to 1, then $t_{n+1} - t_n$ is equal to 3*T.

If $t_{n+1}-t_n$ cannot be determined with any of the previous paragraphs because the previous P- or I-Picture does not exist (which can occur at the beginning of a sequence), then the time interval is arbitrary with the following restrictions:

The time interval between removing one frame (or the first field of a frame) and removing the next frame can be arbitrarily defined equal to T, 2*T or 3*T. The value of vbv_delay can be used to determine which value was used.

C.10 This clause defines the time intervals between successive examination of the VBV buffer in the case where progressive_sequence equals to 1 and low_delay equals to 1. In this case the sequence contains no B-Pictures and there is no frame reordering delay.

The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a multiple of T, where T is the inverse of the frame rate.

If the n'th picture is a P-Picture or I-Picture with repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to T.

If the n'th picture is a P-Picture or I-Picture with repeat_first_field equals to 1 and top_field_first equals to 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a P-Picture or I-Picture with repeat_first_field equals to 1 and top_field_first equals to 1, then $t_{n+1} - t_n$ is equal to 3*T.

C.11 This clause defines the time intervals between successive examination of the VBV buffer in the case where progressive_sequence equals to 0 and low_delay equals to 0. In this case, the frame reordering delay always exists and B pictures can occur.

The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a multiple of T, where T is the inverse of two times the frame rate.

If the n'th picture is a *frame-structure* coded B-frame with repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a *frame-structure* coded B-frame with repeat_first_field equals to 1, then $t_{n+1} - t_n$ is equal to 3*T.

If the n'th picture is a *field-structure* B-picture (B-field picture), then $t_{n+1} - t_n$ is equal to T.

If the n'th picture is a *frame-structure* coded P-frame or coded I-Frame and if the previous coded P-Frame or coded I-Frame has repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a *frame-structure* coded P-Frame or coded I-Frame and if the previous coded P-Frame or coded I -Frame has repeat_first_field equals to 1, then $t_{n+1} - t_n$ is equal to 3*T.

If the n'th picture is the *first* field of a *field-structure* coded P-frame or coded I-Frame, then $t_{n+1} - t_n$ is equal to T.

If the n'th picture is the *second* field of a *field-structure* coded P-Frame or coded I-Frame and if the previous coded P-Frame or coded I-Frame is using field-structure or has *repeat_first_field* equals to 0, then $t_{n+1} - t_n$ is equal to (2*T - T).

If the n'th picture is the *second* field of a *field-structure* coded P-Frame or coded I-Frame and if the previous coded P-Frame or coded I-Frame is using frame-structure and has *repeat_first_field* equals to 1, then $t_{n+1} - t_n$ is equal to (3*T - T).

If $t_{n+1}-t_n$ cannot be determined with any of the previous paragraphs because the previous coded P- or I frame does not exist (which can occur at the beginning of a sequence), then the time interval is arbitrary with the following restrictions:

The time interval between removing one frame (or the first field of a frame) and removing the next frame (or the first field of a frame) can be arbitrarily defined equal to 2*T or 3*T. The value of vbv_delay can be used to determine which value was used.

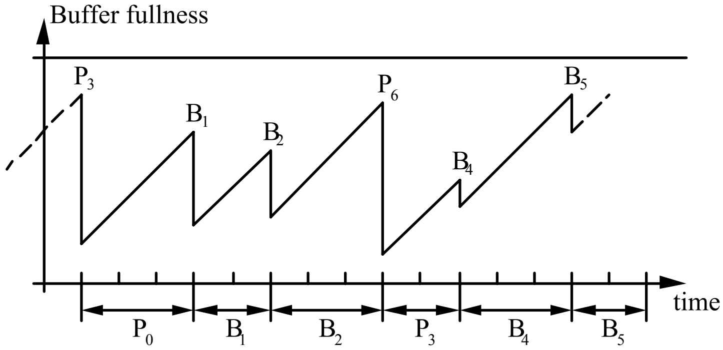


Figure C-2

Figure C-2 shows the VBV in a simple case with only frame-pictures. Frames P_0, B_2 and B_4 have a display duration of 3 fields.

C.12 This clause defines the time intervals between successive examination of the VBV buffer in the case where progressive_sequence equals to 0 and low_delay equals to 1. In this case the sequence contains no B-Pictures and there is no frame reordering delay.

The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a multiple of T, where T is the inverse of two times the frame rate.

If the n'th picture is a *frame-structure* coded P-Frame or coded I-Frame with repeat_first_field equals to 0, then $t_{n+1} - t_n$ is equal to 2*T.

If the n'th picture is a *frame-structure* coded P-Frame or coded I-Frame with repeat_first_field equals to 1, then $t_{n+1} - t_n$ is equal to 3*T.

If the n'th picture is a *field-structure* coded P-Frame or coded I-Frame, then $t_{n+1} - t_n$ is equal to T.

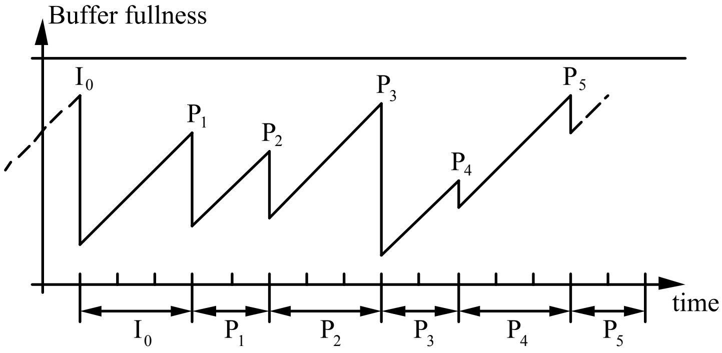


Figure C-3

Figure C-3 shows the VBV in a simple case with only frame-pictures. Frames I_0, P_2 and P_4 have repeat_first_field equals to 1.

Annex D
Features supported by the algorithm

(This annex does not form an integral part of this Recommendation | International Standard)

D.1 Overview

The following non-exhaustive list of features is included in this specification:

1) Different chrominance sampling formats (i.e., 4:2:0, 4:2:2 and 4:4:4) can be represented.
2) Video in both the progressive and interlaced scan formats can be encoded.
3) The decoder can use 3:2 pull down to represent a ~24 fps film as ~30 fps video.
4) The displayed video can be selected by a movable pan-scan window within a larger raster.
5) A wide range of picture qualities can be used.
6) Both constant and variable bitrate channels are supported.
7) A low delay mode for face-to-face applications is available.
8) Random access (for DSM, channel acquisition, and channel hopping) is available.
9) ISO/IEC 11172-2 bitstreams are decodable.
10) Bitstreams for high and low (hardware) complexity decoders can be generated.
11) Editing of encoded video is supported.
12) Fast-forward and fast-reverse playback recorded bitstreams can be implemented.
13) The encoded bitstream is resilient to errors.

D.2 Video Formats

D.2.1 Sampling Formats and Colour

This specification video coding supports both interlaced and progressive video. The respective indication is provided with a *progressive_sequence* flag transmitted in the Sequence Extension code.

Allowed raster sizes are between 1 and (2^14 - 1) luminance samples each of the horizontal and vertical directions. The video is represented in a luminance/chrominance colour space with selectable colour primaries. The chrominance can be sampled in either the 4:2:0 (half as many samples in the horizontal and vertical directions), 4:2:2 (half as many samples in the horizontal direction only). Furthermore, application specific sample aspect ratios and image aspect ratios are flexibly supported. A *chroma_format* parameter is contained in the Sequence Extension code.

Sample aspect ratio information is provided by means of aspect_ratio_information and (optional) display_horizontal_size and display_vertical_size in the sequence_display_extension(). Examples of appropriate values for signals sampled in accordance with ITU-R Rec. 601 are given in Table D-1.

Table D-1. Example display size values.

Signal Format	display_horizontal_size	display_vertical_size
525-line	711	483
625-line	702	575

This specification implements tools to support 4:4:4 chrominance, for possible future use. However, this is currently not supported in any profile.

D.2.2 Movie Timing

A decoder can implement 3:2 pull down when a sequence of progressive pictures is encoded. Each encoded movie picture can independently specify whether it is displayed for two or three video field periods, so "irregular" 3:2 pull down source material can be transmitted as progressive video. Two flags, *top_field_first* and *repeat_first_field*, are transmitted with the Picture Coding Extensions and adequately describe the necessary display timing.

D.2.3 Display Format Control

The display process converts a sequence of digital frames (in the case of progressive video) or fields (in the case of interlaced video) to output video. It is not a normative part of the this standard. The video syntax of this specification does communicate certain display parameters for use in reconstructing the video. Optional information (in the sequence display extension) specifies the chromaticities, the display primaries, the opto-electronic transfer characteristics (e.g., the value of gamma) and the RGB-to-luminance/chrominance conversion matrix.

Moreover, a display window within the encoded raster may be defined as, e.g., in the case of pan and scan. Alternatively the encoded raster may be defined as a window on a large area display device. In the case of pan-scan the position of the window representing the displayed region of a larger picture can be specified on a field-by-field basis. It is specified in the Picture display extension described in 6.3.12. A typical use for the pan-scan window is to describe the "important" 4:3 aspect ratio rectangle within a 16:9 video sequence. Similarly, in the case of small encoded pictures on a large display the size of the display and the position of the window within that display may be specified.

D.2.5 Transparent coding of composite video

Decoding from PAL/NTSC before transmission and recoding to PAL/NTSC after transmission of composite source signals in non low quality applications, such as contribution and distribution, requires a precise reconstruction of the carrier amplitude and phase reference signal (and v-axis switch for PAL).

The input format can be indicated in the sequence header using the video_format bits. Possible source formats are: PAL, NTSC, SECAM and MAC. Reconstruction of the carrier signal is possible by using the carrier parameters: *V_axis, field_sequence, sub_carrier, burst_amplitude* and *sub_carrier phase* that are enabled by setting the *composite_display_flag* in the picture header.

D.3 Picture Quality

High picture quality is provided according to the bitrate used. Provision for very high picture quality is made by sufficiently high bitrate limits relating to a certain level in a particular profile. High chrominance band quality can be achieved by using 4:2:2 chrominance

Quantiser matrices can be downloaded and used with a small a quantiser_scale_code to achieve near lossless coding.

Moreover, scalable coding with flexible bitrate allows for service or quality hierarchy and graceful degradation. E.g., decoding a subset of the bitstream carrying a lower resolution picture allows for decoding this signal in a low-cost receiver with related quality; decoding the complete bitstream allows to obtain the high overall quality.

Furthermore, operation at low bitrates can be accommodated by using low frame rates (by either pre-processing before coding or frame skipping indicated by the *temporal_reference* in the picture header) and low spatial resolution.

D.4 Data Rate Control

The number of transmitted bits per unit time, which is selectable in a wide range, may be controlled in two ways, which are both supported by this specification. A *bit_rate* description is transmitted with the Sequence Header Code.

For constant bitrate (CBR) coding, the number of transmitted bits per unit time is constant on the channel. Since the encoder output rate generally varies depending on the picture content, it shall regulate the rate constant by buffering etc. In CBR, picture quality may vary depending on its content.

The other mode is the variable bitrate (VBR) coding, in which case the number of transmitted bits per unit time may vary on the channel under some constriction. VBR is meant to provide constant quality coding. A model for VBR application is near-constant-quality coding over B-ISDN channels subject to Usage Parameter Control (UPC).

D.5 Low Delay Mode

A low encoding and decoding delay mode is accommodated for real-time video communications such as visual telephony, video-conferencing, monitoring. Total encoding and decoding delay of less than 150 milliseconds can be achieved for low delay mode operation of this specification. Setting the *low_delay flag* in the Sequence Header code defines a low delay bitstream.

The total encoding and decoding delay can be kept low by generating a bitstream which does not contain B-pictures. This prevents frame reordering delay. By using dual-prime prediction for P-frames the picture quality can still be high.

A low buffer occupancy for both encoder and decoder is needed for low delay. Large coded pictures should be avoided by the encoder. By using intra update on the basis of one or more slices per frame (intra slices) instead of intra frames this can be accommodated.

In case of exceeding, for low delay operation, the desired number of bits per frame the encoder can skip one or more frames. This action is indicated, to the decoder, by the state of the VBV buffer or the STD buffer; i.e. underflow in the decoder buffer indicates that the encoder has skipped pictures.

D.6 Random Access/Channel Hopping

The syntax of this specification supports random access and channel hopping. Sufficient random access/channel hopping functionality is possible by encoding suitable random access points into the bitstream without significant loss of image quality.

Random access is an essential feature for video on a storage medium. It requires that any picture can be accessed and 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. In this specification access points are provided by sequence_header() and this is then followed by intra information (picture data that can be decoded without access to previously decoded pictures). A spacing of two random access points per second can be achieved without significant loss of picture quality.

Channel hopping is the similar situation in transmission applications such as broadcasting. As soon as a new channel has been selected and the bitstream of the selected channel is available to the decoder, the next data entry, i.e. random access point has to be found to start decoding the new program in the manner outlined in the previous paragraph.

D.7 Scalability

The syntax of this specification supports bitstream scalability. To accommodate the diverse functionality requirements of the applications envisaged by this specification a number of bitstream scalability tools have been developed:

- **SNR scalability** mainly targets for applications which require graceful degradation.
- **Chroma simulcast** targets at applications with high chrominance quality requirements.
- **Data partitioning** is primarily targeted for cell loss resilience in ATM networks.
- **Temporal scalability** is a method suitable for interworking of services using high temporal resolution progressive video formats. Also suitable for high quality graceful degradation in the presence of channel errors.
- **Spatial scalability** allows multiresolution coding technique suitable for video service interworking applications. This tool can also provide coding modes to achieve compatibility with existing coding standards, i.e. ISO/IEC 11172-2, at the lower layer.

D.7.1 Use of SNR scalability at a single spatial resolution

The aim of SNR scalability is primarily to provide a mechanism for transmission of a two layer service, these two layers providing the same picture resolution but different quality level. For example, the transmission of service with two different quality levels is expected to become useful in the future for some TV broadcast applications, especially when very good picture quality is needed for large size display receivers. The sequence is encoded into two bitstreams called lower and enhancement layer bitstreams. The lower layer bitstream can be decoded independently from the enhancement layer bitstream. The lower layer, at 3 to 4 Mbit/s, would provide a picture quality equivalent to the current NTSC/PAL/SECAM quality. Then, by using both the lower and the enhancement layer bitstreams, an enhanced decoder can deliver a picture quality subjectively close to the studio quality, with a total bitrate of 7 to 12 Mbit/s.

D.7.1.1 Additional features

D.7.1.1.1 Error resilience

As described in D.12 the SNR scalable scheme can be used as a mechanism for error resilience. If the two layer bitstreams are received with different error rate, the lower layer, better protected, stands as a good substitute to fall back on, if the enhancement layer is damaged.

D.7.1.1.2 Chroma simulcast

The SNR scalable syntax can be used in a chroma simulcast system. The goal of such a scheme would be to provide a mechanism for simultaneous distribution of services with the same luminance resolution but different chrominance sampling format (e.g. 4:2:0 in the lower layer and 4:2:2, when adding the enhancement layer and the simulcast chrominance components) for applications which would require such a feature. The SNR scalable enhancement layer contains some luminance refinement. The 4:2:2 chrominance is sent in simulcast. Only chrominance DC is predicted from the lower layer. The combination of both layer luminance and of the 4:2:2 chrominance constitutes the high quality level.

D.7.1.2 SNR scalable encoding process

D.7.1.2.1 Description

In the lower layer, the encoding is similar to the non scalable situation in terms of decisions, adaptive quantisation, buffer regulation. The intra or error prediction macroblocks are DCT transformed. The coefficients are then quantised using a first rather coarse quantiser. The quantised coefficients are then VLC coded and sent together with the required side information (macroblock_type, motion vectors, coded_block_pattern()).

In parallel, the quantised DCT coefficients coming from the lower layer, are dequantised. The residual error between the coefficients and the dequantised coefficients is then re-quantised, using a second finer quantiser. The resulting refinement coefficients are VLC coded and form the additional enhancement layer, together with a marginal amount of side information (quantiser_scale_code, coded_block_pattern()...). The non-intra VLC table is used for all the coefficients in the enhancement layer, since the transmitted signal is of differential nature.

D.7.1.2.2 A few important remarks

Since the prediction is the same for both layers, it is recommended to use the refined images in the motion estimation loop (e.g. the images obtained by the conjunction of the lower and the enhancement layer). Thus, there is a drift between the prediction signal used at the encoder side and what the low level decoder can get as a prediction. This drift does accumulate from P-picture to P-picture and is reset to zero at each I-Picture. However the drift has been found to have little visual effect intra pictures every 15 pictures or so.

Since the enhancement layer only contains refinement coefficients, the needed overhead is quite reduced: most of the information about the macroblocks (macroblock types, motion vectors...) are included in the lower layer. Therefore the syntax of this stream is very much simplified:

- the macroblock type table only indicates if the quantiser_scale_code in the enhancement layer has changed and if the macroblock is NOT-CODED (for first and last macroblock of the slices), which amounts to three VLC words.

- quantiser_scale_code in the enhancement layer is sent if the value has changed.

- coded_block_pattern() is transmitted for all coded macroblocks.

All NON-CODED macroblocks that are not at the beginning or end of a slice are skipped, since the overhead information can be deduced from the lower layer.

It is recommended to use different weighting matrices for the lower and the enhancement layer. Some better results are obtained when the first quantisation is steeper than the second one. However it is recommended not to quantise too coarsely the DCT coefficient that corresponds to the interlace motion, to avoid juddering effects.

D.7.2 Multiple resolution scalability bitstreams using SNR scalability

The aim of resolution scalability is to decode the base layer video suitable for display at reduced spatial resolution. In addition it is desirable to implement a decoder with reduced complexity for this purpose. This functionality is useful for applications where the receiver display is either not capable or willing to display the full spatial resolution supported by both layers and for applications where software decoding is targeted. The method described in this clause uses the SNR Scalability syntax outlined in clause 7 to transmit the video in two layers. Note that none of the options suggested in this clause changes the structure of the highest resolution decoder, which remains identical to the one outlined in Figure 7-14. The bitstream generated on both layers is compatible with the HIGH profile. However, the base layer decoder could be implemented differently with reduced implementation complexity suitable to software decoding.

D.7.2.1 Decoder Implementation

In decoding to a smaller spatial resolution, an inverse DCT of reduced size could be used when decoding the base layer. The frame memory requirement in the decoder MC loop would also be reduced accordingly.

If the bitstream of the two SNR Scalability layers was generated with only one MC loop at the encoder the base video will be subject to drift. This drift may or may not be acceptable depending on the application. Image quality will, to a large extend, depend on the sub-sample accuracy used for motion compensation in the decoder. It is possible to use the full precision motion vector as transmitted in the base layer for motion compensation with a sub-sample accuracy comparable to that of the higher layer. Drift can be minimised by using advanced sub-sample interpolation filters (see [12], [13] and [16] in Annex G).

D.7.2.2 Encoder Implementation

It is possible to tailor the base layer SNR Scalability bitstream to the particular requirements of the resolution scaled decoder. A smaller DCT size can be more easily supported by only transmitting the appropriate DCT-coefficients belonging to the appropriate subset in the base layer bitstream.

Finally it is possible to support a drift-free decoding at lower resolution scale by incorporating more than one MC loop in the encoder scheme. An identical reconstruction process is used in the encoder and decoder .

D.7.3 Bitrate allocation in data partitioning

Data partitioning allows splitting a bitstream for increased error resilience when two channels with different error performance are available. It is often required to constrain the bitrate of each partition. This can be achieved at the encoder by adaptively changing priority breakpoint at each slice.

The encoder can use two virtual buffers for the two bitstreams, and implement feedback rate control by picking a priority breakpoint that approximately meets the target rate for each channel. Difference between target and actual rates is used to revise the target for the next frame in a feedback loop.

It is desirable to vary the bitrate split from frame to frame for higher error resilience. Typically, I-pictures benefit from having more of the data in partition 0 than the P-pictures while B-pictures could be placed entirely in partition 1.

D.7.4 Temporal scalability

A two layer temporally scalable coding structure consisting of a base and an enhancement layer is shown in Figure D-1. Consider video input at full temporal rate to temporal demultiplexer; in our example it is temporally demultiplexed to form two video sequences, one input to the base layer encoder and the other input to the enhancement layer encoder. The base layer encoder is a non hierarchical encoder operating at half temporal rate, the enhancement layer encoder is like a MAIN profile encoder and also operates at half temporal rate except that it uses base layer decoded pictures for motion compensated prediction. The encoded bitstreams of base and enhancement layers are multiplexed as a single stream in the systems multiplexer. The systems demultiplexer extracts two bitstreams and inputs corresponding bitstreams to base and enhancement layer decoders. The output of the base layer decoder can be shown standalone at half temporal rate or after multiplexing with enhancement layer decoded frames and shown at full temporal rate.

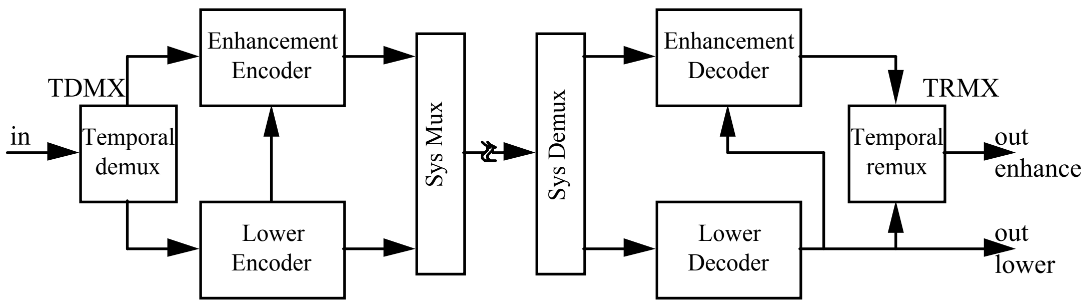


Figure D-1. A two layer codec structure for temporal scalability

The following forms of temporal scalability are supported and are expressed as higher layer: base layer-to-enhancement layer picture formats.

1. Progressive: progressive-to-progressive Temporal Scalability

2. Progressive: interlace-to-interlace Temporal Scalability

3. Interlace: interlace-to-interlace Temporal Scalability

D.7.4.1 Progressive: progressive-to-progressive Temporal Scalability

Assuming progressive video input, if it is necessary to code progressive- format video in base and enhancement layers, the operation of *temporal demux* may be relatively simple and involve temporal demultiplexing of input frames into two progressive sequences; The operation of *temporal remux* is inverse, i.e., it performs remultiplexing of two progressive sequences to generate full temporal rate progressive output. See Figure D-2.

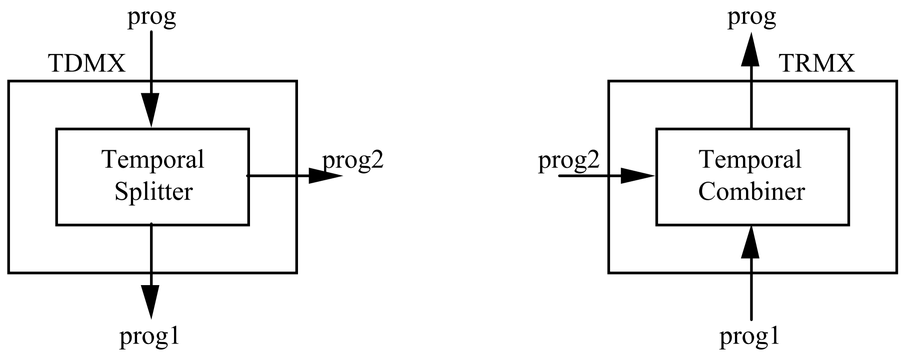


Figure D-2. Temporal demultiplexer and remultiplexer for progressive: progressive-to-progressive temporal scalability

D.7.4.2 Progressive: interlace-to-interlace temporal scalability

Again, assuming full temporal rate progressive video input, if it is necessary to code interlaced format video in base layer, the operation of *temporal demux* may involve progressive to two interlace conversion; this process involves extraction of a normal interlaced- and a complementary interlaced sequence from progressive input video. The operation of *temporal remux* is inverse, i.e., it performs two interlace to progressive conversion to generate full temporal rate progressive output. Figure D-3 and Figure D-4 show operations required in progressive to two interlace and two interlace to progressive conversion.

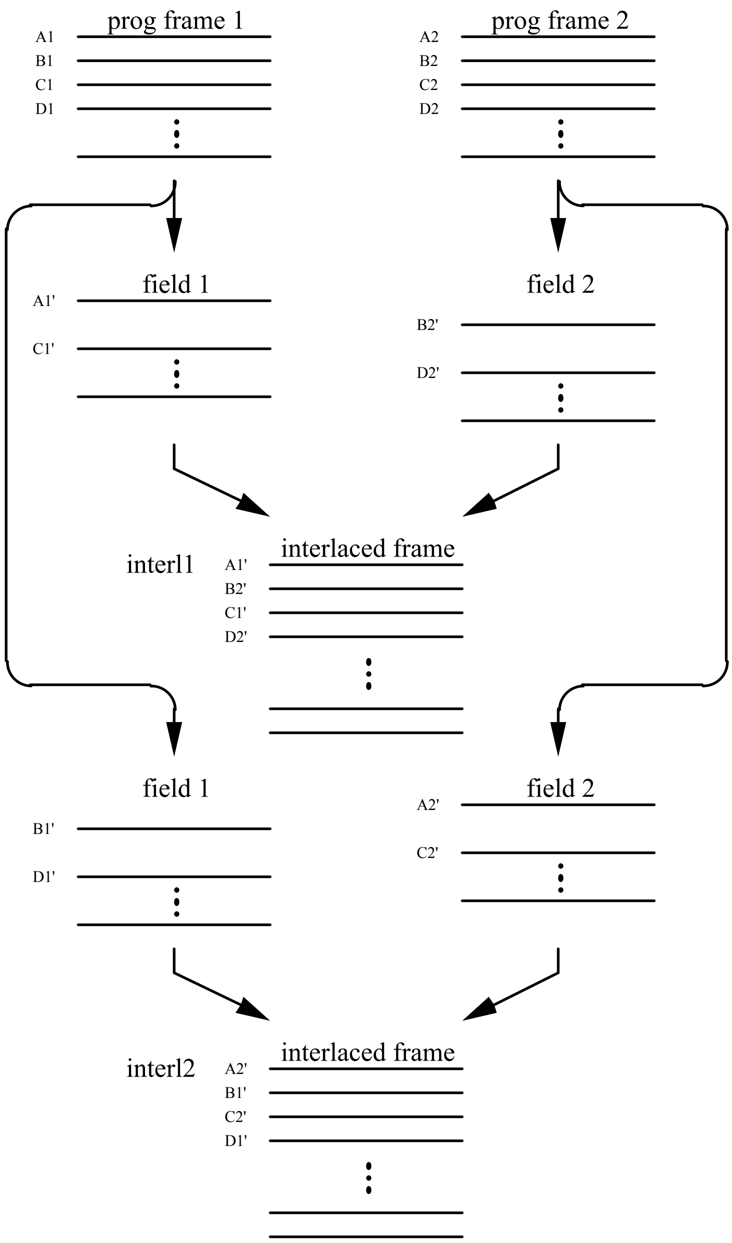


Figure D-3. Progressive to two interlace conversion.

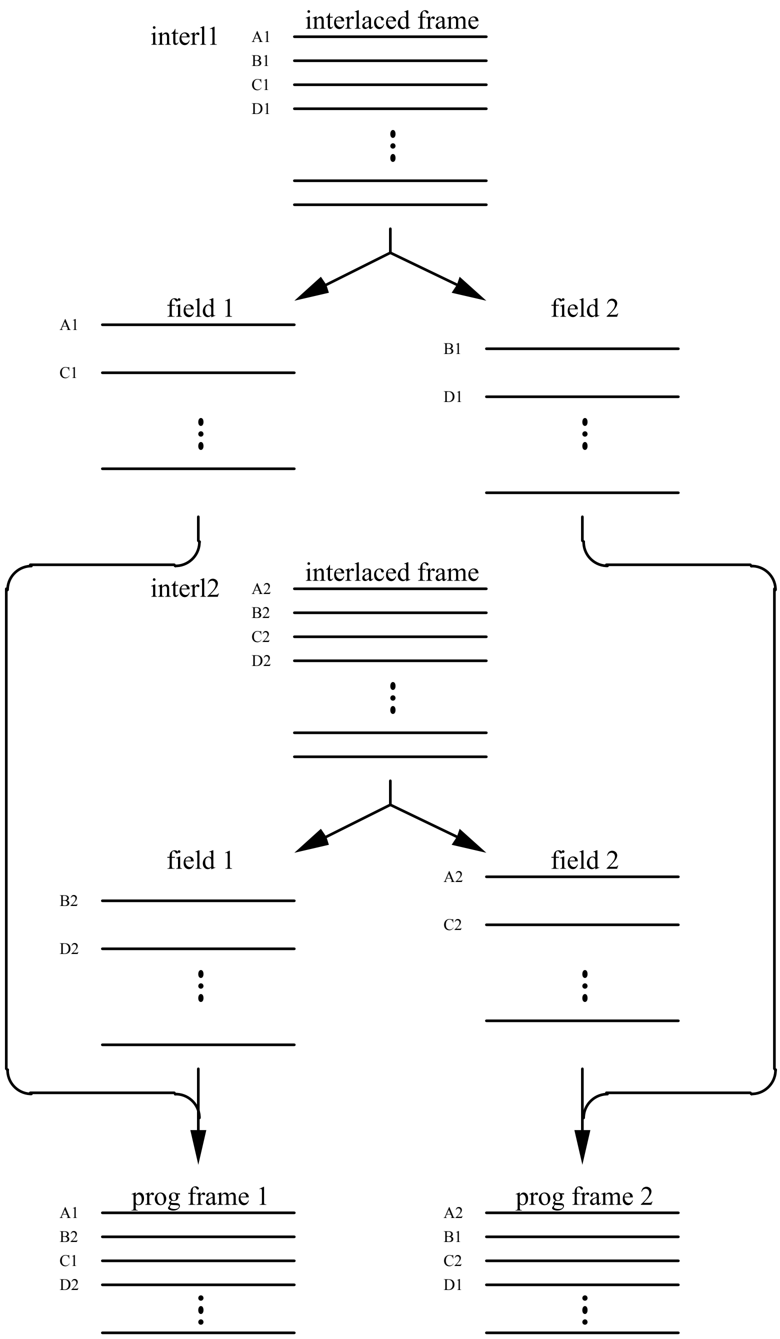


Figure D-4. Two interlace to progressive conversion.

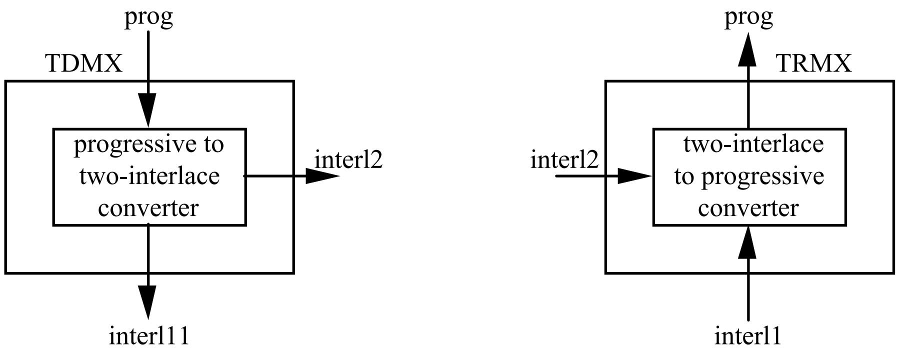


Figure D-5. Temporal demultiplexer and remultiplexer for progressive: interlace-to-interlace temporal scalability

D.7.4.3 Interlace: interlace-to-interlace Temporal Scalability

Assuming interlaced video input, if it is necessary to code interlaced- format video in base and enhancement layers, the operation of *temporal demux* may be relatively simple and involve temporal demultiplexing of input frames into two interlaced sequences; The operation of *temporal remux* is inverse, i.e., it performs remultiplexing of two interlaced sequences to generate full temporal rate interlaced output. The demultiplexing and remultiplexing is similar to that in Figure D-2.

D.7.5 Hybrids of the spatial, the SNR and the temporal scalable extensions

This standard also allows combinations of scalability tools to produce more than 2 video layers as may be useful and practical to support more demanding applications. Taken two at a time, 3 explicit combinations result. Moreover, within each combination, the order in which each scalability is applied, when interchanged, results in distinct applications. In the hybrid scalabilities involving three layers, the layers are referred to as base layer, enhancement layer 1 and enhancement layer 2.

D.7.5.1 Spatial and SNR hybrid scalability applications

A) HDTV with standard TV at two qualities:

Base layer provides standard TV resolution at basic quality, enhancement layer 1 helps generate standard TV resolution but at higher quality by SNR scalability and the enhancement layer 2 employs HDTV resolution and format which is coded with spatial scalability with respect to high quality standard TV resolution generated by using enhancement layer 1.

B) Standard TV at two qualities and low definition TV/videophone:

Base layer provides videophone/low definition quality, using spatial scalability enhancement layer 1 provides standard TV resolution at a basic quality and enhancement layer 2 uses SNR scalability to help generate high quality standard TV.

C) HDTV at two qualities and standard TV:

Base layer provides standard TV resolution. Using spatial scalability enhancement layer 1 provides basic quality HDTV and enhancement layer 2 uses SNR scalability to help generate high quality HDTV.

D.7.5.2 Spatial and temporal hybrid scalability applications

A) High temporal resolution progressive HDTV with basic interlaced HDTV and standard TV:

Base layer provides standard TV resolution, using spatial scalability enhancement layer 1 provides basic HDTV of interlaced format and enhancement layer 2 uses temporal scalability to help generate full temporal resolution progressive HDTV.

B) High resolution progressive HDTV with enhanced progressive HDTV and basic progressive HDTV:

Base layer provides basic progressive HDTV format at temporal resolution, using temporal scalability enhancement layer 1 helps generate progressive HDTV at full temporal resolution and enhancement layer 2 uses spatial scalability to provide high spatial resolution progressive HDTV (at full temporal resolution).

C) High resolution progressive HDTV with enhanced progressive HDTV and basic interlaced HDTV:

Base layer provides basic interlaced HDTV format, using temporal scalability enhancement layer 1 helps generate progressive HDTV at full temporal resolution and enhancement layer 2 uses spatial scalability to provide high spatial resolution progressive HDTV (at full temporal resolution).

D.7.5.3 Temporal and SNR hybrid scalability applications

A) Enhanced progressive HDTV with basic progressive HDTV at two qualities:

Base layer provides basic progressive HDTV at lower temporal rate, using temporal scalability enhancement layer 1 helps generate progressive HDTV at full temporal rate but with basic quality and enhancement layer 2 uses SNR scalability to help generate progressive HDTV with high quality (at full temporal resolution).

B) Enhanced progressive HDTV with basic interlaced HDTV at two qualities:

Base layer provides interlaced HDTV of basic quality, using SNR scalability enhancement layer 1 helps generate interlaced HDTV at high quality and enhancement layer 2 uses temporal scalability to help generate progressive HDTV at full temporal resolution (at high quality).

D.8 Compatibility

The standard supports compatibility between different resolution formats as well as compatibility with ISO/IEC 11172-2 (and ITU-T Rec. H.261).

D.8.1 Compatibility with higher and lower resolution formats

This specification supports compatibility between different resolution video formats. Compatibility is provided for spatial and temporal resolutions with the Spatial Scalability and Temporal Scalability tools. The video is encoded into two resolution layers. A decoder only capable or willing to display a lower resolution video accepts and decodes the lower layer bitstream. The full resolution video can be reconstructed by accepting and decoding both resolution layers provided.

D.8.2 Compatibility with ISO/IEC 11172-2 (and ITU-T Rec. H.261)

The syntax of this specification supports both backward and forward compatibility with ISO/IEC 11172-2. Forward compatibility with ISO/IEC 11172-2 is provided since the syntax of this specification is a superset of the ISO/IEC 11172-2 syntax. An MPEG-2 bitstream that does not contain a *sequence_extension* is backward compatible. The Spatial Scalability tool provided by this specification allows using ISO/IEC 11172-2 coding in the lower resolution, i.e. base layer, thus achieving backward compatibility.

The video syntax contains tools that are needed to implement H.261 compatibility that may be needed for possible future use, however, this is currently not supported by any profile.

Simulcast serves as a simple alternative method to provide backward compatibility with both H.261 and ISO/IEC 11172-2.

D.9 Differences bewteen this specification and ISO/IEC 11172-2

This clause lists the differences between MPEG-1 Video and MPEG-2 Video.

All MPEG-2 Video decoders that comply with currently defined profiles and levels are required to decode MPEG-1 contraints bitstreams.

In most instances, MPEG-2 represents a super-set of MPEG-1. For example, the MPEG-1 coefficient zigzag scanning is one of the two coefficient scanning modes of MPEG-2. However, in some cases, there are syntax elements (or semantics) of MPEG-1 that does not have a direct equivalent in MPEG-2. This document lists all those elements.

This document may help implementers identify those elements of the MPEG-1 video syntax (or semantics) that do not have their direct equivalent in MPEG-2, and therefore require a special care in order to have garantee MPEG-1 compatibility.

In this clause, MPEG-1 refers to ISO/IEC 11172-2 whilst MPEG-2 refers to this specification.

D.9.1 IDCT mismatch

MPEG-1 - The IDCT mismatch control consists in adding (or removing) oneto each non-zero coefficient that would have been even after inversequantization. This is described as part of the inverse quantizationprocess, in sections 2.4.4.1, 2.4.4.2 and 2.4.4.3 of MPEG-1.

MPEG-2 - The IDCT mismatch control consists in adding (or removing) oneto coefficient [7][7] if the sum of all coefficients is even afterinverse quantization. This is described in section 7.4.4 of MPEG-2.

D.9.1 Macroblock stuffing

MPEG-1 - The VLC code "0000 0001 111" (macroblock_stuffing) can beinserted any number of times before each macroblock_address_increment. This code must be discarded by the decoder. This is described insection 2.4.2.7 of MPEG-1.

MPEG-2 - This VLC code is reserved and not used in MPEG-2. In MPEG-2,stuffing can be generated only by inserting zero bytes before astart-code. This is described in section 5.2.3 of MPEG-2.

D.9.1 Run-level escape syntax

MPEG-1 - Run-level values that cannot be coded with a VLC are coded bythe espace code "0000 01" followed by either a 14-bit FLC (-127 <=level <= 127), or a 22-bit FLC (-255 <= level <= 255). This is described in Annex B, section 2-B5 of MPEG-1.

MPEG-2 - Run-level values that cannot be coded with a VLC are coded by the espace code "0000 01" followed by either a 18-bit FLC (-2047 <= level <= 2047). This is described in section 7.2.2.3 of MPEG-2.

D.9.1 Chrominance samples horizontal position

MPEG-1 - The horizontal position of chrominance samples is half the way between luma samples. This is described in section 2.4.1 of MPEG-1.

MPEG-2 - The horizontal position of chrominance samples is co-located with luma samples. This is described in section 6.1.2.1 of MPEG-2.

D.9.1 Slices

MPEG-1 - Slices do not have to start and end on the same horizontal row of macroblocks. Consequently is is possible to have all the macroblocks of a picture in a single slice. This is described in section 2.4.1 of MPEG-1.

MPEG-2 - Slices always start and end on the same horizontal row of macroblocks. This is described in section 6.1.3 of MPEG-2.

D.9.1 D-Pictures

MPEG-1 - A special syntax is defined for D-pictures (picture_coding_type = 4). D-pictures are like I-pictures with only Intra-DC coefficients, no end_of_block, and a special end_of_macroblock code "1".

MPEG-2 - D-pictures (picture_coding_type = 4) are not permitted. This is described in section 6.3.10 of MPEG-2.

D.9.1 Full-pel motion vectors

MPEG-1 - The syntax elements full_pel_forward_vector and full_pel_backward_vector can be set to "1". When this is the case, the motion vectors that are coded are in full-pel units instead of half-pel units. Motion vector coordinates must be multiplied by two before being used for the prediction. This is described in sections 2.4.4.2 and 2.4.4.3 of MPEG-1.

MPEG-2 - The syntax elements full_pel_forward_vector and full_pel_backward_vector must be equal to "0". Motion vectors are always coded in half-pel units.

D.9.1 Aspect ratio information

MPEG-1 - The 4-bit pel_aspect_ratio value coded in the sequence header specifies the pel aspect ratio. This is described in section 2.4.3.2 of MPEG-1.

MPEG-2 - The 4-bit aspect_ratio_information value coded in the sequence header specifies the display aspect ratio. The pel aspect ratio is derived from this and from the frame size and display size. This is described in section 6.3.3 of MPEG-2.

D.9.1 forward_f_code and backward_f_code

MPEG-1 - The f_code values used for decoding the motion vectors are forward_f_code and backward_f_code, located in the picture_header().

MPEG-2 - The f_code values used for decoding the motion vectors are f_code[s][t], located in the picture_coding_extension(). The values of forward_f_code and backward_f_code must be "111" and are ignored. This is described in section 6.3.10 of MPEG-2.

D.9.1 Contrained_parameter_flag and maximum horizontal_size

MPEG-1 - When the contrained_parameter_flag is set to "1", this indicates that a certain number of contraints are verified. One of those constraints is that horizontal_size <= 768. It should be noted that a contrained MPEG-1 video bitstream can have pictures with an horizontal size of up to 768 pels. This is described in section 2.4.3.2 of MPEG-1.

MPEG-2 - The contrained_parameter_flag mechanism has been replaced by the profile and level mechanism. However, it should be noted that MP@ML bitstreams cannot have horizontal size larger than 720 pels. This is described in section 8.2.3.1 of MPEG-2.

D.9.1 MPEG-2 syntax vs. MPEG-1 syntax

It is possible to make MPEG-2 bitstreams that have a syntax very close to MPEG-1, by using particular values for the various MPEG-2 syntax elements that do not exist in the MPEG-1 syntax.

In other words, the MPEG-1 decoding process is the same (except for the particular points mentionned earlier) as the MPEG-2 decoding process when :

progressive_sequence = "1" (progressive sequence).

chroma_format = "01" (4:2:0)

frame_rate_extension_n = 0 and frame_rate_extension_d = 0 (MPEG-1 frame-rate)

intra_dc_precision = "00" (8-bit Intra-DC precision)

picture_structure = "11" (frame-picture, because progressive_sequence = "1")

frame_pred_frame_dct = 1 (only frame-based prediction and frame DCT)

concealment_motion_vectors = "0" (no concealment motion vectors).

q_scale_type = "0" (linear mquant)

intra_vlc_format = "0" (MPEG-1 VLC table for Intra MBs).

alternate_scan = "0" (MPEG-1 zigzag scan)

repeat_first_field = "0" (because progressive_sequence = "1")

chroma_420_type = "1" (chrominance is "frame-based", because progressive_sequence = "1")

progressive_frame = "1" (because progressive_sequence = "1")

D.10 Complexity

The MPEG-2 standard supports combinations of high performance/high complexity and low performance/low complexity decoders. This is accommodated by MPEG-2 with the Profiles and Levels definitions which introduce new sets of tool and functionality with every new profile. It is thus possible to trade-off performance of the MPEG-2 coding schemes by decreasing implementation complexity.

Moreover, certain restrictions could allow reducing decoder implementation cost.

D.11 Editing Encoded Bitstreams

Many operations on the encoded bitstream are supported to avoid the expense and quality costs of re-coding. Editing, and concatenation of encoded bitstreams with no recoding and no disruption of the decoded image sequence is possible.

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.

Editing of encoded MPEG-2 bitstreams is supported due to the syntactic hierarchy of the encoded video bitstream. Unique start codes are encoded with different level in the hierarchy (i.e. video sequence, group of pictures etc.). Video can be encoded with Intra-picture/intra-slices access points in the bitstream. This enables the identification, access and editing of parts of the bitstream without the necessity to decode the entire video.

D.12 Trick modes

Certain DSM (Digital Storage Media) provide the capability of trick modes, such as FF/FR (Fast Forward/Fast Reverse). The MPEG-2 syntax supports all special access, search and scan modes of ISO/IEC 11172-2. This functionality is supported with the syntactic hierarchy of the video bitstream which enables the identification of relevant parts within a video sequence. It can be assisted by MPEG-2 tools which provide bitstream scalability to limit the access bitrate (i.e. Data Partitioning and the general slice structure). This clause provides some guideline for decoding a bitstream provided by a DSM.

The decoder is informed by means of a 1-bit flag (DSM_trick_mode_flag) in the PES packet header. This flag indicates that the bitstream is reconstructed by DSM in trick mode, and the bitstream is valid from syntax point of view, but invalid from semantics point of view. When this bit is set, an 8-bit field (DSM_trick_modes) follows. The semantics of DSM_trick_modes are in the ISO/IEC 13818-1.

D.12.1 Decoder

While the decoder is decoding PES Packet whose DSM_trick_mode_flag is set to 1, the decoder is recommended to:

Decode bitstream and display according to DSM_trick_modes

Pre-processing

When the decoder encounters PES Packet whose DSM_trick_mode_flag is set to 1, the decoder is recommended to:

Clear non trick mode bitstream from buffer

Post-processing

When the decoder encounters PES Packet whose DSM_trick_mode_flag is set to 0, the decoder is recommended to:

Clear trick mode bitstream from buffer

Video Part

While the decoder is decoding PES Packet whose DSM_trick_mode_flag is set to 1, the decoder is recommended to:

Neglect vbv_delay and temporal_reference value

Decode one picture and display it until next picture is decoded.

The bitstream in trick mode may have a gap between slices. When the decoder encounters a gap between slices, the decoder is recommended to:

Decode the slice and display it according to the slice vertical position in slice header

Fill up the gap with co-sited part of the last displayed picture

D.12.2 Encoder

The encoder is recommended to:

Encode with short size of slice with intra macroblocks.

Encode with short periodic refreshment by intra picture or intra slice.

DSM

DSM is recommended to provide the bitstream in trick mode with perfect syntax.

Pre-processing

DSM is recommended to:

Complete "normal" bitstream at picture_header() and higher syntactic structures.

System Part

DSM is recommended to:

Set DSM_trick_mode_flag to 1 in a PES Packet header.

Set DSM_trick_modes(8-bit) according to the trick mode.

Video Part

DSM is recommended to:

Insert sequence_header() with the same parameter as normal bitstream.

.

Insert Sequence extension header with the same parameter as normal bitstream

Insert Picture header with the same parameter as normal bitstream except vbv_delay. Set vbv_delay = FFFF, to indicate variable bitrate.

Note temporal_reference and vbv_delay are ignored in the decoder, therefore DSM needs not to set temporal_reference and vbv_delay to correct values.

Concatenate slices which consists of intra coded macroblocks. The concatenated slices should have slice vertical positions in increasing order.

D.13 Error Resilience

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 specification. Nevertheless the MPEG-2 syntax supports error resilient modes relevant to cell loss in ATM networks and bit errors (isolated and in bursts) in transmissions. The slice structure of the compression scheme defined in this specification allows a decoder to recover after a residual data error and to resynchronise 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. Error resilience includes graceful degradation in proportion to bit error rate (BER) and graceful recovery in the face of missing video bits or data packets. It has to be noted that all items may require additional support at the system level.

Being an example of a packet-based system, B-ISDN with its Asynchronous Transfer Mode (ATM) is addressed in some detail in the following. Similar statements can be made for other systems where certain packets of data are protected individually by means of forward error-correcting coding.

ATM uses short, fixed length packets, called cells, consisting of a 5 byte header containing routing information, and a user payload of 48 bytes. The nature of errors on ATM is such that some cells may be lost, and the user payload of some cells may contain bit errors. Depending on AAL (ATM Adaptation layer) functionality, indications of lost cells and cells containing bit errors may be available.

As an indication of the impact of cell loss in an ATM environment Table D-2 summarises the average interval between cell losses for a range of CLR and service bitrates based on simple statistical modelling. (A cell payload must be assumed for this. Allowing 1 byte/cell for AAL functions leaves 376 bits = 47 bytes). Note, however, that this summary ignores cell loss bursts and other shorter term temporal statistics.

Table D-2. Average interval between cell losses for a range of CLR and service bitrates.

	Average interval time of error			
	5 Mb/s	**10 Mb/s**	**50 Mb/s**	**100 Mb/s**
10^{-2}	7,52 ms	3,76 ms	0,752 ms	0,376 ms
10^{-3}	75,2 ms	37,6 ms	7,52 ms	3,76 ms
10^{-4}	752 ms	376 ms	75,2 ms	37,6 ms
10^{-5}	7,52 s	3,76 s	752 ms	376 ms
10^{-6}	1,25 m	37,6 s	7,52 s	3,76 s
10^{-7}	12,5 m	6,27 m	1,25 m	37,6 s
10^{-8}	2,09 h	1,04 h	12,5 m	6,27 m

Bit Error Ratios (BERs) corresponding to the above mean times between errors can be calculated easily for the case of isolated bit errors. The BER that would cause the same incidence rate of errors is found by dividing by the cell payload size. i.e. BER = CLR/376.

The following techniques of minimising the impact of lost cells and other error/loss effects are provided for reference, and indicate example methods of using the various tools available in this specification to provide good performance in the presence of those errors. Note that the techniques described may be applicable in the cases of packets of other sizes (e.g. LANs or certain storage media) or video data with uncorrected errors of different characteristics, in addition to cell loss. It may be appropriate to treat a known erasure (uncorrected bit error(s) known to exist somewhere in a data block) as a lost data block, since the impact of bit errors cannot be predicted. However, this should be a decoder option. The discussion that follows refers generally to "transport packets" where appropriate, to emphasise the applicability to a variety of transport and storage systems. However, specific examples will refer to Cell Loss Ratios (CLRs) because cell transport is the most completely defined at the time of preparing this specification.

The error resilience techniques are summarised in three categories, covering methods of concealing the error once it has occurred, and the restriction of the influence of a loss or error in both space (within a picture) and time (from picture to picture).

D.13.1 Concealment possibilities

Concealment techniques hide the effect of losses/errors once they have occurred. Some concealment methods can be implemented using any encoded bitstream, while others are reliant on the encoder to structure the data or provide additional information to enable enhanced performance.

D.13.1.1 Temporal predictive concealment

A decoder can provide concealment of the errors by estimating the lost data from spatio-temporally adjacent data. The decoder uses information which has been successfully received to make an informed estimate of what should be displayed in place of the lost/errored data, under the assumption that the picture characteristics are fairly similar across adjacent blocks (in both the spatial and temporal dimensions). In the temporal case, this means estimation of errored or lost data from nearby fields or frames.

D.13.1.1.1 Substitution from previous frame

The simplest possible approach is to replace a lost macroblock with the macroblock in the same location in the previous picture. This approach is suitable for relatively static picture areas but block displacement is noticeable for moving areas.

The "previous picture" must be interpreted with care due to the use of bi-directional prediction and a difference between picture decoding order and picture display order. When a macroblock is lost in a P- or I-picture, it can be concealed by copying the data corresponding to the same macroblock in the previous P-picture or I-picture. This ensures that the picture is complete before it is used for further prediction. Lost macroblocks in B-pictures can be substituted from the last displayed picture, of any type, or from a future I- or P-picture held in memory but not yet displayed.

D.13.1.1.2 Motion compensated concealment

The concealment from neighbouring pictures can be improved by estimating the motion vectors for the lost macroblock, based on the motion vectors of neighbouring macroblocks in the affected picture (provided these are not also lost). This improves the concealment in moving picture areas, but there is an obvious problem with errors in macroblocks whose neighbouring macroblocks are coded intra, because there are ordinarily no motion vectors. Encoder assistance to get around this problem is discussed in D.12.1.1.3.

Sophisticated motion vector estimation might involve storage of adjacent macroblock motion vectors from above and below the lost macroblock, for predictions both forward and backward (for B-pictures) in time. The motion vectors from above and below (if available) could then be averaged.

Less complex decoders could use, for example, only forward prediction and/or only the motion vector from the macroblock above the lost macroblock. This would save on storage and interpolation.

D.13.1.1.3 Use of Intra MVs

The motion compensated concealment technique outlined in D.12.1.1.2 could not ordinarily be applied when the macroblocks above and below the lost/errored macroblock are Intra-coded, since there is no motion vector associated with Intra-coded macroblocks. In particular, in I-pictures, this type of concealment would not be possible with the normal calculation and use of motion vectors.

The encoding process can be extended to include motion vectors for intra macroblocks. Of course, the motion vector and coded information for a particular macroblock must be transmitted separately (e.g. in different packets) so that the motion vector is still available in the event that the image data is lost.

When "concealment_motion_vectors" = 1, motion vectors are transmitted with Intra macroblocks, allowing improved concealment performance of the decoders. The concealment motion vector associated with an Intra-coded macroblock is intended to be used only for concealment (if necessary) of the macroblock located immediately below the Intra-coded macroblock.

For simplicity, concealment motion vectors associated with Intra-coded macroblocks are always forward, and are considered as frame motion-vectors in Frame pictures and field motion-vectors in field pictures.

Therefore, encoders that choose to generate concealment motion vectors should transmit, for a given Intra-coded macroblock, the frame- or field-motion vector that should be used to conceal (i.e. to predict, with forward frame- or field-based prediction respectively) the macroblock located immediately below the Intra-coded macroblock.

Concealment motion vectors are intended primarily for I- and P-pictures, but the syntax allows their use in B-pictures. Concealment in B-pictures is not critical, since B-pictures are not used as predictors and so errors do not propagate to other pictures. Therefore, it may be wasteful to transmit concealment motion vectors in B-pictures.

Concealment motion vectors transmitted with Intra macroblocks located in the bottom row of a picture cannot be used for concealment. However, if "concealment_motion_vectors" = 1, those concealment motion vectors must be transmitted. Encoders can use the (0, 0) motion vector to minimise the coding overhead.

When concealment motion vectors are used, it is a good idea to have one slice contain one row of macroblocks (or smaller), so that concealment can be limited to less than one row of macroblocks when a slice, or part of a slice, is lost. This means that the loss of macroblocks in two successive rows is much less likely, and therefore the chances of achieving effective concealment using concealment motion vectors is improved.

Note when "concealment_motion_vectors" = 1, PMVs (Predictors for Motion Vectors) are NOT reset when an Intra macroblock is transmitted. Ordinarily, an Intra macroblock would reset the PMVs.

D.13.1.2 Spatial predictive concealment

The generation of predicted, concealment macroblocks is also possible by interpolation from neighbouring macroblocks within the one picture (Annex G [17]). This is best suited to areas of high motion, where temporal prediction is not successful, or as an alternative means of concealment for Intra macroblocks when concealment motion vectors (D.12.1.1.3) are not available. It also could be particularly useful for cell loss after scene changes.

There are several possible approaches to spatial interpolation, and it could be carried out in the spatial or DCT domain, but normally it is only feasible and useful to predict the broad features of a lost macroblock, such as the DC coefficient and perhaps the lowest AC coefficients. Spatial prediction of fine detail (high frequencies) is likely to be unsuccessful and is of little value in fast-moving pictures anyway.

Spatial predictive macroblock concealment may also be useful in combination with layered coding methods (i.e. Data Partitioning or SNR scalable pyramid, see D.12.1.3). If in the event of cell loss some DCT coefficients in a macroblock are recovered from the lower layer, it is possible to use all information available (DCT coefficients recovered in the same macroblock from the lower layer and all DCT coefficients received in the adjacent macroblocks) for error concealment. This is especially useful if the lower layer only contains DC coefficients due to bandwidth constraints.

D.13.1.3 Layered coding to facilitate concealment

It is possible to assist the concealment process further by arranging the coded video information such that the most important information is most likely to be received. The loss of the less important information can then be more effectively concealed. This approach can gain from use of a transmission medium or storage device with different priority levels (such as priority-controlled cell-based transmission in the B-ISDN, or where different error protection or correction is provided on different channels). The components produced by the coding process can be placed in a hierarchy of importance according to the effect of loss on the reconstructed image. By indicating the priority of bitstream components and treating the individual components with due importance, superior error concealment performance may be possible.

Strategies available for producing hierarchically ordered bitstreams, or layers, include

data partitioning - the coded macroblock data is partitioned into multiple layers such that partition zero contains address and control information and lower order DCT coefficients, while partition one contains high frequency DCT coefficients.

SNR scalable pyramid - two sets of coefficients are dequantised and then added together at the receiver before decoding. One set of coefficients could be a refinement of the quantisation error of the other, but other combinations (including an emulation of data partitioning) are possible.

spatial scalable pyramid - the lower layer may be coded without regard for the enhancement layer, and could use other standard coding methods (ISO/IEC 11172-2 etc.). The enhancement layer contains the coded difference signal from a prediction based on the lower layer.

temporal scalable pyramid - the enhancement layer defines additional pictures which, when remultiplexed with the base layer, provides a combined picture sequence of greater picture rate.

These strategies produce layers which, when added progressively, produce increasing quality of the reconstructed signal. While some of these source coding techniques may result in a bitrate increase compared to the system without layering, the performance of the layered systems, when subjected to channel errors, may be greater.

Considering error resilience alone, the hierarchically ordered layers should be handled with due quality, such that some function (such as picture quality for a given total bitrate) is optimised. The bitstream components may be treated differently at one or more of the following locations:

- encoder - different channel coding might be used
- channel - the channel may be able to provide different cell/packet loss probabilities or error characteristics to the different bitstream components.
- decoder - error concealment could be performed differently within each bitstream

D.13.1.3.1 Use of data partitioning

Data partitioning allows a straightforward division of macroblock data into two layers. The PBP pointer determines the contents of each layer. Ordinarily, data partition 0 contains the address and control information and the low frequency DCT coefficients, while data partition 1 contains the high frequency DCT coefficients.

At the encoder the value of the PBP pointer may be different for each slice such that the distribution of bits between the two layers may be controlled (e.g. maintained constant). The distribution may be different for I, P, and B frames. The management of rate between the layers could mean that, for some macroblocks, data partition 0 contains no DCT coefficients or motion vectors.

Good tolerance to errors can be achieved if channel errors are distributed so that data partition 1 receives most errors.

It is assumed that errors can be detected at the decoder, so that actions can be taken to prevent errored data from being displayed. For data partition 1, errored data is simply not displayed (i.e. only data partition 0 is used). Losses or errors in data partition 0 should be minimised through use of high reliability transport. Decoder concealment actions may also be necessary.

D.13.1.3.2 Use of SNR scalable coding

SNR scalable coding provides two layers with the same spatial resolution but different image quality, depending on whether one or both layers are decoded. This technique is mainly intended to provide a lower-quality layer that is usable even when the enhancement layer is absent. However, it also provides good error resilience if the errors can be mainly confined to the enhancement layer.

In case of errors in the enhancement layer the lower layer signal can be used alone for the affected image area. Especially in the case of frequent errors, temporary loss or permanent unavailability of the enhancement layer this concealment is very effective, since the displayed signal can be made relatively free of non-linear distortions like blocking or motion jerkiness.

If the enhancement layer is permanently unavailable and so only the lower layer is decoded, a small drift may occur in the case where only one MC prediction loop is implemented in the encoder. However, this drift is likely to be invisible in most configurations (e.g. M=3, N=12 would normally provide correction often enough).

The lower-layer signal of an SNR Scalable system is well suited to concealment in the case of a very high error rate, temporary or permanent loss of the upper-layer signal. However, the upper-layer quality in the error-free case does not achieve that of a sub-band like layered scheme (e.g. data partitioning).

D.13.1.3.3 Use of spatial scalable coding

Spatial scalable coding allows the lower layer to be coded without regard for the enhancement layer, and other standard coding methods (ISO/IEC 11172-2 etc.) could be used. The enhancement layer contains the coded difference signal from a prediction based on the lower layer. In case of errors in the enhancement layer the upconverted lower layer signal can be used directly as concealment information for the affected image area. Especially in case of frequent errors or temporary loss of the enhancement layer signal this concealment signal is relatively free of non-linear distortions like blocking (which could arise if high frequency DCT coefficients are completely absent from the lower layer) or motion jerkiness (if the motion information is omitted from the high priority layer).

In the error-free case the upconverted lower layer signal is used as an additional prediction signal in a macroblock-adaptive way to improve the upper-layer coding performance. The enhancement layer bitstream therefore consists of the quantised residual temporal or lower layer prediction errors.

Spatial scalable coding provides a lower layer signal that is very suitable for concealment in case of a high error rate or temporary loss of the enhancement layer signal. However, the quality of the enhanced picture when both layers are available will not, in general, be as good as other layered coding approaches.

D.13.1.3.4 Use of temporal scalable coding

Temporal scalability is a coding technique that allows layering of video frames. The spatial resolution of frames in each layer is the same but the temporal rates of each layer are lower than that of the source; however the combined temporal rate of the two layers results in full temporal rate of the source. In case of errors in the enhancement layer, the base layer of full spatial resolution can be easily used for concealment. Especially in case of frequent errors or temporary loss of the enhancement layer signal, the base layer signal offers good concealment properties.

In telecommunications applications such as those employing the SCIF format high degree of error resilience can be achieved with temporal scalability by encoding the base layer using the SCIF spatial resolution but only half the temporal resolution; the remaining frames corresponding to the other half of the temporal resolution are coded in the enhancement layer. Typically, the enhancement layer data may be assigned lower priority and when lost, the base layer decoded frames can be used for concealment by frame repetition. This type of concealment leads to only a temporary loss of full temporal resolution while maintaining full spatial quality and full spatial resolution.

In HDTV applications such as those using high temporal resolution progressive video format as source, high degree of error resilience can be achieved with temporal scalability. Such an application is envisaged to require 2 layers, a base layer and an enhancement layer, each of which process same picture formats (either both progressive or both interlaced) but at half the temporal rates. Temporal remultiplexing of the base and enhancement layer signals irrespective of their chosen formats always results in full progressive temporal resolution of the source. In HDTV transmission, if the lower priority enhancement layer signal is corrupted, the base layer signal can be used for concealment, either directly, as in case of progressive format base layer or after reversal of parity of fields for interlaced format base layer.

Typically, the enhancement layer data may be assigned lower priority and when lost, the base layer decoded frames can be used for concealment by either frame repetition or frame averaging. This type of concealment leads to only a temporary indistinguishable loss in temporal resolution while maintaining full spatial quality and full spatial resolution.

D.13.2 Spatial localisation

Spatial localisation encompasses those methods aimed at minimising the extent to which errors propagate within a picture, by providing early resynchronisation of the elements in the bitstream that are coded differentially between macroblocks.

Isolated bit errors may be detected through invalid codewords and so a decoder designer may choose to allow an errored sequence to be decoded. However, the effect on the picture is difficult to predict (legal, but incorrect, codewords could be generated) and it may be preferable to control the error through concealment of the entire affected slice(s) even when only one bit is known to be in error somewhere in a block of data.

When long consecutive errors occur (e.g. packet or cell loss), virtually the only option is to discard data until the next resynchronisation point is located (a start code at the next slice or picture header). By providing more resynchronisation points, the area of the screen affected by a loss or error can be reduced, in turn reducing the demands on the concealment techniques and making the errors less visible at the expense of coding efficiency. Spatial localisation of errors is therefore dependent on controlling the slice size since this is the smallest coded unit with resynchronisation points (start codes).

D.13.2.1 Small slices

The most basic method for achieving spatial localisation of errors is to reduce the (fixed) number of macroblocks in a slice. The increased frequency of resynchronisation points will reduce the affected picture area in the event of a loss. It is effective in any transport or storage media, and in any profile since the slice structure is always present in MPEG coded video.

The method results in a small loss of coding efficiency due to the increase of overhead information. The loss is about 3% for 11 Macroblocks per slice and 12% for 4 Macroblocks per slice based on Rec. 601 picture format at 4 Mb/s, (percentages calculated relative to a system using 44 Macroblocks, or one picture width, per slice). The efficiency loss results in degradation of picture quality up to about 1 dB with 4 Macroblocks per slice and 0,2 dB with 11 Macroblocks per slice without errors at 4 Mb/s. However, the method performs approximately 1 to 5 dB better at CLR = 10^{-2}, depending on the concealment method used (simple macroblock replacement or motion compensated concealment).

From the view point of perceived picture quality, the performance of this method is generally dependent on the relative size of slice size and picture. Therefore, the slice size should be decided by considering the picture size (in macroblocks) and the trade-off between coding efficiency and visual degradation due to errors.

D.13.2.2 Adaptive slice size

There is a significant variation in the number of bits required to code a picture slice, depending on the coding mode, picture activity, etc. If slices contain only a few macroblocks, it will be possible that one transport packet, even a short packet or cell, could contain several slices. Offering multiple resynchronisation points in the same transport packet serves no purpose. Another problem with the simplistic short slice approach is that, because no account is taken of the transport packet structure, the first valid transport packet after a loss could contain most of the information for a slice, but it is unusable because the start code was lost.

An improvement over the small slice method may be to use adaptive slice sizes. As the encoder is producing the bitstream, it keeps track of the data contents within transport packets. The start of a slice is placed at the first opportunity in every transport packet (or in every second, third, ...). This approach can achieve about the same spatial localisation of errors as small, fixed size slices, but with a greater efficiency.

However, this method ONLY gives an advantage for cell or packet based transmission, or where error detection occurs over a large block of data. The frequent resynchronisation points of small slice localisation are only wasteful if more than one is lost in the event of an error. If isolated bit errors affect just one slice anyway, then there is no advantage in adapting the slice size.

Furthermore, the adaptive slice size technique requires an intimate connection between encoder and packetiser, to allow a new slice for a new packet or cell. As such, it may not be appropriate for some applications (e.g. stored video intended to be distributed by multiple means) because only one transport packet structure would be assumed during encoding.

D.13.3 Temporal localisation

Temporal localisation encompasses those methods aimed at minimising the extent to which errors propagate from picture to picture in the temporal sequence, by providing early resynchronisation of pictures that are coded differentially. An obvious way to do this is to make use of intra mode coding.

D.13.3.1 Intra pictures

By use of intra pictures a single error will not stay in the decoded picture longer than (N + M -1) pictures if every Nth picture is coded intra and (M-1) B pictures are displayed before each I picture.

While the intra pictures, normally used as "anchors" for synchronising the video decoding part way through a sequence, are useful for temporal localisation, care should be taken in adding extra intra pictures (i.e. reducing N) for error resilience. Intra pictures require a large number of bits to code, take up a relatively large proportion of the encoded bitstream and, as a result, are more likely to be affected by losses or errors themselves.

D.13.3.2 Intra slices

To avoid the additional delay caused by intra pictures, some applications requiring low delay may want to update the picture by coding only parts of the picture intra. This may provide the same kind of error resilience as intra pictures. As an example assume that a constant number of slices per picture from top to bottom are intra coded so that the whole picture is updated every P pictures. Three aspects of this kind of updating should be kept in mind:

- While an errored portion of the scene will ordinarily be erased within P pictures (with an average duration of about P/2), it is possible that motion compensation will allow the disturbance to bypass the intra refresh and it may persist as long as 2P pictures.
- To ensure that errors are not propagating into the updated region of the picture, restrictions could be put on motion vectors, limiting the vertical vector components to ensure that predictions are not made from the "oldest" parts of the picture.
- The visual effect of clearing errors can be similar to a windscreen wiper clearing water. This *windscreen wiper* effect can become noticeable in some cases in the error free sequence, unless the rate control mechanism ensures that the quality of the intra slice is close to that of the surrounding non-intra macroblocks.

D.13.4 Summary

Table D-3 summarises the above error resilience techniques, with a guide to their applicability.

Table D-3. Summary of error concealment techniques.

Category	Technique	Profile/Applicability
Concealment	Temporal predictive - substitution from previous picture	Any profile. Most suited to static pictures.
	Temporal predictive - Motion compensated	Any profile. Choice of sophistication in motion vector estimation.
	Temporal predictive - using concealment MVs	Any profile, but calculation of Intra MVs is an encoder option.
	Spatial predictive	Any profile. Not suitable for static, complex pictures.
	Data Partitioning	Not currently used in a profile, but may be added as post/pre-processing. Minimal overhead and complexity. Depending on bitrate allocation, lower layer may not provide usable pictures by itself.
	SNR Scalability	SNR SCALABLE, SPATIALLY SCALABLE, HIGH profiles. Suitable for very high error rates or temporary unavailability of the enhancement layer. Relatively simple to implement.
	Spatial Scalability	SPATIALLY SCALABLE and HIGH profiles. Suitable for very high error rates or temporary unavailability of the enhancement layer.
	Temporal Scalability	Not currently used in a profile. Suitable for very high error rates or temporary unavailability of the enhancement layer.
Spatial Localisation	Small Slices	Any profile
	Adaptive slice sizes	Any profile, but requires knowledge of transmission characteristics when packet size is decided.
Temporal Localisation	Intra pictures	Any profile, but has delay implications.
	Intra slices	Any profile, but errors may persist longer than for Intra picture method.

It is not possible to provide a concise indication of error resilience performance, because assessments must necessarily be subjective and application dependent, and so should be taken as nothing more than a guide. It is also true that several different approaches to error resilience are likely to be used in combination. However, the following descriptions are provided as some guidance to performance. They are the results of cell loss experiments, looking only at cell-based transmission of video information.

A simple macroblock substitution from a previous frame combined with the small-slice method (4 macroblocks per slice) will provide adequate picture quality for most sequences in the presence of rather low error rates of around CLR = 10^{-5} (in a reference 4 Mbit/s, Main Profile, Main Level system).

Including sophisticated motion compensated concealment (with full spatial and temporal interpolation of motion vectors for lost macroblocks, and concealing losses in P pictures that use intra slice updating, i.e. N= infinity, M=1) provides adequate picture quality at CLR = 10^{-3} (again, in a reference 4 Mbit/s, Main Profile, Main Level system).

Operation in environments with greater loss may require use of one of the layered coding methods. With adequate protection of the high priority information, these schemes can provide adequate performance in the face of CLRs as high as 10^{-2} or even 10^{-1}. Data partitioning, implemented as a post-processing function to a 4 Mbit/s Main Profile, Main Level system, with 50% of the rate allocated to each partition and no loss in the base layer, has been shown in one example to give approximately 0,5 dB loss in SNR at a CLR of 10^{-3}, about 1,5 dB loss at 10^{-2}, and with almost no visible degradation in either case.

Given the range of different layered coding approaches that are possible, some general comments may be useful. In general, it is not expected that inclusion of the most complex layered coding methods could be justified purely on the basis of error resilience. Instead, they could be utilised for error resilience if they were required to satisfy other system requirements. Data partitioning is very simple to implement and is likely to provide error resilience very nearly the same as any of the other methods except in the case of extremely high error rates (>10% loss) or where the enhancement signal could be lost completely. SNR scalability is slightly more complex, and has slightly lower efficiency than data partitioning, but it is easier to produce lower layer signals of a usable quality when the enhancement layer is absent. Spatial scalability is more complex again, but provides a good lower layer picture quality at the expense of overall (two layer) efficiency.

Annex E

Profile and level restrictions

(This annex does not form an integral part of this Recommendation | International Standard)

E.1 Syntax element restrictions in profiles

This Clause tabulates all of the syntactic elements defined in this specification. Each is classified to indicate whether it is required to be supported by a decoder compliant to a particular profile and level. Note that normative specifications for compliance are given in ISO/IEC 13818-4.

Note This Clause is informative and is simply intended as a summary of the normative restrictions set out in Clause 8. If, because of an error in the preparation of this text, a discrepancy exists between Clause 8 and Annex E the normative text in Clause 8 shall always take precedence.

In the tables that follow a number of abbreviations are used as shown in Table E-1.

Table E-1. Abbreviations used in the Tables of Clause E

Abbreviation	Used in	Meaning
x	Status	must be supported by the decoder
o	Status	need not be supported by the decoder
D	Type	item with Level-dependent parameters
I	Type	item independent of the Level in the Profile
P	Type	item for post-processing after decoding; the decoder must be capable of decoding bitstreams which contain these items, but their use is beyond the scope of this Recommendation | International Standard.

Note - "Status" is kept blank if an entry is not a syntactic element.

Table E-2. Sequence header

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**horizontal_size_value**	x	x	x	x	x	D	see Table 8-7
02	**vertical_size_value**	x	x	x	x	x	D	see Table 8-7
03	**aspect_ratio_information**	x	x	x	x	x	P	
04	**frame_rate_code**	x	x	x	x	x	D	see Tables 8-7 and 8-6
05	(pel rate) Note: this is not a syntactic element						D	see Table 8-8; pel rate is a product of pels/line, lines/frame and frames/sec
06	**bit_rate_value**	x	x	x	x	x	D	see Table 8-9
07	**vbv_buffer_size_value**	x	x	x	x	x	D	see Table 8-10
08	**constrained_parameters_flag**	x	x	x	x	x	I	set to '1' if MPEG-1 constrained, set to '0' if MPEG-2
09	**load_intra_quantiser_matrix**	x	x	x	x	x	I	
10	**intra_quantiser_matrix[64]**	x	x	x	x	x	I	
11	**load_non_intra_quantiser_matrix**	x	x	x	x	x	I	
12	**non_intra_quantiser_matrix[64]**	x	x	x	x	x	I	
13	sequence_extension()	x	x	x	x	x	I	always present if MPEG-2
14	sequence_display_extension()	x	x	x	x	x	P	
15	sequence_scalable_extension()	o	o	x	x	x	I	see Table 8-11 for maximum number of scalable layers
16	user_data()	x	x	x	x	x	I	decoder may skip this data

1 **Table E-3. Sequence extension**

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**profile_and_level_indication**	x	x	x	x	x	D	profile: one of 8 values level: one of 16 values escape bit: one of 2 values
02	**progressive_sequence**	x	x	x	x	x	I	
03	**chroma_format**	x	x	x	x	x	I	see Table 8-5
04	**horizontal_size_extension**	x	x	x	x	x	D	input picture size related
05	**vertical_size_extension**	x	x	x	x	x	D	input picture size related
06	**bit_rate_extension**	x	x	x	x	x	D	input picture size related
07	**vbv_buffer_size_extension**	x	x	x	x	x	D	input picture size related
08	**low_delay**	x	x	x	x	x	I	
09	**frame_rate_extension_n**	x	x	x	x	x	I	set to 0 for all defined profiles
10	**frame_rate_extension_d**	x	x	x	x	x	I	set to 0 for all defined profiles

2

3 **Table E-4. Sequence display extension elements**

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**video_format**	x	x	x	x	x	P	
02	**colour_description**	x	x	x	x	x	P	input format related
03	**colour_primaries**	x	x	x	x	x	P	
04	**transfer_characteristics**	x	x	x	x	x	P	
05	**matrix_coefficients**	x	x	x	x	x	P	
06	**display_horizontal_size**	x	x	x	x	x	P	input format related
07	**display_vertical_size**	x	x	x	x	x	P	input format related

4

1 **Table E-5. Sequence scalable extension**

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
		Status						
01	**scalable_mode**	o	o	x	x	x	I	SNR Profile: SNR Scalability Spatial and High Profile: SNR or Spatial Scalability
02	**layer_id**	o	o	x	x	x	I	
	if(spatial scalable)							
03	**lower_layer_prediction_ horizontal_size**	o	o	o	x	x	D	see table 8-8 for luminance sampling density
04	**lower_layer_prediction_ vertical_size**	o	o	o	x	x	D	see table 8-8 for luminance sampling density
05	**horizontal_subsampling_ factor_m**	o	o	o	x	x	I	
06	**horizontal_subsampling_ factor_n**	o	o	o	x	x	I	
07	**vertical_subsampling_ factor_m**	o	o	o	x	x	I	
08	**vertical_subsampling_ factor_n**	o	o	o	x	x	I	
	if(temporal scalable)							
09	**picture_mux_enable**	o	o	o	o	o	I	
10	**mux_to_progressive_sequence**	o	o	o	o	o	I	
11	**picture_mux_order**	o	o	o	o	o	I	
12	**picture_mux_factor**	o	o	o	o	o	I	

2

Table E-6. Group of pictures header

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**time_code**	x	x	x	x	x	I	
02	**closed_gop**	x	x	x	x	x	I	
03	**broken_link**	x	x	x	x	x	I	

Table E-7. Picture header

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**temporal_reference**	x	x	x	x	x	I	
02	**picture_coding_type**	x	x	x	x	x	I	Simple Profile: I,P Main, SNR, Spatial & High Profile: I, P, B
03	**vbv_delay**	x	x	x	x	x	I	
04	**full_pel_forward_vector**	x	x	x	x	x	I	"0" for MPEG-2
05	**forward_f_code**	x	x	x	x	x	I	"111" for MPEG-2
06	**full_pel_backward_vector**	x	x	x	x	x	I	"0" for MPEG-2
07	**backward_f_code**	x	x	x	x	x	I	"111" for MPEG-2
08	**extra_information_picture**	x	x	x	x	x	I	
09	picture_coding_extension()	x	x	x	x	x	I	
10	quant_matrix_extension()	x	x	x	x	x	I	
11	picture_display_extension()	x	x	x	x	x	P	
12	picture_spatial_scalable_extension()	o	o	o	x	x	I	
13	picture_temporal_scalable_extension()	o	o	o	o	o	I	

1 **Table E-8. Picture coding extension**

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**f_code[0][0] (forward horizontal)**	x	x	x	x	x	D	Low Level [1:7] Main Level [1:8] High-1440 & High Level [1:9]
02	**f_code[0][1] (forward vertical)**	x	x	x	x	x	D	Low Level [1:4] Main, High-1440 & High Level [1:5]
03	**f_code[1][0] (backward horizontal)**	x	x	x	x	x	D	Low Level [1:7] Main Level [1:8] High-1440 & High Level [1:9]
04	**f_code[1][1] (backward vertical)**	x	x	x	x	x	D	Low level [1:4] Main, H-14 & High Level [1:5]
05	**intra_dc_precision**	x	x	x	x	x	I	Simple, Main, SNR & Spatial Profile: [8:10] High Profile: [8:11]
06	**picture_structure**	x	x	x	x	x	I	
07	**top_field_first**	x	x	x	x	x	I	
08	**frame_pred_frame_dct**	x	x	x	x	x	I	
09	**concealment_motion_vectors**	x	x	x	x	x	I	
10	**q_scale_type**	x	x	x	x	x	I	
11	**intra_vlc_format**	x	x	x	x	x	I	
12	**alternate_scan**	x	x	x	x	x	I	
13	**repeat_first_field**	x	x	x	x	x	I	
14	**chroma_420_type**	x	x	x	x	x	P	
15	**progressive_frame**	x	x	x	x	x	P	
16	**composite_display_flag**	x	x	x	x	x	P	
17	**v_axis**	x	x	x	x	x	P	
18	**field_sequence**	x	x	x	x	x	P	
19	**sub_carrier**	x	x	x	x	x	P	
20	**burst_amplitude**	x	x	x	x	x	P	
21	**sub_carrier_phase**	x	x	x	x	x	P	

Table E-9. Quant matrix extension

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**load_intra_quantiser_matrix**	x	x	x	x	x	I	
02	**intra_quantiser_matrix[64]**	x	x	x	x	x	I	
03	**load_non_intra_quantiser_matrix**	x	x	x	x	x	I	
04	**non_intra_quantiser_ matrix[64]**	x	x	x	x	x	I	
05	**load_chroma_intra_quantiser_ matrix**	o	o	o	o	x	I	
06	**chroma_intra_quantiser_ matrix[64]**	o	o	o	o	x	I	
07	**load_chroma_non_intra_ quantiser_matrix**	o	o	o	o	x	I	
08	**chroma_non_intra_quantiser_ matrix[64]**	o	o	o	o	x	I	

Table E-10. Picture display extension.

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**frame_centre_horizontal_offset**	x	x	x	x	x	P	input format related
02	**frame_centre_vertical_offset**	x	x	x	x	x	P	input format related

Table E-11. Picture temporal scalable extension

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**reference_select_code**	o	o	o	o	o	I	
02	**forward_temporal_reference**	o	o	o	o	o	I	
03	**backward_temporal_reference**	o	o	o	o	o	I	

Table E-12. Picture spatial scalable extension

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**lower_layer_temporal_reference**	o	o	o	x	x	I	
02	**lower_layer_horizontal_offset**	o	o	o	x	x	D	input format related
03	**lower_layer_vertical_offset**	o	o	o	x	x	D	input format related
04	**spatial_temporal_weight_code_ table_index**	o	o	o	x	x	I	
05	**lower_layer_progressive_frame**	o	o	o	x	x	I	
06	**lower_layer_deinterlaced_field_ select**	o	o	o	x	x	I	

Table E-13. Slice layer

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**slice_vertical_position_ extension**	x	x	x	x	x	D	input format related
02	**priority_breakpoint**	o	o	o	o	o	I	only required for data partitioning
03	**quantiser_scale_code**	x	x	x	x	x	I	
04	**intra_slice**	x	x	x	x	x	I	
05	**extra_information_slice**	x	x	x	x	x	I	decoder may skip this data
06	macroblock()	x	x	x	x	x	I	

Table E-14. Macroblock layer

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**macroblock_escape**	x	x	x	x	x	I	
02	**macroblock_address_increment**	x	x	x	x	x	I	
03	macroblock_modes()	x	x	x	x	x	I	
04	**quantiser_scale_code**	x	x	x	x	x	I	
05	motion_vectors(0)	x	x	x	x	x	I	forward motion vector
06	motion_vectors(1)	o	x	x	x	x	I	backward motion vector
07	coded_block_pattern()	x	x	x	x	x	I	
08	block(i)	x	x	x	x	x	I	

Table E-15. Macroblock modes

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**macroblock_type**	x	x	x	x	x	I	
02	**spatial_temporal_weight_code**	o	o	o	x	x	I	
03	**frame_motion_type**	x	x	x	x	x	I	01: Field-based prediction 10: Frame-based prediction 11: Dual-prime
04	**field_motion_type**	x	x	x	x	x	I	01: Field-based prediction 10: 16x8 MC 11: Dual-prime
05	**dct_type**	x	x	x	x	x	I	

Table E-16. Motion vectors

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
01	**motion_vertical_field_select**	x	x	x	x	x	I	
02	motion_vector()	x	x	x	x	x	I	

Table E-17. Motion vector

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
		Status						
01	**motion_horizontal_code**	x	x	x	x	x	I	
02	**motion_horizontal_r**	x	x	x	x	x	I	
03	**dmv_horizontal**	x	x	x	x	x	I	
04	**motion_vertical_code**	x	x	x	x	x	I	
05	**motion_vertical_r**	x	x	x	x	x	I	
06	**dmv_vertical**	x	x	x	x	x	I	

Table E-18. Coded Block Pattern

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
		Status						
01	**coded_block_pattern_420**	x	x	x	x	x	I	
02	**coded_block_pattern_1**	o	o	o	o	x	I	4:2:2
03	**coded_block_pattern_2**	o	o	o	o	o	I	4:4:4

Table E-19. Block layer

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	Type	Comments
		Status						
01	DCT coefficients	x	x	x	x	x	I	
02	End of block	x	x	x	x	x	I	

E.2 Permissible layer combinations (see 8.4.1)

The following tables indicate the parameter limits that apply to each layer of a bitstream, and the minimum profile / level of a compliant decoder capable of fully decoding each layer. Each table describes the limits of a single complinace point in the profile / level matrix.

High Profiles

In the High profile tables, it is assumed that intra_dc_precision = 11 is used only with 4:2:2 chroma_format. Relaxing this restriction results in 'Minimum decoder' always being a High profile decoder and poor interoperability.

Table E-20. High profile @ Main level

# Layers	layer_id	Scalable mode	Chroma Format	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
1	0	-	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
2	0	-	4:2:0	MP@ML	HP@ML	15	1 835 008	HP@ML
2	1	SNR	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
2	0	-	4:2:0	MP@ML	HP@ML	15	1 835 008	HP@ML
2	1	SNR	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
2	0	-	4:2:2	MP@ML	HP@ML	15	1 835 008	HP@ML
2	1	SNR	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
2	0	-	4:2:0	MP@LL	MP@LL	4	475 136	MP@LL
2	1	Spatial	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	0	-	4:2:0	MP@LL	MP@LL	3	360 448	MP@LL
3	1	SNR	4:2:0	MP@LL	MP@LL	4	475 136	SNR@LL
3	2	Spatial	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	0	-	4:2:0	MP@LL	MP@LL	4	475 136	MP@LL
3	1	SNR	4:2:2	MP@LL	MP@LL	15	1 835 008	HP@ML
3	2	Spatial	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
3	0	-	4:2:0	MP@LL	MP@LL	4	475 136	MP@LL
3	1	Spatial	4:2:0	MP@ML	HP@ML	15	1 835 008	HP@ML
3	2	SNR	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	0	-	4:2:0	MP@LL	MP@LL	4	475 136	MP@LL
3	1	Spatial	4:2:0	MP@ML	HP@ML	15	1 835 008	HP@ML
3	2	SNR	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML

Note that the Main level of High profile supports a higher resolution than the Main level of other defined profiles. This is to support the coding of all the active lines of NTSC (483 lines): the Main level of other profiles supports a maximum of 480 lines at 30 Hz frame rate.

Coding of 4:2:2 chrominance as a lower layer of a Spatially scaled hierarchy is not permitted at High profile @ Main level (see table 8-11).

Table E-21. High profile @ High-1440 level

# Layers	layer_id	Scalable mode	Chroma Format	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	4:2:0	MP@H-14	HP@H-14	80	9 781 248	HP@H-14
1	0	-	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
2	0	-	4:2:0	MP@H-14	HP@H-14	20	2 441 216	HP@H-14
2	1	SNR	4:2:0	MP@H-14	HP@H-14	80	9 781 248	HP@H-14
2	0	-	4:2:0	MP@H-14	MP@H-14	20	2 441 216	MP@H-14
2	1	SNR	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
2	0	-	4:2:2	MP@H-14	MP@H-14	20	2 441 216	MP@H-14
2	1	SNR	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
2	0	-	4:2:0	MP@ML	HP@ML	20	2 441 216	MP@H-14
2	1	Spatial	4:2:0	MP@H-14	HP@H-14	80	9 781 248	HP@H-14
2	0	-	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
2	1	Spatial	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	1	SNR	4:2:0	MP@ML	HP@ML	60	7 340 032	HP@ML
3	2	Spatial	4:2:0	MP@H-14	HP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	1	SNR	4:2:2	MP@ML	HP@ML	60	7 340 032	HP@ML
3	2	Spatial	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:2	MP@ML	HP@ML	20	2 441 216	HP@ML
3	1	SNR	4:2:2	MP@ML	HP@ML	60	7 340 032	HP@ML
3	2	Spatial	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:0	MP@ML	HP@ML	20	2 441 216	HP@ML
3	1	Spatial	4:2:0	MP@H-14	HP@H-14	60	7 340 032	HP@H-14
3	2	SNR	4:2:0	MP@H-14	HP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:0	ML@MP	HP@ML	20	2 441 216	HP@ML
3	1	Spatial	4:2:0	MP@H-14	MP@H-14	60	7 340 032	HP@H-14
3	2	SNR	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14
3	0	-	4:2:2	ML@MP	HP@ML	20	2 441 216	HP@ML
3	1	Spatial	4:2:2	MP@H-14	MP@H-14	60	7 340 032	HP@H-14
3	2	SNR	4:2:2	MP@H-14	MP@H-14	80	9 781 248	HP@H-14

Table E-22. High profile @ High level

# Layers	layer_id	Scalable mode	Chroma Format	Maximum sample density	Maximum luminance sample rate	Maximum total bit rate /1 000 000	Maximum total VBV buffer	Minimum decoder
1	0	-	4:2:0	MP@HL	HP@HL	100	12 222 464	HP@HL
1	0	-	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
2	0	-	4:2:0	MP@HL	HP@HL	25	3 047 424	HP@HL
2	1	SNR	4:2:0	MP@HL	HP@HL	100	12 222 464	HP@HL
2	0	-	4:2:0	MP@HL	MP@HL	25	3 047 424	HP@H-14
2	1	SNR	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
2	0	-	4:2:2	MP@HL	MP@HL	25	3 047 424	HP@HL
2	1	SNR	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
2	0	-	4:2:0	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
2	1	Spatial	4:2:0	MP@HL	HP@HL	100	12 222 464	HP@HL
2	0	-	4:2:2	HP@HL, LL	HP@ML	25	3 047 424	HP@H-14
2	1	Spatial	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
3	0	-	4:2:0	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
3	1	SNR	4:2:0	HP@HL, LL	HP@HL, LL	80	9 781 248	HP@H-14
3	2	Spatial	4:2:0	MP@HL	HP@HL	100	12 222 464	HP@HL
3	0	-	4:2:0	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
	1	SNR	4:2:2	HP@HL, LL	HP@HL, LL	80	9 781 248	HP@H-14
3	2	Spatial	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
3	0	-	4:2:2	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
3	1	SNR	4:2:2	HP@HL, LL	HP@HL, LL	80	9 781 248	HP@H-14
3	2	Spatial	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
3	0	-	4:2:0	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
3	1	Spatial	4:2:0	MP@HL	HP@HL	80	9 781 248	HP@HL
3	2	SNR	4:2:0	MP@HL	HP@HL	100	12 222 464	HP@HL
3	0	-	4:2:0	HP@HL, LL	HP@HL, LL	25	3 047 424	HP@H-14
3	1	Spatial	4:2:0	MP@HL	MP@HL	80	9 781 248	HP@HL
3	2	SNR	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL
3	0	-	4:2:2	HP@HL, LL	HP@ML	25	3 047 424	HP@H-14
3	1	Spatial	4:2:2	MP@HL	MP@HL	80	9 781 248	HP@HL
3	2	SNR	4:2:2	MP@HL	MP@HL	100	12 222 464	HP@HL

Note: ‘HP@HL,LL’ indicates the sample density / sample rate is constrained to limits define for the lower layer of High profile, High level.

Annex F
Patent statements

(This annex does not form an integral part of this Recommendation | International Standard)

The user's attention is called to the possibility that, for some of the processes specified in this part of ISO/IEC 13818, conformance with this specification may require use of an invention covered by patent rights.

By publication of this part of ISO/IEC 13818, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. However, each company listed in this Annex has undertaken to file with the Information Technology Task Force (ITTF) a statement of willingness to grant a license under such rights that they hold on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain such a license.

Information regarding such patents can be obtained from the following organisations.

The table summarises the formal patent statements received and indicates the parts of the standard to which the statement applies. The list includes all organisations that have submitted informal patent statements. However, if no "X" is present, no formal patent statement has yet been received from that organisation.

Company	**V**	**A**	**S**
AT&T	X	X	X
BBC Research Department			
Bellcore	X		
Belgian Science Policy Office	X		
BOSCH	X	X	X
CCETT			
CSELT	X		
David Sarnoff Research Center	X	X	X
Deutsche Thomson-Brandt GmbH	X	X	X
France Telecom CNET			
Fraunhofer Gesellschaft		X	X
GC Technology Corporation	X	X	X
General Instruments			
Goldstar			
Hitachi, Ltd.			
International Business Machines Corporation	X	X	X
IRT		X	
KDD	X		
Massachusetts Institute of Technology	X	X	X
Matsushita Electric Industrial Co., Ltd.	X	X	X
Mitsubishi Electric Corporation			
National Transcommunications Limited			
NEC Corporation		X	
Nippon Hoso Kyokai	X		
continued...			

1

Company	**V**	**A**	**S**
Nippon Telegraph and Telephone	X		
Nokia Research Center	X		
Norwegian Telecom Research	X		
Philips Consumer Electronics	X	X	X
OKI			
Qualcomm Incorporated	X		
Royal PTT Nederland N.V., PTT Research (NL)	X	X	X
Samsung Electronics			
Scientific Atlanta	X	X	X
Siemens AG	X		
Sharp Corporation			
Sony Corporation			
Texas Instruments			
Thomson Consumer Electronics			
Toshiba Corporation	X		
TV/Com	X	X	X
Victor Company of Japan Limited			

Annex G

Bibliography

(This annex does not form an integral part of this Recommendation | International Standard)

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

3 C Loeffler, A Ligtenberg, G S Moschytz "Practical fast 1-D DCT algorithms with 11 multiplications" Proceedings IEEE ICASSP-89, Vol. 2, pp 988-991, Feb. 1989

4 See the Normative Reference for ITU-R Rec 601 (formerly CCIR Rec 601)

5 See the Normative Reference for IEC Standard Publication 461

6 See the Normative Reference for ITU-T Rec. H.261

7 See the Normative reference for IEEE Standard Specification P1180-1990

8 ISO/IEC 10918-1 | ITU-T T.81 (JPEG)

9 E Viscito and C Gonzales "A Video Compression Algorithm with Adaptive Bit Allocation and Quantization", Proc SPIE Visual Communications and Image Proc '91 Boston MA November 10-15 Vol. 1605 205, 1991

10 A Puri and R Aravind "Motion Compensated Video Coding with Adaptive Perceptual Quantization", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 1 pp 351 Dec. 1991.

11 C. Gonzales and E. Viscito, "Flexibly scalable digital video coding". Image Communications, Vol. 5, Nos. 1-2, February 1993

12 A.W.Johnson, T.Sikora and T.K. Tan, "Filters for Drift Reduction in Frequency Scalable Video Coding Schemes" *<Transmitted for publication to Electronic Letters.>*

13 R.Mokry and D.Anastassiou, "Minimal Error Drift in Frequency Scalability for Motion-Compensated DCT Coding". IEEE Transactions on Circuits and Systems for Video Technology, *<accepted for publication>*

14 K.N. Ngan, J. Arnold, T. Sikora, T.K. Tan and A.W. Johnson. "Frequency Scalability Experiments for MPEG-2 Standard". Asia-Pacific Conference on Communications, Korea, August 1993.

15 T. Sikora, T.K. Tan and K.N. Ngan, "A Performance Comparison of Frequency Domain Pyramid Scalable Coding Schemes Within the MPEG Framework". Proc. PCS, Picture Coding Symposium, Lausanne, pp. 16.1 - 16.2, Switzerland March 1993.

16 Masahiro Iwahashi, “Motion Compensation Technique for 2:1 Scaled-down Moving Pictures”. 8-14, Picture Coding Symposium '93.

17 Sikora, T. and Pang, K., "Experiments with Optimal Block-Overlapping Filters for Cell Loss Concealment in Packet Video", Proc. IEEE Visual Signal Processing and Communications Workshop, Melbourne, 21-22 Sept. 1993, pp. 247-250.

18 A. Puri "Video Coding Using the MPEG-2 Compression Standard", <to appear> Proc SPIE Visual Communications and Image Proc '93 Boston MA November,1993.

19 A. Puri and A. Wong "Spatial Domain Resolution Scalable Video Coding", <to appear> Proc SPIE Visual Communications and Image Proc '93 Boston MA November,1993.