



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.262

Amendment 4
(02/2012)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving
video

Information technology – Generic coding of moving
pictures and associated audio information: Video:
Frame packing arrangement signalling for 3D
content

Amendment 4

CAUTION !

PREPUBLISHED RECOMMENDATION

This prepublication is an unedited version of a recently approved Recommendation. It will be replaced by the published version after editing. Therefore, there will be differences between this prepublication and the published version.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2012

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

**Information technology –
Generic coding of moving pictures and associated audio information: Video:
Frame packing arrangement signalling for 3D content**

.....

INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: VIDEO

Summary

This Recommendation | International Standard specifies coded representation of video data and the decoding process required to reconstruct pictures. It provides a generic video coding scheme which serves a wide range of applications, bit rates, picture resolutions and qualities. Its basic coding algorithm is a hybrid of motion compensated prediction and DCT. Pictures to be coded can be either interlaced or progressive. Necessary algorithmic elements are integrated into a single syntax, and a limited number of subsets are defined in terms of Profile (functionalities) and Level (parameters) to facilitate practical use of this generic video coding standard.

This third edition of this Recommendation | International Standard consists of ITU-T Rec. H.262 (1995) | ISO/IEC 13818-2:1996, subsequently altered by four corrigenda and ten amendments:

1. A first corrigendum adding a slice picture identifier, allowing an application to define default colour description parameters, removing a prohibition of field-structured DCT coding in progressive frames, clarifying ambiguity on restricted range of reconstructed motion vectors, clarifying ambiguity of VBV at boundaries of sequences, and making various minor corrections.
2. A second corrigendum altering the inverse discrete cosine transform requirements, altering temporal_reference for low delay, and making various minor corrections.
3. A first amendment providing a method of obtaining and registering copyright identifiers.
4. A second amendment defining a 4:2:2 profile.
5. A third amendment adding a camera parameters extension and a multi-view profile.
6. A fourth amendment adding an ITU-T extension.
7. A fifth amendment adding a high level to the 4:2:2 profile.
8. A sixth amendment reducing the upper bound for the number of lines per frame in the high level of all profiles from 1152 to 1088.
9. A third corrigendum clarifying the description of the handling of reserved extension data.
10. A seventh amendment providing an ability to send supplemental "content description data".
11. A fourth corrigendum clarifying and correcting the specification of aspects relating to inverse discrete cosine transform precision.
12. An eighth amendment updating and enhancing the specification of support for colour spaces.
13. A ninth amendment specifying an additional level to support the 1080@50p/60p video format.
14. A tenth amendment specifying a 'frame packing arrangement' signalling for stereoscopic 3D content.

Source

ITU-T Recommendation H.262 was approved on 29February 2012. The identical text is also published as ISO/IEC International Standard 13818-2.

This edition of ITU-T H.262 consolidates H.262 (07/1995), its Amendments 1 and 2 (11/1996), 3 and 4 (02/1998), 5 (05/1999), 6 (02/2000) and Corrigenda 1 and 2 (11/1996), and, relative to the second edition (02/2000), Amendments 1 (11/2000, with Erratum 1 of 04/2002), 2 (01/2007), 3 (03/2009), and 4 (integrated with the third edition approved 02/2012), and Corrigenda 1 (11/2000) and 2 (05/2006).

CONTENTS

Page

Introduction	5
Intro. 1 - Purpose	5
Intro. 2 - Application	5
Intro. 3 - Profiles and levels	5
Intro. 4 - The scalable and the non-scalable syntax	6
1 Scope	10
2 Normative references	10
3 Definitions	10
4 Abbreviations and symbols	16
4.1 Arithmetic operators	16
4.2 Logical operators	17
4.3 Relational operators	17
4.4 Bitwise operators	17
4.5 Assignment	17
4.6 Mnemonics	17
4.7 Constants	18
5 Conventions	18
5.1 Method of describing bitstream syntax	18
5.2 Definition of functions	19
5.3 Reserved, forbidden and marker_bit	19
5.4 Arithmetic precision	19
6 Video bitstream syntax and semantics	20
6.1 Structure of coded video data	20
6.2 Video bitstream syntax	30
6.3 Video bitstream semantics	50
7 The video decoding process	80
7.1 Higher syntactic structures	81
7.2 Variable length decoding	81
7.3 Inverse scan	84
7.4 Inverse quantisation	85
7.5 Inverse DCT	89
7.6 Motion compensation	89
7.7 Spatial scalability	104
7.8 SNR scalability	113
7.9 Temporal scalability	120
7.10 Data partitioning	123
7.11 Hybrid scalability	124
7.12 Output of the decoding process	125
8 Profiles and levels	127
8.1 ISO/IEC 11172-2 compatibility	130
8.2 Relationship between defined profiles	130
8.3 Relationship between defined levels	132
8.4 Scalable layers	132
8.5 Parameter values for defined profiles, levels and layers	135
8.6 Compatibility requirements on decoders	137
9 Registration of Copyright Identifiers	139
9.1 General	139
9.2 Implementation of a Registration Authority (RA)	139

Annex A Inverse discrete cosine transform	141
Annex B Variable length code tables	142
B.1 Macroblock addressing	142
B.2 Macroblock type	143
B.3 Macroblock pattern	148
B.4 Motion vectors	149
B.5 DCT coefficients	150
Annex C Video buffering verifier.....	159
Annex D Features supported by the algorithm	164
D.1 Overview	164
D.2 Video formats.....	164
D.3 Picture quality	165
D.4 Data rate control.....	165
D.5 Low delay mode.....	165
D.6 Random access/channel hopping	166
D.7 Scalability	166
D.8 Compatibility	172
D.9 Differences between this Specification and ISO/IEC 11172-2	172
D.10 Complexity.....	174
D.11 Editing encoded bitstreams	174
D.12 Trick modes	175
D.13 Error resilience.....	176
D.14 Concatenated sequences.....	183
Annex E Profile and level restrictions	184
E.1 Syntax element restrictions in profiles	184
E.2 Permissible layer combinations.....	196
Annex F Bibliography	218
Annex G Registration Procedure	219
G.1 Procedure for the request of a Registered Identifier (RID)	219
G.2 Responsibilities of the Registration Authority	219
G.3 Responsibilities of parties requesting an RID	219
G.4 Appeal procedure for denied applications.....	220
Annex H Registration Application Form.....	221
H.1 Contact information of organization requesting a Registered Identifier (RID).....	221
H.2 Statement of an intention to apply the assigned RID	221
H.3 Date of intended implementation of the RID	221
H.4 Authorized representative	221
H.5 For official use only of the Registration Authority	221
Annex I Registration authority – diagram of administration structure	222
Annex J 4:2:2 Profile test results	223
J.1 Introduction.....	223
Annex K The impact of practices for non-progressive sequence bitstreams in consideration of progressive-scan display	228
K.1 Progressive and non-progressive encoding	228
Annex L Frame packing arrangement signalling for stereoscopic 3D content	234

Information technology – Generic coding of moving pictures and associated audio information: Video: Frame packing arrangement signalling for 3D content

Introduction

Intro. 1 Purpose

This Part of this Recommendation | International Standard 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.

Intro. 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
DSB	Digital Sound Broadcasting (terrestrial and satellite broadcasting)
DTTB	Digital Terrestrial Television Broadcasting
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.)

Intro. 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.

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.

Intro. 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.

Intro. 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 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 quantised DCT information, and encoded using variable length codes.

Intro. 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 Intro. 1 illustrates an example of the relationship among the three different picture types.

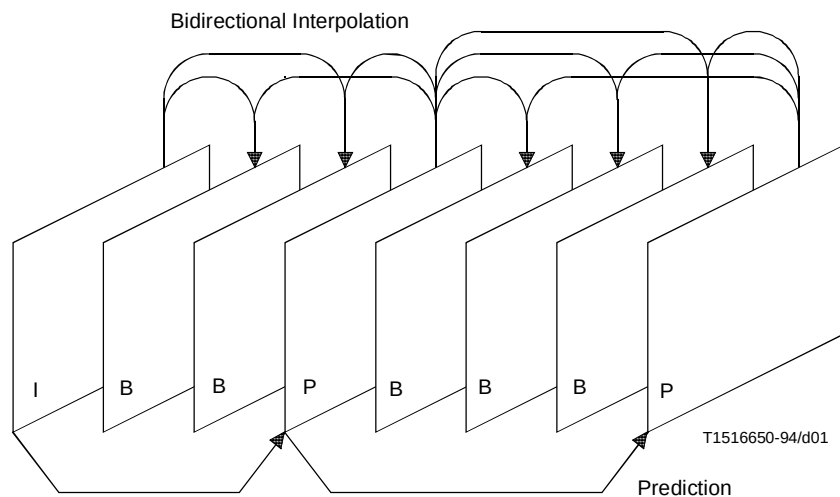


Figure Intro. 1 – Example of temporal picture structure

Intro. 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.

Intro. 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 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. This Specification does not specify how this should be done.

Intro. 4.1.4 Spatial redundancy reduction

Both source pictures and prediction errors 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 resulting prediction error 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 DCT coefficients are zero in value and so two-dimensional run-length and variable length coding is used to encode the remaining DCT coefficients efficiently.

Intro. 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.

Intro. 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 (ATM) networks, 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 re-utilised in coding of the next reproduction of video. In scalable video coding, it is assumed that given a coded 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 Recommendation | International Standard provides several different forms of scalabilities that address non-overlapping 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. Tables Intro. 1 to Intro. 3 provide a few example applications of various scalabilities.

Table Intro. 1 – Applications of SNR scalability

Lower layer	Enhancement layer	Application
Recommendation ITU-R BT.601	Same resolution and format as lower layer	Two quality service for Standard TV (SDTV)
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 Intro. 2 – Applications of spatial scalability

Base	Enhancement	Application
Progressive (30 Hz)	Progressive (30 Hz)	Compatibility or scalability CIF/SCIF
Interlace (30 Hz)	Interlace (30 Hz)	HDTV/SDTV scalability
Progressive (30 Hz)	Interlace (30 Hz)	ISO/IEC 11172-2/compatibility with this Specification
Interlace (30 Hz)	Progressive (60 Hz)	Migration to high resolution progressive HDTV

Table Intro. 3 – Applications of temporal scalability

Base	Enhancement	Higher	Application
Progressive (30 Hz)	Progressive (30 Hz)	Progressive (60 Hz)	Migration to high resolution progressive HDTV
Interlace (30 Hz)	Interlace (30 Hz)	Progressive (60 Hz)	Migration to high resolution progressive HDTV

Intro. 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.

Intro. 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.

Intro. 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.

Intro. 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, low frequency DCT coefficients) are transmitted in the channel with the better error performance, and less critical data (such as higher frequency 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.

INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: VIDEO

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 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 edition of the Recommendations and Standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of ITU maintains a list of currently valid ITU-T Recommendations.

- Recommendation ITU-R BT.601-3 (1992), *Encoding parameters of digital television for studios*.
- Recommendation ITU-R BR.648 (1986), *Digital recording of audio signals*.
- Report ITU-R 955-2 (1990), *Satellite sound broadcasting with portable receivers and receivers in automobiles*.
- ISO/IEC 11172-1:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 1 : Systems*.
- ISO/IEC 11172-2:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2 : Video*.
- ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3 : Audio*.
- ISO/IEC 23002-1:2006, *Information technology – MPEG video technologies – Part 1: Accuracy requirements for implementation of integer-output 8x8 inverse discrete cosine transform*.
- IEC 60908 (1999), *Audiorecording – Compact disc digital audio system*.
- IEC 60461 (1986), *Time and control code for video tape recorders*.
- ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at $p \times 64$ kbit/s*.
- ITU-T Recommendation H.320 (2004), *Narrow-band visual telephone systems and terminal equipment*.
- CCITT Recommendation T.81 (1992) ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines. (JPEG.)*

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 big picture:** A coded picture that would cause VBV buffer underflow as defined in C.7. Big pictures can only occur in sequences where `low_delay` is equal to 1. "Skipped picture" is a term that is sometimes used to describe the same concept.
- 3.3 B-field picture:** A field structure B-Picture.
- 3.4 B-frame picture:** A frame structure B-Picture.
- 3.5 B-picture; bidirectionally predictive-coded picture:** A picture that is coded using motion compensated prediction from past and/or future reference fields or frames.
- 3.6 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.
- 3.7 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.
- 3.8 backward prediction:** Prediction from the future reference frame (field).
- 3.9 base layer:** First, independently decodable layer of a scalable hierarchy.
- 3.10 bitstream; stream:** An ordered series of bits that forms the coded representation of the data.
- 3.11 bitrate:** The rate at which the coded bitstream is delivered from the storage medium to the input of a decoder.
- 3.12 block:** An 8-row by 8-column matrix of samples, or 64 DCT coefficients (source, quantised or dequantised).
- 3.13 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.
- 3.14 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.
- 3.15 byte:** Sequence of 8 bits.
- 3.16 channel:** A digital medium that stores or transports a bitstream constructed according to ITU-T Rec. H.262 | ISO/IEC 13818-2.
- 3.17 chrominance format:** Defines the number of chrominance blocks in a macroblock.
- 3.18 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.
- 3.19 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.
- 3.20 coded B-frame:** A B-frame picture or a pair of B-field pictures.
- 3.21 coded frame:** A coded frame is a coded I-frame, a coded P-frame or a coded B-frame.
- 3.22 coded I-frame:** An I-frame picture or a pair of field pictures, where the first field picture is an I-picture and the second field picture is an I-picture or a P-picture.
- 3.23 coded P-frame:** A P-frame picture or a pair of P-field pictures.
- 3.24 coded picture:** A coded picture is made of a picture header, the optional extensions immediately following it, and the following picture data. A coded picture may be a coded frame or a coded field.
- 3.25 coded video bitstream:** A coded representation of a series of one or more pictures as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2.
- 3.26 coded order:** The order in which the pictures are transmitted and decoded. This order is not necessarily the same as the display order.
- 3.27 coded representation:** A data element as represented in its encoded form.
- 3.28 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.

- 3.29 component:** A matrix, block or single sample from one of the three matrices (luminance and two chrominance) that make up a picture.
- 3.30 compression:** Reduction in the number of bits used to represent an item of data.
- 3.31 constant bitrate coded video:** A coded video bitstream with a constant bitrate.
- 3.32 constant bitrate:** Operation where the bitrate is constant from start to finish of the coded bitstream.
- 3.33 data element:** An item of data as represented before encoding and after decoding.
- 3.34 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.
- 3.35 D-Picture:** A type of picture that shall not be used except in ISO/IEC 11172-2.
- 3.36 DC coefficient:** The DCT coefficient for which the frequency is zero in both dimensions.
- 3.37 DCT coefficient:** The amplitude of a specific cosine basis function.
- 3.38 decoder input buffer:** The First-In First-Out (FIFO) buffer specified in the video buffering verifier.
- 3.39 decoder:** An embodiment of a decoding process.
- 3.40 decoding (process):** The process defined in ITU-T Rec. H.262 | ISO/IEC 13818-2 that reads an input coded bitstream and produces decoded pictures.
- 3.41 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.
- 3.42 digital storage media (DSM):** A digital storage or transmission device or system.
- 3.43 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 ITU-T Rec. H.262 | ISO/IEC 13818-2.
- 3.44 display aspect ratio:** The ratio height/width (in spatial measurement units such as centimeters) of the intended display.
- 3.45 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.
- 3.46 display process:** The (non-normative) process by which reconstructed frames are displayed.
- 3.47 dual-prime prediction:** A prediction mode in which two forward field-based predictions are averaged. The predicted block size is 16×16 luminance samples.
- 3.48 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 ITU-T Rec. H.262 | ISO/IEC 13818-2.
- 3.49 encoder:** An embodiment of an encoding process.
- 3.50 encoding (process):** A process, not specified in ITU-T Rec. H.262 | ISO/IEC 13818-2, that reads a stream of input pictures and produces a valid coded bitstream as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2.
- 3.51 enhancement layer:** A relative reference to a layer (above the base layer) in a scalable hierarchy. For all forms of scalability, its decoding process can be described by reference to the lower layer decoding process and the appropriate additional decoding process for the enhancement layer itself.
- 3.52 fast forward playback:** The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.
- 3.53 fast reverse playback:** The process of displaying the picture sequence in the reverse of display order faster than real-time.
- 3.54 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.
- 3.55 field-based prediction:** A prediction mode using only one field of the reference frame. The predicted block size is 16×16 luminance samples.
- 3.56 field period:** The reciprocal of twice the frame rate.

- 3.57 field picture; field structure picture:** A field structure picture is a coded picture with picture_structure is equal to "Top field" or "Bottom field".
- 3.58 flag:** A one bit integer variable which may take one of only two values (zero and one).
- 3.59 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.
- 3.60 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.
- 3.61 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.
- 3.62 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.
- 3.63 forward prediction:** Prediction from the past reference frame (field).
- 3.64 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.
- 3.65 frame-based prediction:** A prediction mode using both fields of the reference frame.
- 3.66 frame period:** The reciprocal of the frame rate.
- 3.67 frame picture; frame structure picture:** A frame structure picture is a coded picture with picture_structure is equal to "Frame".
- 3.68 frame rate:** The rate at which frames are output from the decoding process.
- 3.69 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.
- 3.70 frame re-ordering:** The process of re-ordering the reconstructed frames when the coded order is different from the display order. Frame re-ordering occurs when B-frames are present in a bitstream. There is no frame re-ordering when decoding low delay bitstreams.
- 3.71 group of pictures:** A notion defined only in ISO/IEC 11172-2 (MPEG-1 Video). In ITU-T Rec. H.262 | ISO/IEC 13818-2, a similar functionality can be achieved by the mean of inserting group of pictures headers.
- 3.72 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.
- 3.73 hybrid scalability:** Hybrid scalability is the combination of two (or more) types of scalability.
- 3.74 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.
- 3.75 I-field picture:** A field structure I-Picture.
- 3.76 I-frame picture:** A frame structure I-Picture.
- 3.77 I-picture; intra-coded picture:** A picture coded using information only from itself.
- 3.78 intra coding:** Coding of a macroblock or picture that uses information only from that macroblock or picture.
- 3.78.1 Inverse DCT, IDCT:** Inverse discrete cosine transform, as defined in Annex A.
- 3.79 level:** A defined set of constraints on the values which may be taken by the parameters of ITU-T Rec. H.262 | ISO/IEC 13818-2 within a particular profile. A profile may contain one or more levels. In a different context, level is the absolute value of a non-zero coefficient (see "run").
- 3.80 layer:** In a scalable hierarchy denotes one out of the ordered set of bitstreams and (the result of) its associated decoding process (implicitly including decoding of **all** layers below this layer).
- 3.81 layer bitstream:** A single bitstream associated to a specific layer (always used in conjunction with layer qualifiers, e. g. "enhancement layer bitstream").
- 3.82 lower layer:** A relative reference to the layer immediately below a given enhancement layer (implicitly including decoding of **all** layers below this enhancement layer).

- 3.83 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.
- 3.84 Mbit:** 1 000 000 bits.
- 3.85 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 ITU-T Rec. H.262 | ISO/IEC 13818-2. The usage is clear from the context.
- 3.86 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.
- 3.87 motion estimation:** The process of estimating motion vectors during the encoding process.
- 3.88 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.
- 3.89 non-intra coding:** Coding of a macroblock or picture that uses information both from itself and from macroblocks and pictures occurring at other times.
- 3.90 opposite parity:** The opposite parity of top is bottom, and vice versa.
- 3.91 P-field picture:** A field structure P-Picture.
- 3.92 P-frame picture:** A frame structure P-Picture.
- 3.93 P-picture; predictive-coded picture:** A picture that is coded using motion compensated prediction from past reference fields or frame.
- 3.94 parameter:** A variable within the syntax of ITU-T Rec. H.262 | ISO/IEC 13818-2 which may take one of a range of values. A variable which can take one of only two values is called a flag.
- 3.95 parity (of field):** The parity of a field can be top or bottom.
- 3.96 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.
- 3.97 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. A "coded picture" is defined in 3.21 of ITU-T Rec. H.262 | ISO/IEC 13818-2. 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.
- 3.98 picture data:** In the VBV operations, picture data is defined as all the bits of the coded picture, all the header(s) and user data immediately preceding it if any (including any stuffing between them) and all the stuffing following it, up to (but not including) the next start code, except in the case where the next start code is an end of sequence code, in which case it is included in the picture data.
- 3.99 prediction:** The use of a predictor to provide an estimate of the sample value or data element currently being decoded.
- 3.100 prediction error:** The difference between the actual value of a sample or data element and its predictor.
- 3.101 predictor:** A linear combination of previously decoded sample values or data elements.
- 3.102 profile:** A defined subset of the syntax of ITU-T Rec. H.262 | ISO/IEC 13818-2.
NOTE – In ITU-T Rec. H.262 | ISO/IEC 13818-2, 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.
- 3.103 progressive:** The property of film frames where all the samples of the frame represent the same instances in time.
- 3.104 quantisation matrix:** A set of sixty-four 8-bit values used by the dequantiser.
- 3.105 quantised DCT coefficients:** DCT coefficients before dequantisation. A variable length coded representation of quantised DCT coefficients is transmitted as part of the coded video bitstream.
- 3.106 quantiser scale:** A scale factor coded in the bitstream and used by the decoding process to scale the dequantisation.

- 3.107 random access:** The process of beginning to read and decode the coded bitstream at an arbitrary point.
- 3.108 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.
- 3.109 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.
- 3.110 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.
- 3.111 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.
- 3.112 re-ordering delay:** A delay in the decoding process that is caused by frame re-ordering.
- 3.113 reserved:** The term "reserved" when used in the clauses defining the coded bitstream, indicates that the value may be used in the future for ITU-T | ISO/IEC defined extensions.
- 3.114 sample aspect ratio (SAR):** This specifies the relative distance between samples. It is defined (for the purposes of ITU-T Rec. H.262 | ISO/IEC 13818-2), 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).
- 3.115 scalable hierarchy:** Coded video data consisting of an ordered set of more than one video bitstream.
- 3.116 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.
- 3.117 side information:** Information in the bitstream necessary for controlling the decoder.
- 3.118 16 × 8 prediction:** A prediction mode similar to field-based prediction but where the predicted block size is 16 × 8 luminance samples.
- 3.119 run:** The number of zero coefficients preceding a non-zero coefficient, in the scan order. The absolute value of the non-zero coefficient is called "level".
- 3.120 saturation:** Limiting a value that exceeds a defined range by setting its value to the maximum or minimum of the range as appropriate.
- 3.121 skipped macroblock:** A macroblock for which no data is encoded.
- 3.122 slice:** A consecutive series of macroblocks which are all located in the same horizontal row of macroblocks.
- 3.123 SNR scalability:** A type of scalability where the enhancement layer(s) contain only coded refinement data for the DCT coefficients of the lower layer.
- 3.124 source; input:** Term used to describe the video material or some of its attributes before encoding.
- 3.125 spatial prediction:** Prediction derived from a decoded frame of the lower layer decoder used in spatial scalability.
- 3.126 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.
- 3.127 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.
- 3.128 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 which would otherwise be lower than the desired bitrate.

- 3.129 temporal prediction:** Prediction derived from reference frames or fields other than those defined as spatial prediction.
- 3.130 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.
- 3.131 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.
- 3.132 top layer:** The topmost layer (with the highest layer_id) of a scalable hierarchy.
- 3.133 variable bitrate:** Operation where the bitrate varies with time during the decoding of a coded bitstream.
- 3.134 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.
- 3.135 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.
- 3.136 video sequence:** The highest syntactic structure of coded video bitstreams. It contains a series of one or more coded frames.
- 3.137 xxx profile decoder:** Decoder able to decode one or a scalable hierarchy of bitstreams of which the top layer conforms to the specifications of the xxx profile (with xxx being any of the defined Profile names).
- 3.138 xxx profile scalable hierarchy:** Set of bitstreams of which the top layer conforms to the specifications of the xxx profile.
- 3.139 xxx profile bitstream:** A bitstream of a scalable hierarchy with a profile indication corresponding to xxx. Note that this bitstream is only decodable together with all its lower layer bitstreams (unless it is a base layer bitstream).
- 3.140 zigzag 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, i.e. $x++$ is equivalent to $x = x + 1$
---	Decrement, i.e. $x---$ is equivalent to $x = x - 1$
$\times$ * }	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 $3 \text{ DIV } 2$ is rounded to 1, and $-3 \text{ DIV } 2$ 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.

$$\text{Sign}() \quad \text{Sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

$$\text{Abs}() \quad \text{Abs}(x) = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

$\sum_{i=a}^{i=b} f(i)$ The summation of the $f(i)$ with i taking integral values from a up to, but not including b .

$\text{Floor}()$ the largest integer less than or equal to the argument.

$\text{Round}()$ $\text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5)$, for an argument x .

4.2 Logical operators

$\parallel$ 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
 $\text{max} [, \dots ,]$ The maximum value in the argument list
 $\text{min} [, \dots ,]$ 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 this Specification. Bit strings are generally 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. For convenience, large strings are occasionally written in hexadecimal; in this case, conversion to a binary in the conventional manner will yield the value of the bit string. Thus, the left most hexadecimal digit is first and in each hexadecimal digit the most significant of the four bits is first.

uimbsf 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 the most significant byte first.

In the clauses describing the syntax, any syntactic element that can only take positive or unsigned values (such as a flag that can be equal to 0 or 1) is described with the mnemonic 'uimbsf'. If the syntactic element can have a negative value, it is described with the mnemonic 'simsbf'. If the syntactic element has a constant value (e.g. marker_bit) then it is described with the mnemonic 'bslbf'. If the syntactic element represents a variable-length code, it is described with the mnemonic 'vlcblf'.

4.7 Constants

π 3,141 592 653 58...

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) { data_element ... }	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.
do { data_element ... } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.
if (condition) { data_element ... } else { data_element ... }	If the condition is true, then the first group of data elements occurs next in the data stream. If the condition is not true, then the second group of data elements occurs next in the data stream.
for (i = m; i < n; i++) { data_element ... }	The group of data elements occurs (m – 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 by one for the second occurrence, and so forth.
/* 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 and a variable or expression evaluating to a zero value is equivalent to a condition that is false. 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.

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

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 stuffing 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 ITU-T | 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 integer in which the value zero is forbidden (and it therefore shall have the value '1'). 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:

- Where an arithmetically-precise result is not fully 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.
- Where ranges of values are given by a colon, the end points are included if a bracket is present, and excluded if the 'less than' (<) and 'greater than' (>) 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 and 7.12 for a description of the decoding process for the base layer, except in the case of Data partitioning, 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 for a description of the decoding process for scalable hierarchy.

See Rec. ITU-T H.222.0 | 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.)

6.1.1.1 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.

6.1.1.2 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.

6.1.1.3 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.

6.1.1.4 Picture

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

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

A P-frame picture or a pair of P-field pictures is called a coded P-frame.

A B-frame picture or a pair of B-field pictures is called a coded B-frame.

A coded I-frame, a coded P-frame or a coded B-frame is called a coded frame.

A reconstructed picture is obtained by decoding a coded picture, i.e. a picture header, the optional 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).

6.1.1.4.1 Field pictures

If field pictures are used, then they shall occur in pairs (one top field followed by one bottom field, or one bottom field followed by one top 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. If the second picture is a P-field picture, then certain restrictions apply (see 7.6.3.5).

6.1.1.4.2 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.

6.1.1.5 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 reference frame or past reference field;
- a **Bidirectionally predictive-coded (B) picture** is a picture which is coded using motion compensated prediction from a past and/or future reference frame(s).

6.1.1.6 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 do not apply, and the present Specification applies.

In repeat 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, the first picture following a sequence header or a repeated sequence header shall be 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 repeat 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.

6.1.1.7 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 or intra coding. This is more precisely defined in the clause 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.

6.1.1.8 4:2:0 format

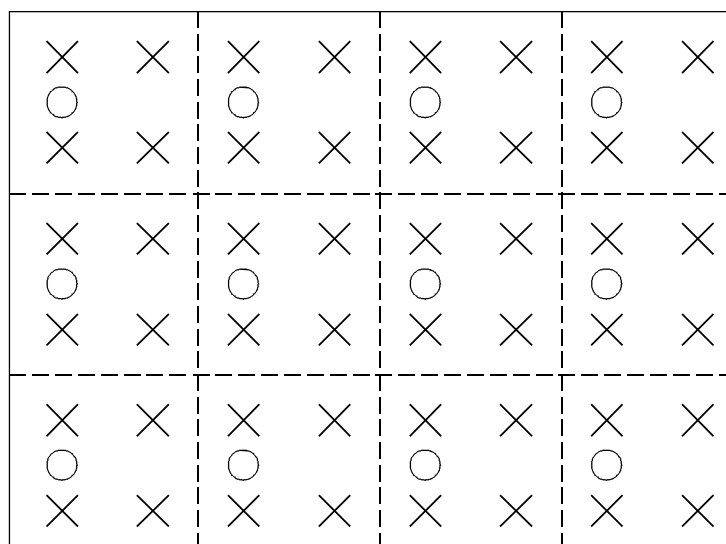
In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

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 6-2 and 6-3 show the vertical and temporal positioning of the samples in an interlaced frame. Figure 6-4 shows the vertical and temporal positioning of the samples in a progressive frame.

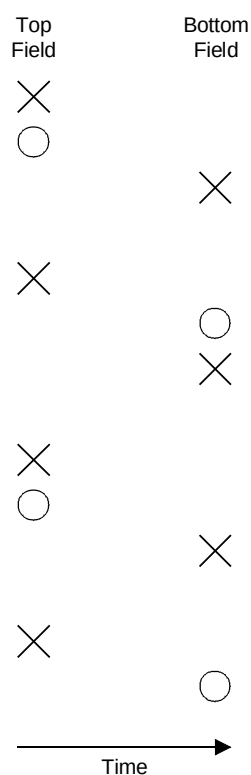
In each field of an interlaced frame, the chrominance samples do not lie (vertically) midway between the luminance samples of the field, this is so that the spatial location of the chrominance samples in the frame is the same whether the frame is represented as a single frame-picture or two field-pictures.



T1515950-94/d02

- × Represents luminance samples
- Represents chrominance samples

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



T1515960-94/d03

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

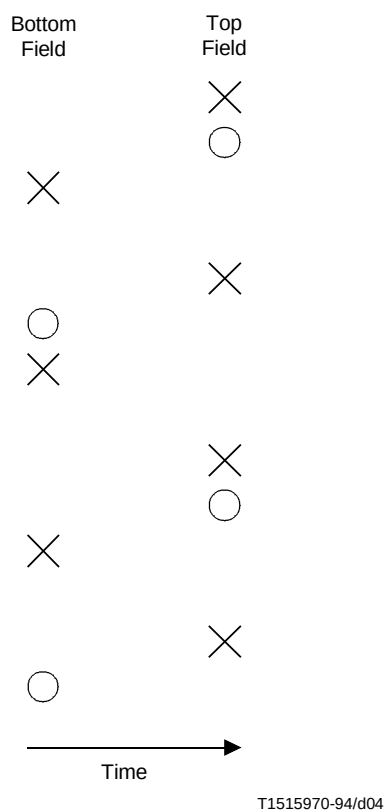


Figure 6-3 – Vertical and temporal positions of samples in an interlaced frame with top_field_first = 0



Figure 6-4 – Vertical and temporal positions of samples in a progressive frame

6.1.1.9 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 6-5.

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

6.1.1.10 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 6-6 and 6-7.

6.1.1.11 Frame re-ordering

When the sequence contains coded B-frames, the number of consecutive coded B-frames is variable and unbounded. The first coded frame after a sequence header shall not be a B-frame.

A sequence may contain no coded P-frames. A sequence may also contain no coded I-frames 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.

The order of the coded frames in the bitstream, also called coded order, is the order in which a decoder reconstructs them. The order of the reconstructed frames at the output of the decoding process, also called the display order, is not always the same as the coded order and this subclause defines the rules of frame re-ordering that shall happen within the decoding process.

When the sequence contains no coded B-frames, the coded order is the same as the display order. This is true in particular always when low_delay is one.

When B-frames are present in the sequence, re-ordering is performed according to the following rules:

- If the current frame in coded order is a B-frame, the output frame is the frame reconstructed from that B-frame.
- If the current frame in coded order is a I-frame or P-frame, the output frame is the frame reconstructed from the previous I-frame or P-frame if one exists. If none exists, at the start of the sequence, no frame is output.

The frame reconstructed from the final I-frame or P-frame is output immediately after the frame reconstructed when the last coded frame in the sequence was removed from the VBV buffer.

The following is an example of frames 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
---	---	---	---	---	---	---	---	---	----	----	----	----

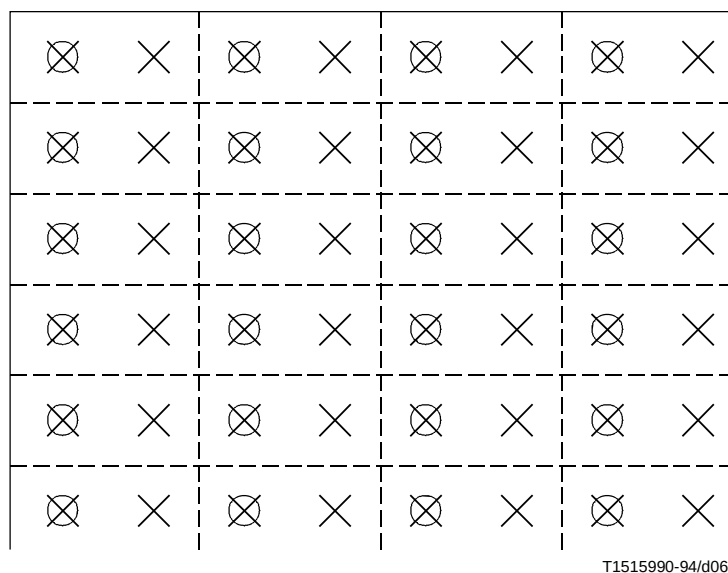


Figure 6-5 – The position of luminance and chrominance samples – 4:2:2 data

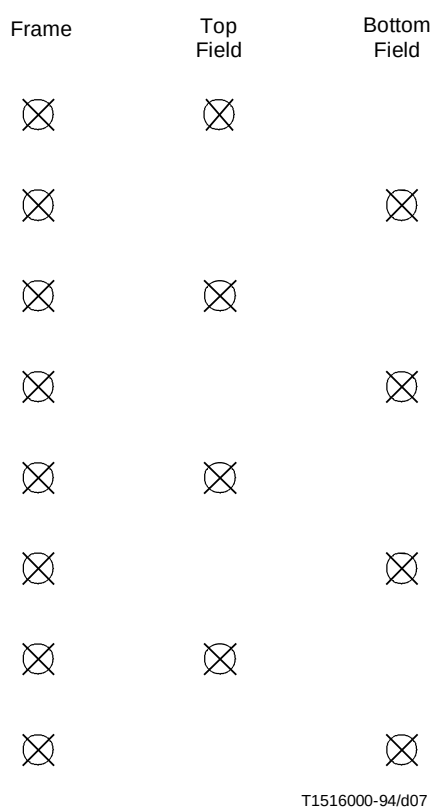
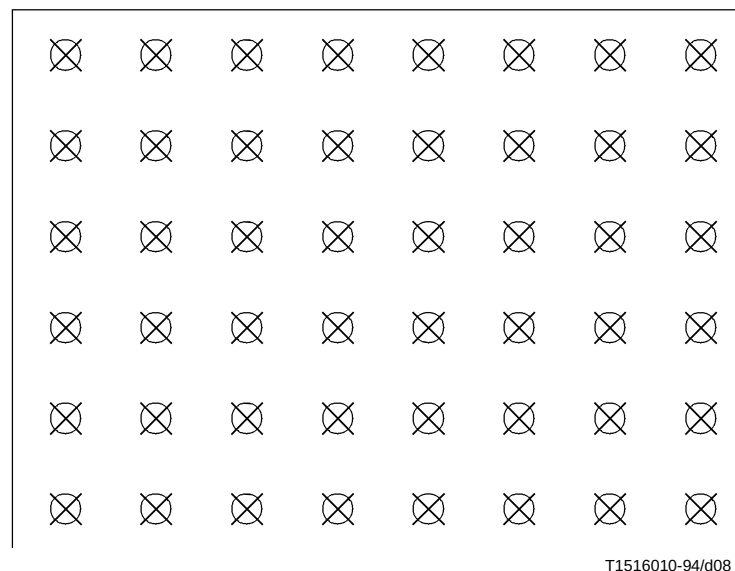


Figure 6-6 – Vertical positions of samples with 4:2:2 and 4:4:4 data



 Represents luminance samples
 Represents chrominance samples

Figure 6-7 – The position of luminance and chrominance samples – 4:4:4 data

6.1.2 Slice

A **slice** is a series of an arbitrary number of consecutive 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 Figures 6-8 and 6-9 as alphabetical order).

6.1.2.1 The general slice structure

In the most general case it is not necessary for the slices to cover the entire picture. Figure 6-8 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.

6.1.2.2 Restricted slice structure

In certain defined levels of defined profiles a restricted slice structure illustrated in Figure 6-9 shall be used. In this case every macroblock in the picture shall be enclosed in a slice.

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

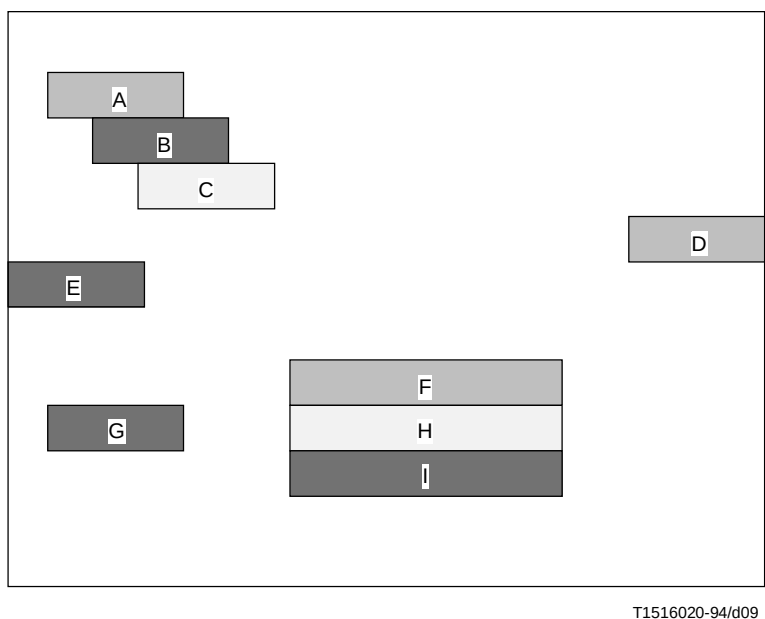


Figure 6-8 – The most general slice structure

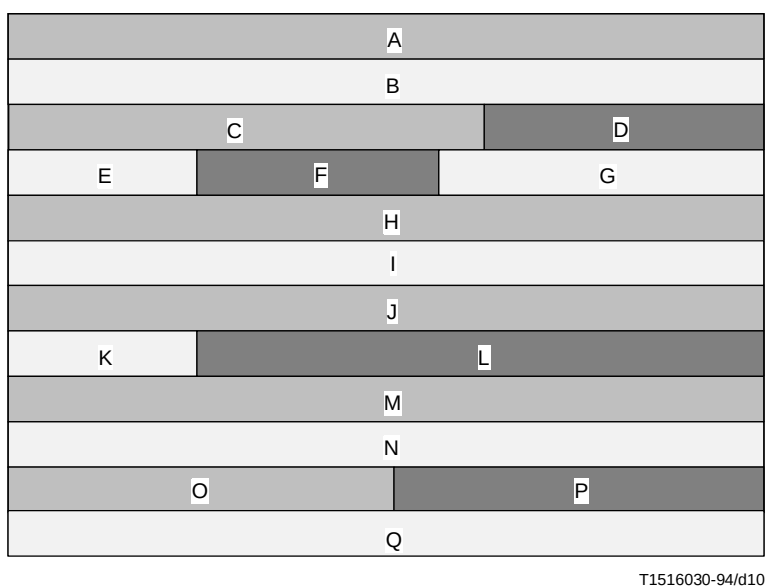


Figure 6-9 – Restricted slice structure

6.1.3 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 6-10.

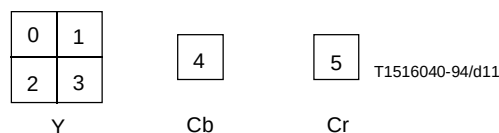


Figure 6-10 – 4:2:0 Macroblock structure

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 6-11.

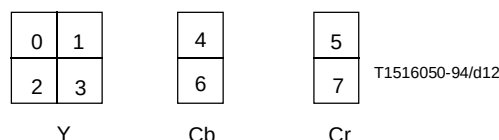


Figure 6-11 – 4:2:2 Macroblock structure

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 6-12.

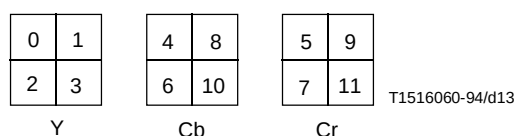


Figure 6-12 – 4:4:4 Macroblock structure

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 6-13.
- 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 6-14.

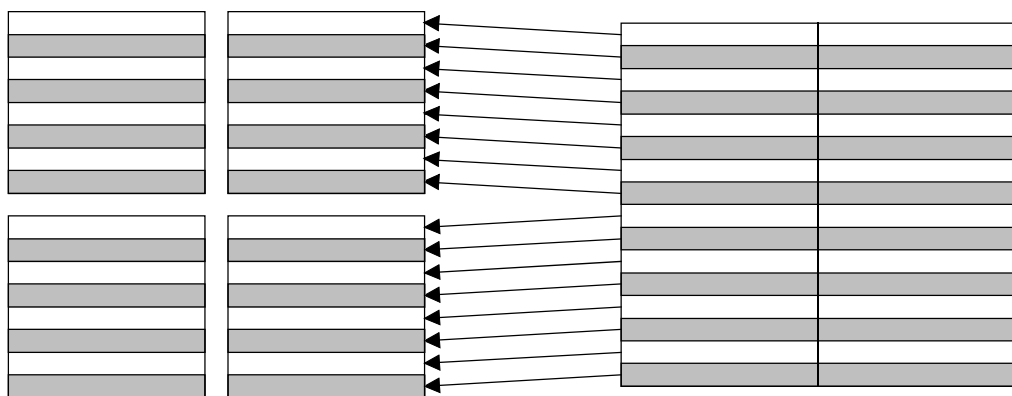
In the case of chrominance blocks the structure depends upon the chrominance format that is being used. In the case of 4:2:2 and 4:4:4 formats (where there are two blocks in the vertical dimension of the macroblock) the chrominance blocks are treated in exactly the same manner as the luminance blocks. 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 8×4 regions (after half-sample filtering) must be made.

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 6-13.

6.1.4 Block

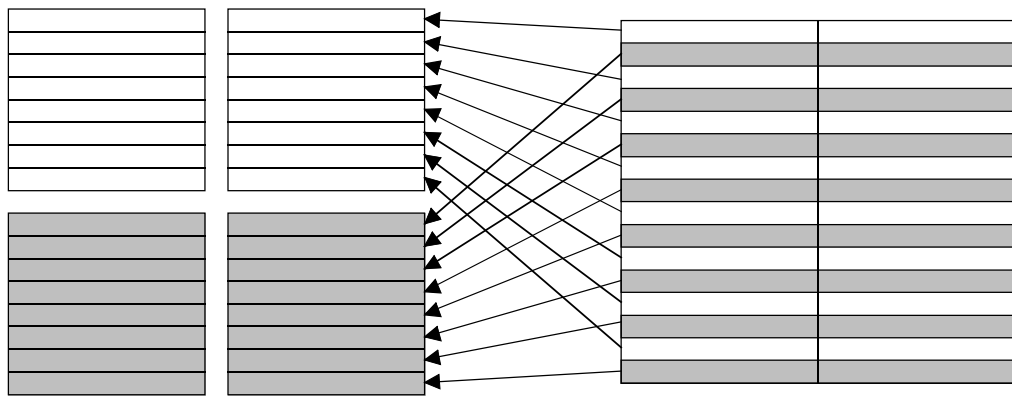
The term "**block**" can refer either to source and reconstructed data or to the DCT coefficients or to the corresponding coded data elements.

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.



T1520180-95/d14

Figure 6-13 – Luminance macroblock structure in frame DCT coding



T1520190-95/d15

Figure 6-14 – Luminance macroblock structure in field DCT coding

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 first (most significant) bit of a byte.

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

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.

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 (Note)	B9 through FF
NOTE – System start codes are defined in Part 1 of this Specification.	

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()		
extension_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
}		

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	bslbf
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
next_start_code()		
}		

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) && (nextbits() == extension_start_code))		
extension_data(i)		
if (nextbits() == user_data_start_code)		
user_data()		
}		
}		

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()		
else if (nextbits()		
== "Sequence Scalable Extension ID")		
sequence_scalable_extension()		
else		
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
reserved_extension_data_byte	8	uimsbf
}		
/* NOTE – i never takes the value 1 because extension_data()		
never follows a group_of_pictures_header() */		
if (i == 2) { /* follows picture_coding_extension() */		
if (nextbits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "Copyright Extension ID")		
copyright_extension()		
else if (nextbits() == "Picture Display Extension ID")		
picture_display_extension()		
else if (nextbits()		
== "Picture Spatial Scalable Extension ID")		
picture_spatial_scalable_extension()		
else if (nextbits()		
== "Picture Temporal Scalable Extension ID")		
picture_temporal_scalable_extension()		
else if (nextbits()		
== "Camera Parameters Extension ID")		
camera_parameters_extension()		
else if (nextbits()		
== "ITU-T Extension ID")		
ITU-T_extension()		
else		
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
reserved_extension_data_byte	8	uimsbf
}		
}		

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	uimsbf
}		
next_start_code()		
}		

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()		
}		

Copyright International Telecommunication Union (ITU-T) 2012. All rights reserved.

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	uimsbf
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	bslbf
forward_f_code	3	bslbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	bslbf
backward_f_code	3	bslbf
}		
while (nextbits() == '1') {		
extra_bit_picture /* with the value '1' */	1	uimsbf
content_description_data() /* with every 9th bit having the value '1' */	8	uimsbf
}		
extra_bit_picture /* with the value '0' */	1	uimsbf
next_start_code()		
}		

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()		
}		

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 Copyright extension

copyright_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
copyright_flag	1	uimsbf
copyright_identifier	8	uimsbf
original_or_copy	1	uimsbf
reserved	7	bslbf
marker_bit	1	bslbf
copyright_number_1	20	uimsbf
marker_bit	1	bslbf
copyright_number_2	22	uimsbf
marker_bit	1	bslbf
copyright_number_3	22	uimsbf
next_start_code()		
}		

6.2.3.7 Picture data

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

6.2.3.7.1 Camera parameters extension

camera_parameters_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
reserved	1	uimsbf
camera_id	7	simsbf
marker_bit	1	bslbf
height_of_image_device	22	uimsbf
marker_bit	1	bslbf
focal_length	22	uimsbf
marker_bit	1	bslbf
f_number	22	uimsbf
marker_bit	1	bslbf
vertical_angle_of_view	22	uimsbf
marker_bit	1	bslbf
camera_position_x_upper	16	simsbf
marker_bit	1	bslbf
camera_position_x_lower	16	
marker_bit	1	bslbf
camera_position_y_upper	16	simsbf
marker_bit	1	bslbf
camera_position_y_lower	16	
marker_bit	1	bslbf
camera_position_z_upper	16	simsbf
marker_bit	1	bslbf
camera_position_z_lower	16	
marker_bit	1	bslbf
camera_direction_x	22	simsbf
marker_bit	1	bslbf
camera_direction_y	22	simsbf
marker_bit	1	bslbf
camera_direction_z	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_x	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_y	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_z	22	simsbf
marker_bit	1	bslbf
reserved	32	bslbf
next_start_code()		
}		
NOTE – The parameters marked 'reserved' in the camera parameters extension shall be equal to zero. Other values are reserved for future use by ITU-T ISO/IEC.		

6.2.3.7.2 ITU-T extension

ITU-T_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
while(nextbits() != '0000 0000 0000 0000 0000 0001') {		
ITU-T_data	1	uimsbf
}		
next_start_code()		
}		
NOTE – The construct with the while-statement prevents start code emulation.		

6.2.3.7.3 Content description data

content_description_data() {	No. of bits	Mnemonic
data_type_upper	8	uimsbf
marker_bit	1	bslbf
data_type_lower	8	
marker_bit	1	bslbf
data_length	8	uimsbf
if (data_type == "Padding Bytes")		
padding_bytes()		
else if (data_type == "Capture Timecode")		
capture_timecode()		
else if (data_type == "Additional Pan-Scan Parameters")		
additional_pan_scan_parameters()		
else if (data_type == "Active Region Window")		
active_region_window()		
else if (data_type == "Coded Picture Length")		
coded_picture_length()		
else		
for (i = 0; i < data_length; i ++) {		
marker_bit	1	bslbf
reserved_content_description_data	8	uimsbf
}		
}		

6.2.3.7.3.1 Padding bytes

padding_bytes() {	No. of bits	Mnemonic
for (i = 0; i < data_length; i ++) {		
marker_bit	1	bslbf
padding_byte	8	bslbf
}		
}		

6.2.3.7.3.2 Capture timecode

capture_timecode() {	No. of bits	Mnemonic
marker_bit	1	bslbf
timecode_type	2	uimsbf
counting_type	3	uimsbf
reserved_bit	1	uimsbf
reserved_bit	1	uimsbf
reserved_bit	1	uimsbf
if (counting_type != 0) {		
marker_bit	1	bslbf
nframes_conversion_code	1	uimsbf
clock_divisor	7	uimsbf
marker_bit	1	bslbf
nframes_multiplier_upper	8	uimsbf
marker_bit	1	bslbf
nframes_multiplier_lower	8	
}		
frame_or_field_capture_timestamp()		
if (timecode_type == '11')		
frame_or_field_capture_timestamp()		
}		

6.2.3.7.3.2.1 Frame or field capture timestamp

frame_or_field_capture_timestamp() {	No. of bits	Mnemonic
if (counting_type != 0) {		
marker_bit	1	bslbf
nframes	8	uimsbf
}		
marker_bit	1	bslbf
time_discontinuity	1	uimsbf
prior_count_dropped	1	uimsbf
time_offset_part_a	6	simsbf
marker_bit	1	bslbf
time_offset_part_b	8	
marker_bit	1	bslbf
time_offset_part_c	8	
marker_bit	1	bslbf
time_offset_part_d	8	
marker_bit	1	bslbf
units_of_seconds	4	uimsbf
tens_of_seconds	4	uimsbf
marker_bit	1	bslbf
units_of_minutes	4	uimsbf
tens_of_minutes	4	uimsbf
marker_bit	1	bslbf
units_of_hours	4	uimsbf
tens_of_hours	4	uimsbf
}		

6.2.3.7.3.3 Additional pan-scan parameters

additional_pan_scan_parameters() {	No. of bits	Mnemonic
marker_bit	1	bslbf
aspect_ratio_information	4	uimsbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_size_present	1	bslbf
if (display_size_present == '1') {		
marker_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_horizontal_size_upper	6	uimsbf
marker_bit	1	bslbf
display_horizontal_size_lower	8	
marker_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_vertical_size_upper	6	uimsbf
marker_bit	1	bslbf
display_vertical_size_lower	8	
}		
for (i = 0; i < number_of_frame_centre_offsets; i ++) {		
marker_bit	1	bslbf
frame_centre_horizontal_offset_upper	8	simsbf
marker_bit	1	bslbf
frame_centre_horizontal_offset_lower	8	
marker_bit	1	bslbf
frame_centre_vertical_offset_upper	8	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset_lower	8	
}		
}		

6.2.3.7.3.4 Active region window

active_region_window() {	No. of bits	Mnemonic
marker_bit	1	bslbf
top_left_x_upper	8	uimsbf
marker_bit	1	bslbf
top_left_x_lower	8	
marker_bit	1	bslbf
top_left_y_upper	8	uimsbf
marker_bit	1	bslbf
top_left_y_lower	8	
marker_bit	1	bslbf
active_horizontal_size_upper	8	uimsbf
marker_bit	1	bslbf
active_horizontal_size_lower	8	
marker_bit	1	bslbf
active_vertical_size_upper	8	uimsbf
marker_bit	1	bslbf
active_vertical_size_lower	8	
}		

6.2.3.7.3.5 Coded picture length

coded_picture_length() {	No. of bits	Mnemonic
marker_bit	1	bslbf
picture_byte_count_part_a	8	uimsbf
marker_bit	1	bslbf
picture_byte_count_part_b	8	
marker_bit	1	bslbf
picture_byte_count_part_c	8	
marker_bit	1	bslbf
picture_byte_count_part_d	8	
}		

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') {		
slice_extension_flag	1	bslbf
intra_slice	1	uimsbf
slice_picture_id_enable	1	uimsbf
slice_picture_id	6	uimsbf
while (nextbits() == '1') {		
extra_bit_slice /* with the value '1' */	1	uimsbf
extra_information_slice	8	uimsbf
}		
}		
extra_bit_slice /* with the value '0' */	1	uimsbf
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

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)		
}		
}		

Copyright International Telecommunication Union 2000. All rights reserved.

6.2.5.1 Macroblock modes

macroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlc1bf
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 ((picture_structure == "Frame picture") &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra macroblock_pattern)) {		
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) && (dmvc != 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)		
}		
}		