

1 **INTERNATIONAL ORGANISATION FOR STANDARDIZATION**
2 **ORGANISATION INTERNATIONALE DE NORMALISATION**
3 **ISO/IEC JTC1/SC29**
4 **CODING OF MOVING PICTURES AND ASSOCIATED AUDIO**

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

ISO/IEC JTC1/SC29
WG11/602
November 1993, Seoul

24 **INFORMATION TECHNOLOGY -**
25 **GENERIC CODING OF MOVING PICTURES AND**
26 **ASSOCIATED AUDIO**
27 **Recommendation H.262**
28 **ISO/IEC 13818-2**
29 **Committee Draft**

1

CONTENTS

2		CONTENTS	i
3		Foreword	vii
4	I	Introduction	viii
5	I.1	Purpose	viii
6	I.2	Application	viii
7	I.3	Profiles and levels	viii
8	I.4	The scalable and the non-scalable syntax	ix
9	I.4.1	Overview of the non-scalable syntax	ix
10	I.4.1.1	Temporal processing	ix
11	I.4.1.2	Coding interlaced video	x
12	I.4.1.3	Motion representation - macroblocks	x
13	I.4.1.4	Spatial redundancy reduction	x
14	I.4.1.5	Chroma formats	x
15	I.4.2	Scalable extensions	x
16	I.4.2.1	Spatial scalable extension	xi
17	I.4.2.2	SNR scalable extension	xii
18	I.4.2.3	Temporal scalable extension	xii
19	I.4.2.4	Data partitioning extension	xii
20	1	Scope	1
21	2	Normative references	1
22	3	Definitions	2
23	4	Abbreviations and symbols	7
24	4.1	Arithmetic operators	7
25	4.2	Logical operators	7
26	4.3	Relational operators	7
27	4.4	Bitwise operators	8
28	4.5	Assignment	8
29	4.6	Mnemonics	8
30	4.7	Constants	8
31	5	Conventions	9
32	5.1	Method of describing bitstream syntax	9
33	5.2	Definition of functions	10
34	5.2.1	Definition of bytealigned() function	10
35	5.2.2	Definition of nextbits() function	10
36	5.2.3	Definition of next_start_code() function	10
37	5.3	Reserved, forbidden and marker_bit	10
38	5.4	Arithmetic precision	10
39	6	Video bitstream syntax and semantics	11
40	6.1	Structure of video data	11
41	6.1.1	Video sequence	11
42	6.1.1.1	Frame reordering	11
43	6.1.1.2	Sequence header	12
44	6.1.1.3	Group of pictures header	12
45	6.1.2	Picture	12
46	6.1.2.1	4:2:0 Format	13
47	6.1.2.2	4:2:2 Format	14
48	6.1.2.3	4:4:4 Format	16
49	6.1.2.4	Picture Types	16
50	6.1.2.5	Progressive and interlaced sequences	16
51	6.1.2.5.1	Field pictures	16
52	6.1.2.5.2	Frame pictures	17
53	6.1.3	Slice	17
54	6.1.3.1	The general slice structure	17
55	6.1.3.2	Restricted slice structure	17
56	6.1.4	Macroblock	18
57	6.1.5	Block	19
58	6.2	Video bitstream syntax	20
59	6.2.1	Start codes	20
60	6.2.2	Video Sequence	21

1	6.2.2.1	Sequence header.....	22
2	6.2.2.2	Extension and user data	22
3	6.2.2.2.1	Extension data.....	23
4	6.2.2.2.2	User data	23
5	6.2.2.3	Sequence extension.....	24
6	6.2.2.4	Sequence display extension	24
7	6.2.2.5	Sequence scalable extension.....	25
8	6.2.2.6	Group of pictures header	25
9	6.2.3	Picture header.....	26
10	6.2.3.1	Picture coding extension.....	27
11	6.2.3.2	Quant matrix extension	28
12	6.2.3.3	Picture display extension	28
13	6.2.3.4	Picture temporal scalable extension	28
14	6.2.3.5	Picture spatial scalable extension.....	29
15	6.2.3.6	Picture data.....	29
16	6.2.4	Slice.....	30
17	6.2.5	Macroblock	31
18	6.2.5.1	Macroblock modes.....	32
19	6.2.5.2	Motion vectors	32
20	6.2.5.2.1	Motion vector	33
21	6.2.5.3	Coded block pattern	33
22	6.2.6	Block layer	34
23	6.3	Video bitstream semantics	35
24	6.3.1	Semantic rules for higher syntactic structures	35
25	6.3.2	Video sequence	36
26	6.3.3	Sequence header.....	36
27	6.3.4	Extension and user data.....	39
28	6.3.5	Sequence extension	39
29	6.3.6	Sequence display extension	41
30	6.3.7	Quant matrix extension	45
31	6.3.8	Sequence scalable extension	46
32	6.3.9	Group of pictures header	47
33	6.3.10	Picture header.....	48
34	6.3.11	Picture Coding Extension	49
35	6.3.12	Picture display extension	51
36	6.3.12.1	Pan-scan	52
37	6.3.13	Picture spatial scalable extension.....	53
38	6.3.14	Picture temporal scalable extension	53
39	6.3.15	Slice header	53
40	6.3.16	Macroblock	54
41	6.3.17	Block	57
42	7	The video decoding process	58
43	7.1	Higher syntactic structures	58
44	7.2	Variable length decoding	59
45	7.2.1	DC intra coefficients	59
46	7.2.2	Other coefficients.....	60
47	7.2.2.1	Table selection	60
48	7.2.2.2	First coefficient of a non-intra block	61
49	7.2.2.3	Escape coding	61
50	7.2.2.4	Summary	61
51	7.3	Inverse scan	61
52	7.3.1	Inverse scan for matrix download	62
53	7.4	Inverse Quantisation.....	62
54	7.4.1	Intra DC coefficient	63
55	7.4.2	Other coefficients.....	63
56	7.4.2.1	Weighting matrices.....	63
57	7.4.2.2	Quantiser scale factor.....	64
58	7.4.2.3	Reconstruction formulae.....	65
59	7.4.3	Saturation	66
60	7.4.4	Mismatch control	66
61	7.4.5	Summary	66

1	7.5	Inverse DCT	67
2	7.5.1	Non-coded blocks and skipped macroblocks	67
3	7.6	Motion compensation	67
4	7.6.1	Prediction modes	68
5	7.6.2	Prediction field and frame selection	69
6	7.6.2.1	Field prediction	69
7	7.6.2.2	Frame prediction	71
8	7.6.3	Motion vectors	71
9	7.6.3.1	Decoding the motion vectors	72
10	7.6.3.2	Vector restrictions	73
11	7.6.3.3	Updating motion vector predictors	73
12	7.6.3.4	Resetting motion vector predictors	75
13	7.6.3.5	Prediction in P-pictures	75
14	7.6.3.6	Dual prime additional arithmetic	75
15	7.6.3.7	Vectors for colour difference components	77
16	7.6.3.8	Semantic restrictions concerning predictions	77
17	7.6.3.9	Concealment motion vectors	77
18	7.6.4	Forming predictions	78
19	7.6.5	Motion vector selection	79
20	7.6.6	Skipped Macroblocks	81
21	7.6.6.1	P field-picture	81
22	7.6.6.2	P frame-picture	82
23	7.6.6.3	B field-picture	82
24	7.6.6.4	B frame-picture	82
25	7.6.7	Combining predictions	82
26	7.6.7.1	Simple frame predictions	82
27	7.6.7.2	Simple field predictions	82
28	7.6.7.3	16x8 Motion compensation	82
29	7.6.7.4	Dual prime	82
30	7.6.8	Adding prediction and coefficient data	83
31	7.7	Spatial Scalability	84
32	7.7.1	Prediction in scalable layer	84
33	7.7.2	Formation of 'spatial' prediction	85
34	7.7.2.1	General	85
35	7.7.2.2	Deinterlacing	86
36	7.7.2.3	Vertical resampling	87
37	7.7.2.4	Horizontal resampling	87
38	7.7.2.5	Chroma formats	88
39	7.7.2.6	Generalised slice structure in the lower layer	88
40	7.7.3	Selection and combination of spatial and temporal predictions	88
41	7.7.4	Updating motion vector predictors and Motion vector selection	90
42	7.7.4.1	Resetting motion vector predictors	92
43	7.7.5	Skipped macroblocks	95
44	7.7.6	Skipped pictures in the lower layer	95
45	7.8	SNR scalability	96
46	7.8.1	Higher syntactic structures	97
47	7.8.2	Macroblock	99
48	7.8.2.1	dct_type	99
49	7.8.2.2	Skipped Macroblocks	99
50	7.8.3	Block	99
51	7.8.3.1	VLC decoding	99
52	7.8.3.2	Inverse scan	99
53	7.8.3.3	Inverse quantisation	100
54	7.8.3.4	Addition of coefficients from the two layers	100
55	7.8.3.5	Remaining macroblock decoding steps	100
56	7.9	Temporal scalability	101
57	7.10	Data Partitioning	103
58	7.11	Hybrid scalability	104
59	8	Profiles and levels	106
60	8.1	Simple profile	107
61	8.1.1	Simple profile syntax	107

1	8.1.1.1	Picture coding type	107
2	8.1.1.2	Chroma sampling structure	108
3	8.1.1.3	Scalability	108
4	8.1.1.4	Slice structure	108
5	8.1.2	Main level	108
6	8.1.2.1	Frame dimensions	108
7	8.1.2.2	Coded data rate and VBV buffer size	108
8	8.1.2.3	Vector range	108
9	8.1.2.4	intra_dc_precision	108
10	8.2	Main profile	108
11	8.2.1	Main profile syntax	109
12	8.2.1.1	Chroma sampling structure	109
13	8.2.1.2	Scalability	109
14	8.2.1.3	Slice structure	109
15	8.2.2	Low level	109
16	8.2.2.1	Frame dimensions	109
17	8.2.2.2	Coded data rate and VBV buffer size	109
18	8.2.2.3	Vector range	109
19	8.2.2.4	intra_dc_precision	109
20	8.2.3	Main level	109
21	8.2.3.1	Frame dimensions	109
22	8.2.3.2	Coded data rate and VBV buffer size	110
23	8.2.3.3	Vector range	110
24	8.2.3.4	intra_dc_precision	110
25	8.2.4	High-1440 level	110
26	8.2.4.1	Frame dimensions	110
27	8.2.4.2	Coded data rate and VBV buffer size	110
28	8.2.5	High level	110
29	8.2.5.1	Frame dimensions	110
30	8.2.5.2	Coded data rate and VBV buffer size	111
31	8.3	SNR Scalable Profile	111
32	8.3.1	SNR Scalable profile syntax	111
33	8.3.1.1	Chroma sampling structure	111
34	8.3.1.2	Slice structure	111
35	8.3.2	Low level	111
36	8.3.2.1	Frame dimensions	111
37	8.3.2.2	Coded data rate and VBV buffer size	111
38	8.3.2.3	Vector range	112
39	8.3.2.4	intra_dc_precision	112
40	8.3.3	Main level	112
41	8.3.3.1	Frame dimensions	112
42	8.3.3.2	Coded data rate and VBV buffer size	112
43	8.3.3.3	Vector range	112
44	8.3.3.4	intra_dc_precision	112
45	8.4	Spatially Scalable Profile	112
46	8.4.1	Spatially Scalable profile syntax	112
47	8.4.1.1	Chroma sampling structure	112
48	8.4.1.2	Slice structure	112
49	8.4.2	High-1440 level	113
50	8.4.2.1	Frame dimensions	113
51	8.4.2.2	Coded data rate and VBV buffer size	113
52	8.4.2.3	Vector range	113
53	8.5	High profile	113
54	8.5.1	High profile syntax	113
55	8.5.1.1	Chroma sampling structure	113
56	8.5.1.2	Slice structure	113
57	8.5.1.3	Scalability	114
58	8.5.2	Main level	114
59	8.5.2.1	Frame dimensions	114
60	8.5.2.2	Coded data rate and VBV buffer size	114
61	8.5.2.3	Vector range	114

1	8.5.2.4	intra_dc_precision	114
2	8.5.3	High-1440 level	114
3	8.5.3.1	Frame dimensions	114
4	8.5.3.2	Coded data rate and VBV buffer size.....	115
5	8.5.3.3	Vector range	115
6	8.5.4	High level	115
7	8.5.4.1	Frame dimensions	115
8	8.5.4.2	Coded data rate and VBV buffer size.....	115
9	8.5.4.3	Vector range	115
10		Annex A Discrete cosine transform	116
11		Annex B Variable length code tables.....	117
12	B.1	Macroblock addressing	117
13	B.2	Macroblock type	118
14	B.3	Macroblock pattern.....	123
15	B.4	Motion vectors	124
16	B.5	DCT coefficients.....	125
17		Annex C Video buffering verifier.....	134
18	C.1	Video buffering verifier	134
19		Annex D Features supported by the algorithm	138
20	D.1	Overview.....	138
21	D.2	Video Formats	138
22	D.2.1	Sampling Formats and Color.....	138
23	D.2.2	Movie Timing	138
24	D.2.3	Display Format Control.....	138
25	D.2.5	Transparent coding of composite video	139
26	D.3	Picture Quality	139
27	D.4	Data Rate Control	139
28	D.5	Low Delay Mode	139
29	D.6	Random Access/Channel Hopping	140
30	D.7	Scalability	140
31	D.7.1	Use of SNR scalability at a single spatial resolution.....	140
32	D.7.1.1	Additional features	140
33	D.7.1.1.1	Error resilience	140
34	D.7.1.1.2	Chroma simulcast.....	140
35	D.7.1.2	SNR scalable encoding process.....	141
36	D.7.1.2.1	Description	141
37	D.7.1.2.2	A few important remarks	141
38	D.7.2	Multiple resolution scalability bitstreams using SNR scalability	141
39	D.7.2.1	Decoder Implementation	142
40	D.7.2.2	Encoder Implementation	142
41	D.7.3	Bitrate allocation in data partitioning	142
42	D.7.4	Temporal scalability	142
43	D.7.4.1	Progressive:progressive-to-progressive Temporal Scalability	143
44	D.7.4.2	Progressive:interlace-to-interlace temporal scalability	143
45	D.7.4.3	Interlace:interlace-to-interlace Temporal Scalability	145
46	D.7.5	Hybrids of the spatial, the SNR and the temporal	145
47	D.7.5.1	Spatial and SNR hybrid scalability applications	145
48	D.7.5.2	Spatial and temporal hybrid scalability applications.....	145
49	D.7.5.3	Temporal and SNR hybrid scalability applications.....	146
50	D.8	Compatibility	146
51	D.8.1	Compatibility with higher and lower resolution formats	146
52	D.8.2	Compatibility with ISO/IEC IS 11172-2 (and ITU-T Rec. H.261)	146
53	D.9	Complexity.....	146
54	D.9.1	Restrictions to reduce decoder implementation cost	146
55	D.10	Editing Encoded Bit Streams	147
56	D.11	Trick modes	147
57	D.12	Error Resilience	148
58	D.12.1.	Concealment possibilities	149
59	D.12.1.1	Temporal predictive concealment	149
60	D.12.1.1.1	Substitution from previous frame.....	149
61	D.12.1.1.2	Motion compensated concealment	150

1	D.12.1.1.3	Use of Intra MVs	150
2	D.12.1.2	Spatial predictive concealment	150
3	D.12.1.3	Layered coding to facilitate concealment	151
4	D.12.1.3.1	Use of data partitioning	151
5	D.12.1.3.2	Use of SNR scalable coding	152
6	D.12.1.3.3	Use of spatial scalable coding.....	152
7	D.12.1.3.4	Use of temporal scalable coding.....	152
8	D.12.2	Spatial localisation	153
9	D.12.2.1	Small slices	153
10	D.12.2.2	Adaptive slice size	153
11	D.12.3	Temporal localisation.....	154
12	D.12.3.1	Intra pictures	154
13	D.12.3.2	Intra slices	154
14	D.12.4	Summary	154
15	Annex E	Profile and level restrictions.....	157
16	Annex F	Patent statements	173
17	Annex G	Bibliography	174

1 Foreword

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

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

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

18 In the field of information technology, ISO and IEC have established a joint technical committee,
19 ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated
20 to national bodies for voting. Publication as an International Standard requires approval by at least
21 75% of the national bodies casting a vote.

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

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

32 ISO/IEC

33 This International Standard is published in four Parts.

34 13818-1 systems — specifies the system coding of the specification. It defines a multiplexed
35 structure for combining audio and video data and means of representing the
36 timing information needed to replay synchronised sequences in real-time.

37 13818-2 video — specifies the coded representation of video data and the decoding process
38 required to reconstruct pictures.

39 13818-3 audio — specifies the coded representation of audio data.

40 13818-4 conformance— specifies the procedures for determining the characteristics of coded
41 bitstreams and for testing compliance with the requirements stated in 13818-
42 1, 13818-2 and 13818-3.

1 **I Introduction**

2 **I.1 Purpose**

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

9 **I.2 Application**

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

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

27 **I.3 Profiles and levels**

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

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

37 A "profile" is a defined sub-set of the entire bitstream syntax that is defined by this specification.
38 Within the bounds imposed by the syntax of a given profile it is still possible to require a very large
39 variation in the performance of encoders and decoders depending upon the values taken by parameters
40 in the bitstream. For instance it is possible to specify frame sizes as large as (approximately) 2^{14} pels
41 wide by 2^{14} lines high. It is currently neither practical nor economic to implement a decoder capable
42 of dealing with all possible frame sizes.

43 In order to deal with this problem "levels" are defined within each profile. A level is a defined set of
44 constraints imposed on parameters in the bitstream. These constraints may be simple limits on

1 numbers. Alternatively they may take the form of constraints on arithmetic combinations of the
2 parameters (e.g. frame width multiplied by frame height multiplied by frame rate).

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

8 **I.4 The scalable and the non-scalable syntax**

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

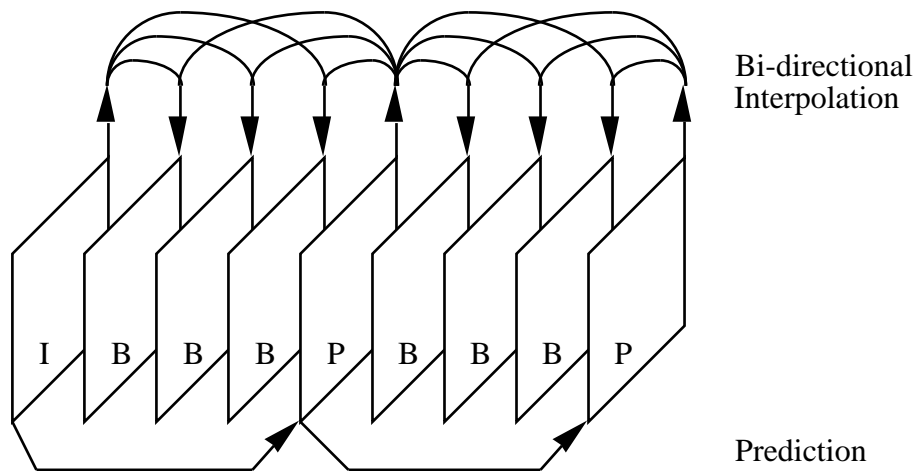
16 **I.4.1 Overview of the non-scalable syntax**

17 The coded representation defined in the non-scalable syntax achieves a high compression ratio while
18 preserving good image quality. The algorithm is not lossless as the exact pixel values are not preserved
19 during coding. The choice of the techniques is based on the need to balance a high image quality and
20 compression ratio with the requirement to make random access to the coded bitstream. Obtaining good
21 image quality at the bitrates of interest demands very high compression, which is not achievable with
22 intra picture coding alone. The need for random access, however, is best satisfied with pure intra
23 picture coding. This requires a careful balance between intra- and interframe coding and between
24 recursive and non-recursive temporal redundancy reduction.

25 A number of techniques are used to achieve high compression. The algorithm first uses block-based
26 motion compensation to reduce the temporal redundancy. Motion compensation is used both for causal
27 prediction of the current picture from a previous picture, and for non-causal, interpolative prediction
28 from past and future pictures. Motion vectors are defined for each 16-pixel by 16-line region of the
29 picture. The difference signal, i.e., the prediction error, is further compressed using the discrete cosine
30 transform (DCT) to remove spatial correlation before it is quantised in an irreversible process that
31 discards the less important information. Finally, the motion vectors are combined with the residual
32 DCT information, and encoded using variable length codes.

33 **I.4.1.1 Temporal processing**

34 Because of the conflicting requirements of random access and highly efficient compression, three main
35 picture types are defined. Intra coded pictures (I-Pictures) are coded without reference to other pictures.
36 They provide access points to the coded sequence where decoding can begin, but are coded with only
37 moderate compression. Predictive coded pictures (P-Pictures) are coded more efficiently using motion
38 compensated prediction from a past intra or predictive coded picture and are generally used as a
39 reference for further prediction. Bidirectionally-predictive coded pictures (B-Pictures) provide the
40 highest degree of compression but require both past and future reference pictures for motion
41 compensation. Bidirectionally-predictive coded pictures are never used as references for prediction.
42 The organisation of the three picture types in a sequence is very flexible. The choice is left to the
43 encoder and will depend on the requirements of the application. Figure 0-1 illustrates the relationship
44 among the three different picture types.



1
2 **Figure 0-1 Example of temporal picture structure**

3 **I.4.1.2 Coding interlaced video**

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

10 **I.4.1.3 Motion representation - macroblocks**

11 As in ISO/IEC 11172-2, the choice of 16 by 16 macroblocks for the motion-compensation unit is a
12 result of the trade-off between the coding gain provided by using motion information and the overhead
13 needed to store it. Each macroblock can be temporally predicted in one of a number of different ways.
14 For example, in frame encoding, the prediction from the previous reference frame can itself be either
15 frame-based or field-based. Depending on the type of the macroblock, motion vector information and
16 other side information is encoded with the compressed prediction error signal in each macroblock. The
17 motion vectors are encoded differentially with respect to the last encoded motion vectors using variable
18 length codes. The maximum length of the vectors that may be represented can be programmed, on a
19 picture-by-picture basis, so that the most demanding applications can be met without compromising the
20 performance of the system in more normal situations.

21 It is the responsibility of the encoder to calculate appropriate motion vectors. The specification does not
22 specify how this should be done.

23 **I.4.1.4 Spatial redundancy reduction**

24 Both original pictures and prediction error signals have high spatial redundancy. This specification uses
25 a block-based DCT method with visually weighted quantisation and run-length coding. After motion
26 compensated prediction or interpolation, the residual picture is split into 8 by 8 blocks. These are
27 transformed into the DCT domain where they are weighted before being quantised. After quantisation
28 many of the coefficients are zero in value and so two-dimensional run-length and variable length
29 coding is used to encode the remaining coefficients efficiently.

30 **I.4.1.5 Chroma formats**

31 In addition to the 4:2:0 format supported in ISO/IEC 11172-2 this specification supports 4:2:2 and
32 4:4:4 chroma formats.

33 **I.4.2 Scalable extensions**

34 The scalability tools in this specification are designed to support applications beyond that supported by
35 single layer video. Among the noteworthy applications areas addressed are video telecommunications,
36 video on asynchronous transfer mode networks (ATM), interworking of video standards, video service
37 hierarchies with multiple spatial, temporal and quality resolutions, HDTV with embedded TV, systems
38 allowing migration to higher temporal resolution HDTV etc. Although a simple solution to scalable
39 video is the simulcast technique which is based on transmission/storage of multiple independently

1 coded reproductions of video, a more efficient alternative is scalable video coding, in which the
 2 bandwidth allocated to a given reproduction of video can be partially reutilised in coding of the next
 3 reproduction of video. In scalable video coding, it is assumed that given an encoded bitstream,
 4 decoders of various complexities can decode and display appropriate reproductions of coded video. A
 5 scalable video encoder is likely to have increased complexity when compared to a single layer encoder.
 6 However, this standard provides several different forms of scalabilities that address nonoverlapping
 7 applications with corresponding complexities. The basic scalability tools offered are: *data partitioning*,
 8 *SNR scalability*, *spatial scalability* and *temporal scalability*. Moreover, combinations of these basic
 9 scalability tools are also supported and are referred to as *hybrid scalability*. In the case of basic
 10 scalability, two layers of video referred to as the *lower layer* and the *enhancement layer* are allowed,
 11 whereas in hybrid scalability up to 3 layers are supported. The following tables provide a few example
 12 applications of various scalabilities.

13 **Table 0-?. Applications of SNR scalability**

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

14

15 **Table 0-?. Applications of spatial scalability**

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

16

17 **Table 0-?. Applications of temporal scalability**

Base	Enhancement	Higher	Application
prog(30Hz)	prog(30Hz)	prog (60Hz)	Migration to HR prog HDTV
interl(30Hz)	interl(30Hz)	prog (60Hz)	Migration to HR prog HDTV

18

19 **I.4.2.1 Spatial scalable extension**

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

1 I.4.2.2 SNR scalable extension

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

14 I.4.2.3 Temporal scalable extension

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

29 I.4.2.4 Data partitioning extension

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

1 **INTERNATIONAL STANDARD 13818-2**
2 **ITU-T RECOMMENDATION H.262**
3 **INFORMATION TECHNOLOGY -**
4 **GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO**
5

6 **1 Scope**

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

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

17 **2 Normative references**

18 The following ITU-T Recommendations and International Standards contain provisions which through
19 reference in this text, constitute provisions of this Recommendation | International Standard. At the
20 time of publication, the editions indicated were valid. All Recommendations and Standards are subject
21 to revision, and parties to agreements based on this Recommendation | International Standard are
22 encouraged to investigate the possibility of applying the most recent editions of the standards indicated
23 below. Members of IEC and ISO maintain registers of currently valid International Standards. The TSB
24 (Telecommunication Standardisation Bureau) maintains a list of currently valid ITU-T
25 Recommendations.

- 26 • Recommendations and reports of the CCIR, 1990
27 XVIIth Plenary Assembly, Dusseldorf, 1990 Volume XI - Part 1
28 Broadcasting Service (Television) Rec. 601-2 "Encoding parameters of digital television for
29 studios"
- 30 • CCIR Volume X and XI Part 3 Recommendation 648: Recording of audio signals.
- 31 • CCIR Volume X and XI Part 3 Report 955-2: Sound broadcasting by satellite for portable
32 and mobile receivers, including Annex IV Summary description of advanced digital system II.
- 33 • ISO/IEC 11172 (1993) "Information technology — Coding of moving picture and
34 associated audio for digital storage media at up to about 1.5 Mbit/s"
- 35 • IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine
36 Transform, IEEE Std 1180-1990, December 6, 1990.
- 37 • IEC Publication 908:198, "CD Digital Audio System"
- 38 • IEC Standard Publication 461 Second edition, 1986 "Time and control code for video tape
39 recorders"
- 40 • ITU-T Recommendation H.261 (Formerly CCITT Recommendation H.261) "Codec for
41 audiovisual services at px64 kbit/s" Geneva, 1990
- 42 • ISO/IEC 10918-1 | ITU-T Rec. T.81 (JPEG) "Digital compression and coding of continuous-
43 tone still images"

44

1 **3 Definitions**

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

3 **3.1 AC coefficient:** Any DCT coefficient for which the frequency in one or both dimensions is
4 non-zero.

5 **3.2 backward compatibility:** A new coding standard is backward compatible with an existing
6 coding standard if existing decoders (designed to operate with the existing coding standard) are able to
7 continue to operate by decoding all or part of a bitstream produced according to the new coding
8 standard.

9 **3.3 backward motion vector:** A motion vector that is used for motion compensation from a
10 reference picture at a later time in display order.

11 **3.4 bidirectionally predictive-coded picture; B-picture:** A picture that is coded using motion
12 compensated prediction from past and/or future reference pictures.

13 **3.5 bitrate:** The rate at which the compressed bitstream is delivered from the storage medium to
14 the input of a decoder.

15 **3.6 block:** An 8-row by 8-column matrix of pels, or 64 DCT coefficients (source, quantised or
16 dequantised).

17 **3.7 bottom field:** One of two fields that comprise a frame of interlaced video. Each line of a
18 bottom field is spatially located immediately below the corresponding line of the top field.

19 **3.8 byte aligned:** A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits
20 from the first bit in the stream.

21 **3.9 byte:** Sequence of 8-bits.

22 **3.10 channel:** A digital medium that stores or transports a bitstream constructed according to this
23 specification.

24 **3.11 chroma format:** Defines the number of chrominance blocks in a macroblock.

25 **3.12 chroma simulcast:** A type of scalability (which is a subset of SNR scalability) where the
26 enhancement layer (s) contain only coded refinement data for the DC coefficients, and all the data for
27 the AC coefficients, of the chroma components.

28 **3.13 chrominance (component):** A matrix, block or single pel representing one of the two colour
29 difference signals related to the primary colours in the manner defined in the bitstream. The symbols
30 used for the colour difference signals are Cr and Cb.

31 **3.14 coded video bitstream:** A coded representation of a series of one or more pictures as defined
32 in this specification.

33 **3.15 coded order:** The order in which the pictures are stored and decoded. This order is not
34 necessarily the same as the display order.

35 **3.16 coded representation:** A data element as represented in its encoded form.

36 **3.17 coding parameters:** The set of user-definable parameters that characterise a coded video
37 bitstream. Bitstreams are characterised by coding parameters. Decoders are characterised by the
38 bitstreams that they are capable of decoding.

39 **3.18 component:** A matrix, block or single pel from one of the three matrices (luminance and two
40 chrominance) that make up a picture.

41 **3.19 compression:** Reduction in the number of bits used to represent an item of data.

42 **3.20 constant bitrate coded video:** A compressed video bitstream with a constant average bitrate.

43 **3.21 constant bitrate:** Operation where the bitrate is constant from start to finish of the
44 compressed bitstream.

45 **3.22 CRC:** Cyclic redundancy code.

46 **3.23 data element:** An item of data as represented before encoding and after decoding.

- 1 **3.24 data partitioning:** A method for dividing a bitstream into two separate bitstreams for error
2 resilience purposes. the two bitstreams have to be recombined before decoding.
- 3 **3.25 DC coefficient:** The DCT coefficient for which the frequency is zero in both dimensions.
- 4 **3.26 DCT coefficient:** The amplitude of a specific cosine basis function.
- 5 **3.27 decoder input buffer:** The first-in first-out (FIFO) buffer specified in the video buffering
6 verifier.
- 7 **3.28 decoder input rate:** The data rate specified in the video buffering verifier and encoded in the
8 coded video bitstream.
- 9 **3.29 decoder:** An embodiment of a decoding process.
- 10 **3.30 decoding (process):** The process defined in this specification that reads an input coded
11 bitstream and produces decoded pictures or audio samples.
- 12 **3.31 dequantisation:** The process of rescaling the quantised DCT coefficients after their
13 representation in the bitstream has been decoded and before they are presented to the inverse DCT.
- 14 **3.32 digital storage media; DSM:** A digital storage or transmission device or system.
- 15 **3.33 discrete cosine transform; DCT:** Either the forward discrete cosine transform or the inverse
16 discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse
17 DCT is defined in Annex A of this specification.
- 18 **3.34 display order:** The order in which the decoded pictures are displayed. Normally this is the
19 same order in which they were presented at the input of the encoder.
- 20 **3.35 editing:** The process by which one or more compressed bitstreams are manipulated to produce
21 a new compressed bitstream. Conforming edited bitstreams must meet the requirements defined in this
22 specification.
- 23 **3.36 encoder:** An embodiment of an encoding process.
- 24 **3.37 encoding (process):** A process, not specified in this specification, that reads a stream of input
25 pictures or audio samples and produces a valid coded bitstream as defined in this specification.
- 26 **3.38 fast forward playback:** The process of displaying a sequence, or parts of a sequence, of
27 pictures in display-order faster than real-time.
- 28 **3.39 fast reverse playback:** The process of displaying the picture sequence in the reverse of
29 display order faster than real-time..
- 30 **3.40 field:** For an interlaced video signal, a "field" is the assembly of alternate lines of a frame.
31 Therefore. an interlaced frame is composed of two fields a top field and a bottom field.
- 32 **3.41 field period:** The reciprocal of twice the frame rate.
- 33 **3.42 flag:** A variable which can take one of only the two values defined in this specification.
- 34 **3.43 forbidden:** The term "forbidden" when used in the clauses defining the coded bitstream
35 indicates that the value shall never be used. This is usually to avoid emulation of start codes.
- 36 **3.44 forced updating:** The process by which macroblocks are intra-coded from time-to-time to
37 ensure that mismatch errors between the inverse DCT processes in encoders and decoders cannot build
38 up excessively.
- 39 **3.45 forward compatibility:** A new coding standard is forward compatible with an existing coding
40 standard if new decoders (designed to operate with the new coding standard) continue to be able to
41 decode bitstreams of the existing coding standard.
- 42 **3.46 forward motion vector:** A motion vector that is used for motion compensation from a
43 reference picture at an earlier time in display order.
- 44 **3.47 frame:** A frame contains lines of spatial information of a video signal. For progressive video,
45 these lines contain samples starting from one time instant and continuing through successive lines to
46 the bottom of the frame. For interlaced video a frame consists of two fields, a top field and a bottom
47 field. One of these fields will commence one field period later than the other.
- 48 **3.48 frame period:** The reciprocal of the frame rate.

- 1 **3.49 frame rate:** The rate at which frames are be output from the decoding process.
- 2 **3.50 future reference picture:** A future reference picture is a reference picture that occurs at a
3 later time than the current picture in display order.
- 4 **3.51 header:** A block of data in the coded bitstream containing the coded representation of a
5 number of data elements pertaining to the coded data that follow the header in the bitstream.
- 6 **3.52 hybrid scalability:** Hybrid scalability is the combination of two (or more) types of
7 scalability.
- 8 **3.53 interlace:** The property of conventional television frames where alternating lines of the frame
9 represent different instances in time.
- 10 **3.54 intra coding:** Coding of a macroblock or picture that uses information only from that
11 macroblock or picture.
- 12 **3.55 intra-coded picture; I-picture:** A picture coded using information only from itself.
- 13 **3.56 level :** A defined set of constraints on the values which may be taken by the parameters of this
14 specification within a particular profile. A profile may contain one or more levels.
- 15 **3.57 luminance (component):** A matrix, block or single pel representing a monochrome
16 representation of the signal and related to the primary colours in the manner defined in the bitstream.
17 The symbol used for luminance is Y.
- 18 **3.58 macroblock:** The four 8 by 8 blocks of luminance data and the two (for 4:2:0 chroma
19 format), four (for 4:2:2 chroma format) or eight (for 4:4:4 chroma format) corresponding 8 by 8
20 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture.
21 Macroblock is sometimes used to refer to the pel data and sometimes to the coded representation of the
22 pel values and other data elements defined in the macroblock header of the syntax defined in this part
23 of this specification. The usage is clear from the context.
- 24 **3.59 motion compensation:** The use of motion vectors to improve the efficiency of the prediction
25 of pel values. The prediction uses motion vectors to provide offsets into the past and/or future
26 reference pictures containing previously decoded pel values that are used to form the prediction error
27 signal.
- 28 **3.60 motion estimation:** The process of estimating motion vectors during the encoding process.
- 29 **3.61 motion vector:** A two-dimensional vector used for motion compensation that provides an
30 offset from the coordinate position in the current picture to the coordinates in a reference picture.
- 31 **3.62 non-intra coding:** Coding of a macroblock or picture that uses information both from itself
32 and from macroblocks and pictures occurring at other times.
- 33 **3.63 parameter:** A variable within the syntax of this specification which may take one of a large
34 range of values. A variable which can take one of only two values is a flag and not a parameter.
- 35 **3.64 past reference picture:** A past reference picture is a reference picture that occurs at an earlier
36 time than the current picture in display order.
- 37 **3.65 pel aspect ratio:** The ratio of the nominal vertical height of pel on the display to its nominal
38 horizontal width.
- 39 **3.66 pel:** Picture element.
- 40 **3.67 picture:** Source, coded or reconstructed image data. A source or reconstructed picture consists
41 of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance
42 signals. For progressive video, a picture is identical to a frame, while for interlaced video, a picture can
43 refer to a frame, or the top field or the bottom field of the frame depending on the context.
- 44 **3.68 prediction:** The use of a predictor to provide an estimate of the pel value or data element
45 currently being decoded.
- 46 **3.69 predictive-coded picture; P-picture:** A picture that is coded using motion compensated
47 prediction from past reference pictures.
- 48 **3.70 prediction error:** The difference between the actual value of a pel or data element and its
49 predictor.

- 1 **3.71 predictor:** A linear combination of previously decoded pel values or data elements.
- 2 **3.72 profile:** A defined sub-set of the syntax of this specification.
- 3 3.73 Note In this specification the word “profile” is used as defined above. It should
4 not be confused with other definitions of “profile” and in particular it does not have the
5 meaning that is defined by JTC1/SGFS.
- 6 **3.74 quantisation matrix:** A set of sixty-four 8-bit values used by the dequantiser.
- 7 **3.75 quantised DCT coefficients:** DCT coefficients before dequantisation. A variable length
8 coded representation of quantised DCT coefficients is stored as part of the compressed video bitstream.
- 9 **3.76 quantiser scale:** A scale factor coded in the bitstream and used by the decoding process to
10 scale the dequantisation.
- 11 **3.77 random access:** The process of beginning to read and decode the coded bitstream at an
12 arbitrary point.
- 13 **3.78 reference picture:** Reference pictures are the nearest adjacent I- or P-pictures to the current
14 picture in display order.
- 15 **3.79 reserved:** The term "reserved" when used in the clauses defining the coded bitstream indicates
16 that the value may be used in the future for ISO/IEC defined extensions.
- 17 **3.80 scalability:** Scalability is the ability of a decoder to decode an ordered set of bitstreams to
18 produce a reconstructed sequence. Moreover, useful video is output when subsets are decoded. The
19 minimum subset that can thus be decoded is the first bitstream in the set which is called the base layer.
20 Each of the other bitstreams in the set is called an enhancement layer. When addressing a specific
21 enhancement layer, "lower layer" refer to the bitstream which precedes the enhancement layer.
- 22 **3.81 side information:** Information in the bitstream necessary for controlling the decoder.
- 23 **3.82 skipped macroblock:** A macroblock for which no data is encoded.
- 24 **3.83 slice:** A series of macroblocks.
- 25 **3.84 SNR scalability:** A type of scalability where the enhancement layer (s) contain only coded
26 refinement data for the DCT coefficients. of the base layer.
- 27 **3.85 spatial scalability:** A type of scalability where an enhancement layer also uses predictions
28 from pel data derived from a lower layer without using motion vectors. The layers can have different
29 frame sizes, frame rates or chroma formats
- 30 **3.86 start codes [system and video]:** 32-bit codes embedded in that coded bitstream that are
31 unique. They are used for several purposes including identifying some of the structures in the coding
32 syntax.
- 33 **3.87 stuffing (bits); stuffing (bytes) :** Code-words that may be inserted into the compressed
34 bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the
35 stream.
- 36 **3.88 temporal scalability:** A type of scalability where an enhancement layer also uses predictions
37 from pel data derived from a lower layer using motion vectors. The layers have identical frame size,
38 and chroma formats, but can have different frame rates.
- 39 **3.89 top field:** One of two fields that comprise a frame of interlaced video. Each line of a top field
40 is spatially located immediately above the corresponding line of the bottom field.
- 41 **3.90 variable bitrate:** Operation where the bitrate varies with time during the decoding of a
42 compressed bitstream.
- 43 **3.91 variable length coding; VLC:** A reversible procedure for coding that assigns shorter code-
44 words to frequent events and longer code-words to less frequent events.
- 45 **3.92 video buffering verifier; VBV:** A hypothetical decoder that is conceptually connected to the
46 output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an
47 encoder or editing process may produce.
- 48 **3.93 video sequence:** A series of one or more pictures.

- 1 **3.94 zig-zag scanning order:** A specific sequential ordering of the DCT coefficients from
2 (approximately) the lowest spatial frequency to the highest.

1 **4 Abbreviations and symbols**

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

5 **4.1 Arithmetic operators**

6 + Addition.

7 - Subtraction (as a binary operator) or negation (as a unary operator).

8 ++ Increment.

9 -- Decrement.

10 * Multiplication.

11 ^ Power.

12 / Integer division with truncation of the result toward zero. For example, 7/4 and -7/-4 are
13 truncated to 1 and -7/4 and 7/-4 are truncated to -1.

14 // Integer division with rounding to the nearest integer. Half-integer values are rounded
15 away from zero unless otherwise specified. For example 3//2 is rounded to 2, and -3//2 is
16 rounded to -2.

17 *DIV* Integer division with truncation of the result toward minus infinity.

18 ÷ Used to denote division in mathematical equations where no truncation or rounding is
19 intended.

20 % Modulus operator. Defined only for positive numbers.

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

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

23 *NINT()* Nearest integer operator. Returns the nearest integer value to the real-valued argument.
24 Half-integer values are rounded away from zero.

25 *sin* Sine.

26 *cos* Cosine.

27 *exp* Exponential.

28 $\sqrt{\quad}$ Square root.

29 *log10* Logarithm to base ten.

30 *loge* Logarithm to base e.

31 **4.2 Logical operators**

32 || Logical OR.

33 && Logical AND.

34 ! Logical NOT.

35 **4.3 Relational operators**

36 > Greater than.

37 >= Greater than or equal to.

1	<	Less than.
2	<=	Less than or equal to.
3	==	Equal to.
4	!=	Not equal to.
5	<i>max</i> [...,]	the maximum value in the argument list.
6	<i>min</i> [...,]	the minimum value in the argument list.
7	4.4	Bitwise operators
8	&	AND
9		OR
10	>>	Shift right with sign extension.
11	<<	Shift left with zero fill.
12	4.5	Assignment
13	=	Assignment operator.
14	4.6	Mnemonics
15		The following mnemonics are defined to describe the different data types used in the coded bit-stream.
16	bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in the
17		specification. Bit strings are written as a string of 1s and 0s within single quote marks,
18		e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no
19		significance.
20	uimsbf	Unsigned integer, most significant bit first.
21	simsbf	Signed integer, in twos complement format, most significant (sign) bit first.
22	vlclbf	Variable length code, left bit first, where "left" refers to the order in which the VLC codes
23		are written. The byte order of multibyte words is most significant byte first.
24	4.7	Constants
25	π	3.14159265359...
26	<i>e</i>	2.71828182845...

1 5 Conventions

2 5.1 Method of describing bitstream syntax

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

6 The action caused by a decoded data element in a bitstream depends on the value of that data element
7 and on data elements previously decoded. The decoding of the data elements and definition of the state
8 variables used in their decoding are described in Clause 6.3. The following constructs are used to
9 express the conditions when data elements are present, and are in normal type:

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

12

<pre>while (condition) { data_element ... }</pre>	<p>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.</p>
<pre>do { data_element ... } while (condition)</pre>	<p>The data element always occurs at least once.</p> <p>The data element is repeated until the condition is not true.</p>
<pre>if (condition) { data_element ... } else { data_element ... }</pre>	<p>If the condition is true, then the first group of data elements occurs next in the data stream.</p> <p>If the condition is not true, then the second group of data elements occurs next in the data stream.</p>
<pre>for (i = 0; i < n; i++) { data_element ... }</pre>	<p>The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.</p>

13

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

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

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

19 While the syntax is expressed in procedural terms, it should not be assumed that Clause 6.3 implements
20 a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream.
21 Actual decoders must include means to look for start codes in order to begin decoding correctly, and to
22 identify errors, erasures or insertions while decoding. The methods to identify these situations, and the
23 actions to be taken, are not standardised.

1 5.2 Definition of functions

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

3 5.2.1 Definition of bytealigned() function

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

6 5.2.2 Definition of nextbits() function

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

9 5.2.3 Definition of next_start_code() function

10 The next_start_code() function removes any zero bit and zero byte stuffing and locates the next start
11 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"
}		

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

15 5.3 Reserved, forbidden and marker_bit

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

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

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

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

23 5.4 Arithmetic precision

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

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

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

1 **6 Video bitstream syntax and semantics**

2 **6.1 Structure of video data**

3 In general the highest syntactic structure of the coded video bitstream is the video sequence. Note
4 however that when a scalable extension is used two or more separate bitstreams are decoded in a single
5 decoder. Each of these bitstreams complies with the description in this clause.

6 See International Standard 13818-1 for a description of the way these separate video bitstreams may be
7 multiplexed together.

8 See clauses 7.7 to 7.10 of this specification for a description of the decoding process for scalable
9 extensions.

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

13 This structure is very similar to that use in ISO/IEC 11172-2 though it should be noted that the “Group
14 of Pictures Layer” in that international standard has become an optional syntactic element within the
15 video_sequence() within this specification. It is thus no longer possible to view a “Group of Pictures”
16 as a subordinate structure within the video sequence.

17 **6.1.1 Video sequence**

18 A coded video sequence commences with a sequence header which may optionally be followed by a
19 group of pictures header and then by one or more pictures. The video sequence is terminated by a
20 sequence_end_code Within each sequence, pictures shall be decoded continuously. At various points
21 in the video sequence a particular picture may be preceded by either a repeat sequence header or a
22 group of pictures header or both. (In the case that both a repeat sequence header and a group of
23 pictures header immediately precede a particular picture the group of pictures header shall follow the
24 repeat sequence header.)

25 A video sequence received at the input of a decoder may be different from the one at the output of the
26 encoder due to editing.

27 **6.1.1.1 Frame reordering**

28 The order of the pictures in the coded bitstream is the order in which the decoder processes them. The
29 reconstructed frames are not necessarily in the correct order for display.

30 The following is an example of pictures taken from the beginning of a video sequence. In this example
31 there are two B-pictures between successive P-pictures and also two B-pictures between successive I-
32 and P-pictures and all pictures are frame-pictures. Picture '1I' is used to form a prediction for picture
33 '4P'. Pictures '4P' and '1I' are both used to form predictions for pictures '2B' and '3B'. Therefore the
34 order of pictures in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder shall display
35 them in the order '1I', '2B', '3B', '4P'.

36 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

37 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

38 At the decoder output,

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

39

40 The number of consecutive B-Pictures is variable. B-pictures may not be present between successive
41 P-pictures (or between I-pictures and P-pictures).

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

1 Lastly a sequence may contain no P-pictures.

2 A sequence shall not be composed of only B-pictures.

3 **6.1.1.2 Sequence header**

4 A video sequence header commences with a `sequence_header_code` and is followed by a series of data
5 elements. In this specification `sequence_header()` shall be followed by `sequence_extension()` which
6 includes further parameters beyond those used by ISO/IEC 11172-2.

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

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

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

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

24 Repeating the sequence header allows the data elements of the initial sequence header to be repeated in
25 order that random access into the video sequence is possible.

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

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

35 **6.1.1.3 Group of pictures header**

36 A group of pictures header is intended to assist random access into the sequence. In the coded
37 bitstream, the first coded picture following a group of pictures header shall be I-picture. In display
38 order, the last picture before a group of pictures headers shall be an I-Picture or a P-Picture, and the
39 first shall be either an I-Picture or the first B-Picture of the consecutive series of B-Pictures which
40 immediately precedes the first I-Picture.

41 Any number of pictures may occur between successive group of pictures header Applications requiring
42 random access, fast-forward playback, or fast reverse playback may use relatively short groups of
43 pictures. Groups of pictures headers may also be used at scene cuts or other cases where motion
44 compensation is ineffective.

45 **6.1.2 Picture**

46 A source or reconstructed picture consists of three rectangular matrices of integers; a luminance matrix
47 (Y), and two chrominance matrices (Cb and Cr).

48 The relationship between these Y, Cb and Cr components and the primary (analogue) Red, Green and
49 Blue Signals (E'_R , E'_G and E'_B), the chromaticity of these primaries and the transfer characteristics of
50 the source picture are specified in the bitstream. This information does not affect the decoding process.

- 1 The code assignment shall be;
- 2 • For the 8 bit luminance signal, the nominal black level corresponds to level 16 and the
 - 3 nominal peak white level corresponds to level 235. However, the signal may extend above and
 - 4 below these values in the range 1 to 254.
 - 5 • For the 8 bit chrominance signal the nominal signal range is from level 16 to level 240 with
 - 6 zero colour difference corresponding to level 128. However, the signal may extend above and
 - 7 below these values in the range 1 to 254.

8

9 Note The decoding process given by this specification does not limit the output pel values.

10 Thus the reserved code words 0 and 255 will occasionally occur at the output of the

11 decoding process.

12 **6.1.2.1 4:2:0 Format**

13 In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and

14 vertical dimensions. The Y-matrix shall have an even number of rows and columns.

15 Note — When interlaced pictures are coded as field pictures each of these field pictures shall have a Y-

16 matrix with half the number of rows as the corresponding frame picture. Thus the total number of rows

17 in the Y-matrix of an entire frame shall be divisible by four.

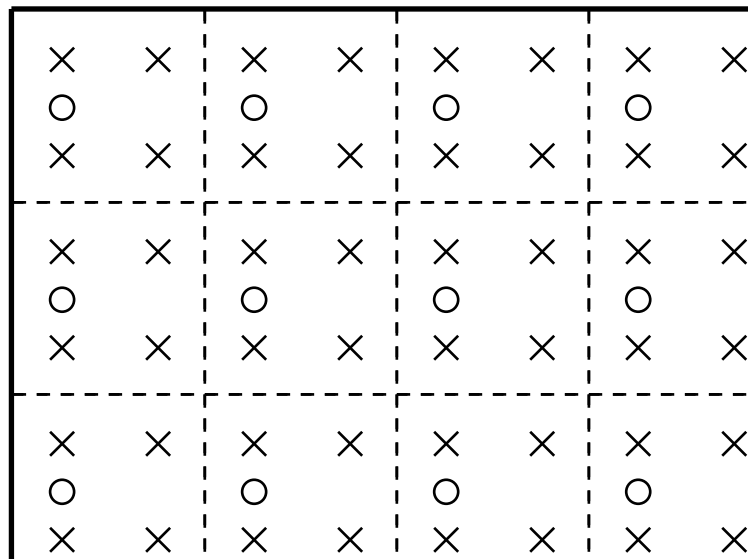
18 The luminance and chrominance samples are positioned as shown in Figure 6-1.

19 In order to clarify the organisation, Figure 6-3 shows the (vertical) positioning of the samples when the

20 frame is separated into two fields. In each field the chrominance samples do not lie mid way between

21 the luminance samples, this is so that the spatial location of the chrominance samples is the same

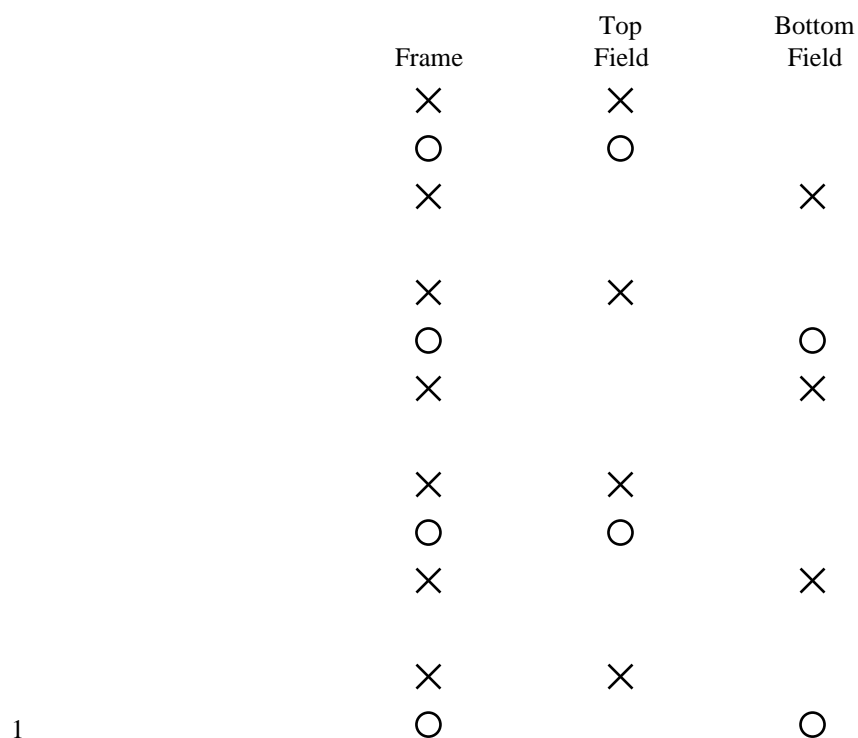
22 whether the frame is represented as a single frame-picture or two field-pictures.



- 23
- 24 × Represent luminance pels
- 25 ○ Represent chrominance pels

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

27



2 **Figure 6-3 – Vertical positions of samples in the frame and the two fields.**

3

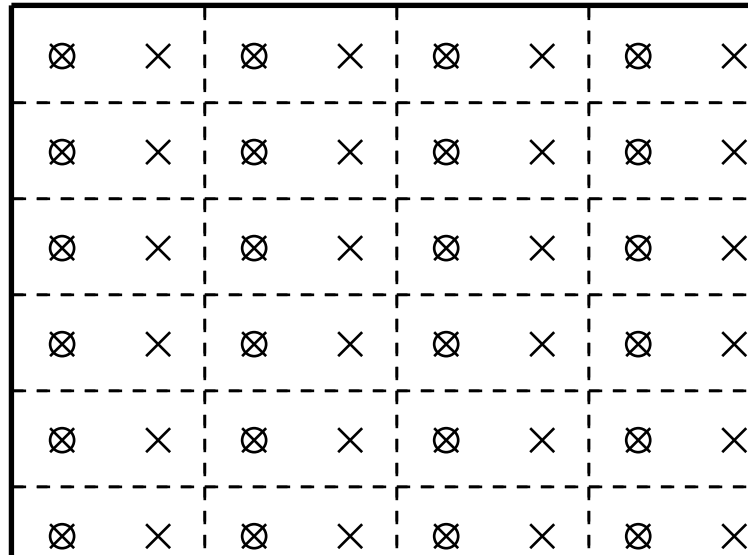
4 **6.1.2.2 4:2:2 Format**

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

8 Note — When interlaced frames are coded as field pictures each of these field pictures shall have a Y-
9 matrix with half the number of rows as the corresponding frame picture. Thus the total number of rows
10 in the Y-matrix of an entire frame shall be divisible by two.

11 The luminance and chrominance samples are positioned as shown in Figure 6-4.

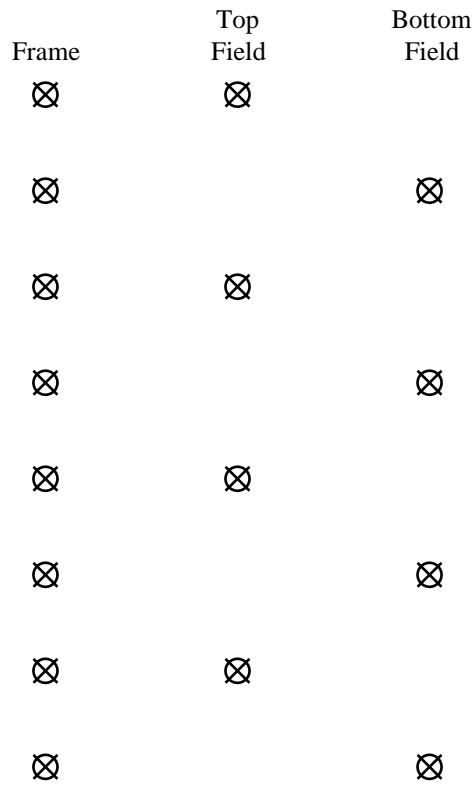
12 In order to clarify the organisation, Figure 6-5 shows the (vertical) positioning of the samples when the
13 frame is separated into two fields.



1
2
3
4
5

⊗ Represent luminance pels
○ Represent chrominance pels

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



6
7
8

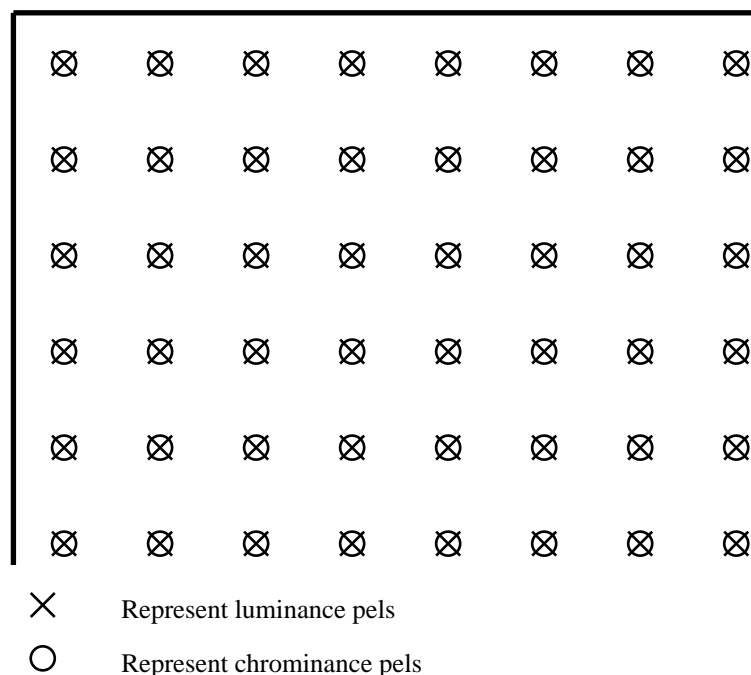
Figure 6-5 – Vertical positions of samples with 4:2:2 and 4:4:4 data

1 6.1.2.3 4:4:4 Format

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

4 Note — When interlaced frames are coded as field pictures each of these field pictures shall have a Y-
5 matrix with half the number of rows as the corresponding frame picture. Thus the total number of rows
6 in the Y-matrix of an entire frame shall be divisible by two.

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



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

12 6.1.2.4 Picture Types

13 There are three types of pictures that use different coding methods.

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

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

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

19 6.1.2.5 Progressive and interlaced sequences

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

21 The output of the decoding process, for interlaced sequences, consists of a series of fields that are
22 separated in time by a field period. The two fields of a frame may be coded independently of one
23 another (field-pictures). Alternatively the two fields may be coded together as a frame (frame-
24 pictures). It is possible to switch between the use of field pictures and frame pictures dynamically.

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

28 6.1.2.5.1 Field pictures

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

1 When the first field of the frame is coded as a P-picture the second field picture of the frame shall also
 2 be coded as a P-picture. Similarly when the first field of the frame is coded as a B-picture the second
 3 field of the frame shall also be coded as a B-picture.

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

6 **6.1.2.5.2 Frame pictures**

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

9 **6.1.3 Slice**

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

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

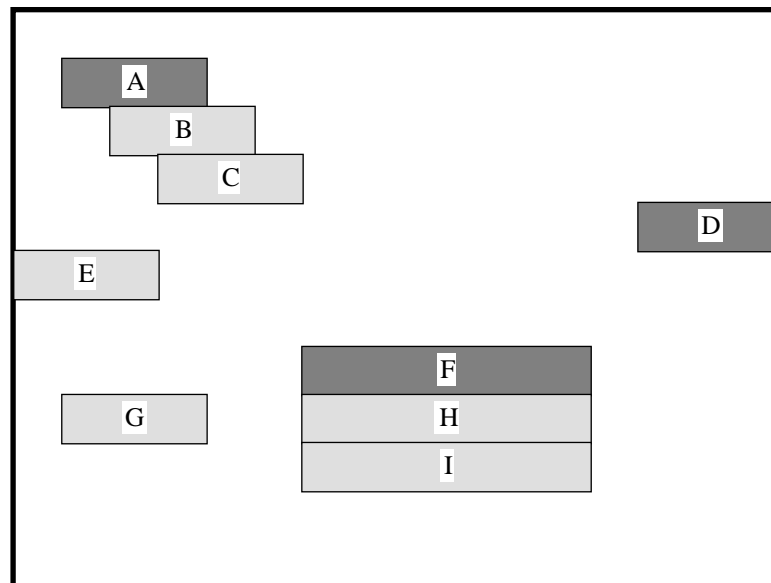
14 Slices shall occur in the bitstream in the order in which they are encountered, starting at the upper-left
 15 of the picture and proceeding by raster-scan order from left to right and top to bottom (illustrated in the
 16 figures of this clause as alphabetical order).

17 **6.1.3.1 The general slice structure**

18 In the most general case it is not necessary for the slices to cover the entire picture. Figure 6-7 shows
 19 this case. Those areas that are not enclosed in a slice are not encoded and no information is encoded
 20 for such areas (in the specific picture).

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

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



25

26

Figure 6-7. The most general slice structure.

27 **6.1.3.2 Restricted slice structure**

28 In certain defined levels of defined profiles a restricted slice structure illustrated in Figure 6-8 shall be
 29 used. In this case every macroblock in the picture shall be enclosed in a slice.

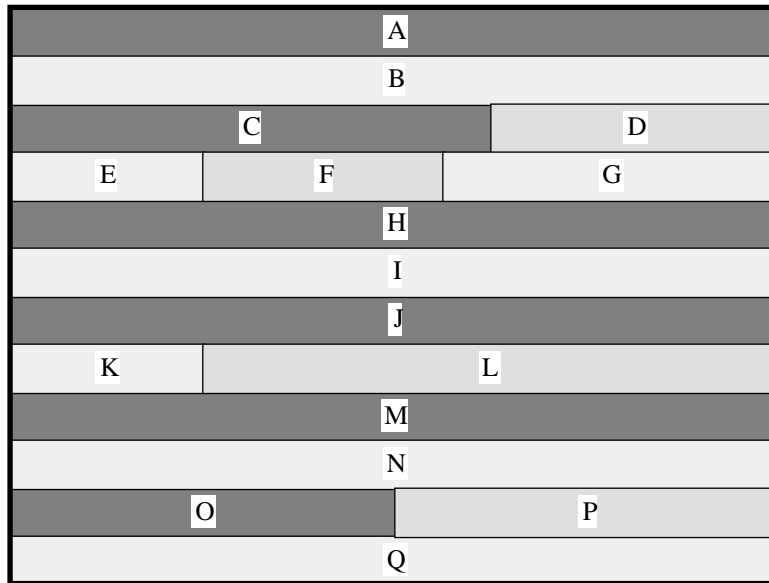


Figure 6-8. Restricted slice structure.

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

6.1.4 Macroblock

A **macroblock** contains a section of the luminance component and the spatially corresponding chrominance components. 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 stored (see Clause 6.6.). There are three chroma 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 chroma 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 6-9.

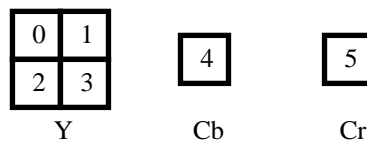


Figure 6-9 4:2:0 Macroblock structure

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 6-10.

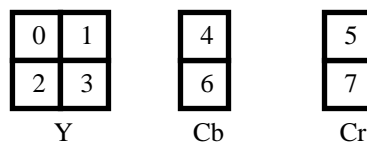
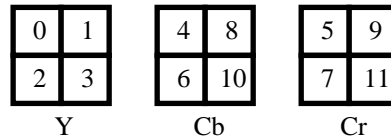


Figure 6-10 4:2:2 Macroblock structure

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 6-11.

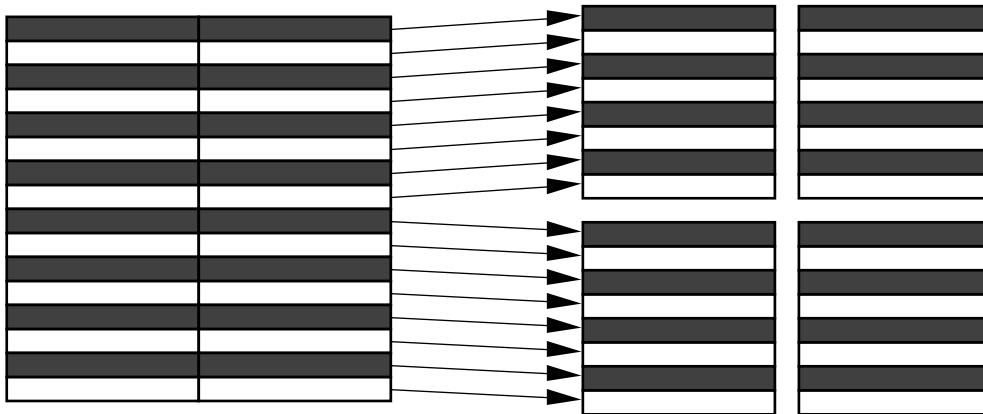


1
2

Figure 6-12 4:4:4 Macroblock structure

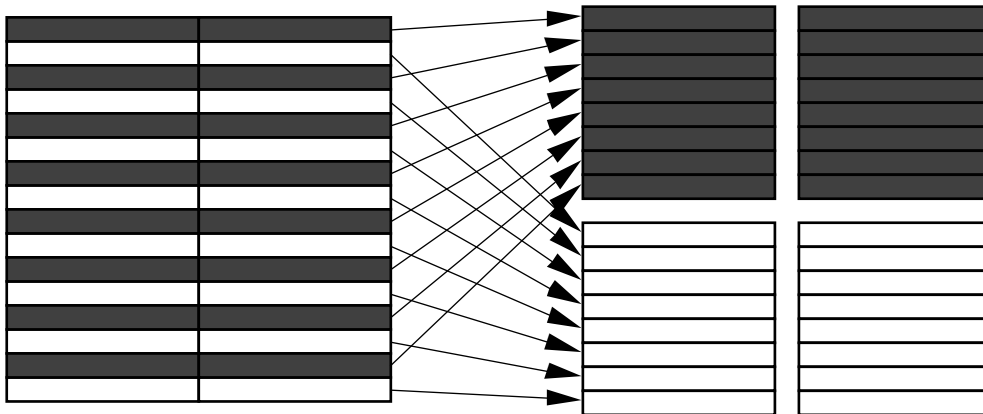
3 In frame pictures, where both frame and field DCT coding may be used, the internal organisation
4 within the macroblock is different. This is depicted for the luminance blocks in Figure 6-12 and 6-13.

5 In the 4:2:0 format the chrominance blocks shall always be organised in frame structure for the
6 purposes of DCT coding. This avoids the need for an 8x4 IDCT. It should however be noted that field
7 based predictions may be made for these blocks which will, in the general case, require that predictions
8 for 8x4 regions (after half-pel filtering) must be made.



9
10

Figure 6-12 Luminance macroblock structure in frame DCT coding



11
12

Figure 6-13 Luminance macroblock structure in field DCT coding

13 In field pictures, there is only one internal organisation of the macroblock, which is shown if Figure 6-
14 12.

15 **6.1.5 Block**

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

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

21 When "block" refers to the DCT coefficients there will be 64 coefficients.

1 **6.2 Video bitstream syntax**

2 **6.2.1 Start codes**

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

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

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

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

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

13 **Table 6-1 — Start code values**

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

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

1 **6.2.2 Video Sequence**

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

2

1 **6.2.2.1 Sequence header**

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

2

3 **6.2.2.2 Extension and user data**

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

4

1 **6.2.2.2.1 Extension data**

	No. of bits	Mnemonic
extension_data(i) {		
while (nextbits()== extension_start_code) {		
extension_start_code	32	bslbf
if (i == 0) { /* follows sequence_extension() */		
if (nextbits()== "Sequence Display Extension ID")		
sequence_display_extension()		
if (nextbits()== "Sequence Scalable Extension ID")		
sequence_scalable_extension()		
}		
if (i == 1) { /* follows group_of_pictures_header() */		
}		
if (i == 2) { /* follows picture_coding_extension() */		
if (nextbits()== "Quant Matrix Extension ID")		
quant_matrix_extension()		
if (nextbits()== "Picture Pan Scan Extension ID")		
picture_display_extension()		
if (nextbits()== "Picture Spatial Scalable Extension ID")		
picture_spatial_scalable_extension()		
if (nextbits()== "Picture Temporal Scalable Ext. ID")		
picture_temporal_scalable_extension()		
}		
}		
}		

2

3 **6.2.2.2.2 User data**

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

4

1 **6.2.2.3 Sequence extension**

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

2

3 **6.2.2.4 Sequence display extension**

<code>sequence_display_extension() {</code>	No. of bits	Mnemonic
<code> extension_start_code_identifier</code>	4	uimsbf
<code> video_format</code>	3	uimsbf
<code> colour_description</code>	1	uimsbf
<code> if (colour_description) {</code>		
<code> colour_primaries</code>	8	uimsbf
<code> transfer_characteristics</code>	8	uimsbf
<code> matrix_coefficients</code>	8	uimsbf
<code> }</code>		
<code> display_horizontal_size</code>	14	uimsbf
<code> marker_bit</code>	1	“1”
<code> display_vertical_size</code>	14	uimsbf
<code> next_start_code()</code>		
<code>}</code>		

4

1 **6.2.2.5 Sequence scalable extension**

	No. of bits	Mnemonic
sequence_scalable_extension() {		
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	"1"
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()		
}		

2

3 **6.2.2.6 Group of pictures header**

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

4

1 **6.2.3 Picture header**

<code>picture_header() {</code>	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
<code>if (picture_coding_type == 2 picture_coding_type == 3) {</code>		
full_pel_forward_vector	1	
forward_f_code	3	uimsbf
<code>}</code>		
<code>if (picture_coding_type == 3) {</code>		
full_pel_backward_vector	1	
backward_f_code	3	uimsbf
<code>}</code>		
<code>while (nextbits() == '1') {</code>		
extra_bit_picture	1	"1"
extra_information_picture	8	
<code>}</code>		
extra_bit_picture	1	"0"
<code>next_start_code()</code>		
<code>}</code>		

2

1 **6.2.3.1 Picture coding extension**

picture_coding_extension() {	No . of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
forward_horizontal_f_code	4	uimsbf
forward_vertical_f_code	4	uimsbf
backward_horizontal_f_code	4	uimsbf
backward_vertical_f_code	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	
 burst_amplitude	7	uimsbf
 sub_carrier_phase	8	uimsbf
 }		
next_start_code()		
}		

2

1 **6.2.3.2 Quant matrix extension**

	No. of bits	Mnemonic
quant_matrix_extension() {		
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()		
}		

2

3 **6.2.3.3 Picture display extension**

	No. of bits	Mnemonic
picture_display_extension() {		
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	"1"
frame_centre_vertical_offset	16	simsbf
marker_bit	1	"1"
}		
next_start_code()		
}		

4

5 **6.2.3.4 Picture temporal scalable extension**

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

6

1 **6.2.3.5 Picture spatial scalable extension**

	No. of bits	Mnemonic
picture_spatial_scalable_extension() {		
extension_start_code_identifier	4	uimsbf
lower_layer_temporal_reference	10	uimsbf
marker_bit	1	"1"
lower_layer_horizontal_offset	15	simsbf
marker_bit	1	"1"
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()		
}		

2

3 **6.2.3.6 Picture data**

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

4

1 **6.2.4 Slice**

	No. of bits	Mnemonic
slice() {		
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') {		
marker_bit	1	"1"
intra_slice	1	uimsbf
reserved_bits	7	uimsbf
while (nextbits() == '1') {		
extra_bit_slice	1	"1"
extra_information_slice	8	
}		
}		
extra_bit_slice	1	"0"
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

2

1 **6.2.5 Macroblock**

	No. of bits	Mnemonic
macroblock() {		
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	vlclbf
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	“1”
if (macroblock_pattern)		
coded_block_pattern()
for (i=0; i<block_count; i++) {		
block(i)
}		
}		

2

3 **6.2.5.1 Macroblock modes**

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

4

1 **6.2.5.2 Motion vectors**

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

2

3 **6.2.5.2.1 Motion vector**

	No. of bits	Mnemonic
motion_vector (s) {		
motion_horizontal_code	1-11	vlc1bf
if ((horizontal_f != 1) && (motion_horizontal_code != 0))		
motion_horizontal_r	1-8	uimsbf
if (dmv == 1)		
dmv_horizontal	1-2	vlc1bf
motion_vertical_code	1-11	vlc1bf
if ((vertical_f != 1) && (motion_vertical_code != 0))		
motion_vertical_r	1-8	uimsbf
if (dmv == 1)		
dmv_vertical	1-2	vlc1bf
}		

4

5 **6.2.5.3 Coded block pattern**

	No. of bits	Mnemonic
coded_block_pattern () {		
coded_block_pattern_420	3-9	vlc1bf
if (chroma_format == 4:2:2)		
coded_block_pattern_1	2	uimsbf
if (chroma_format == 4:4:4)		
coded_block_pattern_2	6	uimsbf
}		

1 **6.2.6 Block layer**

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

4 Note The following does not adequately document the block layer syntax when data
5 partitioning is used. See clause 7.10.

6

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

7

1 **6.3 Video bitstream semantics**

2 **6.3.1 Semantic rules for higher syntactic structures**

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

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

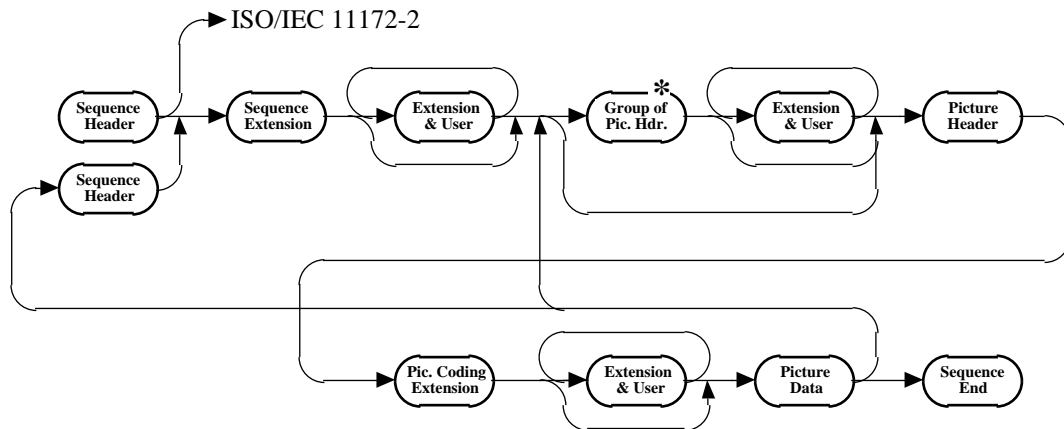
7 The following semantic rules apply:

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

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

22 At each point where extensions are allowed in the bitstream any number of the extensions from the
 23 defined allowable set may be included. However each type of extension shall occur only once.

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



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

28 **Figure 6-14. High level bitstream organisation**

29

1

Table 6-2. extension_start_code_identifier codes.

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

2

3 If a decoder encounters an extension whose extension identification code is described, above, as
 4 reserved it shall discard all following data until the next start code. This behaviour enables backward
 5 compatible extensions to be defined in the future.

6 **6.3.2 Video sequence**

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

9 **6.3.3 Sequence header**

10 **sequence_header_code** -- The sequence_header_code is the bit string 0000 01B3 in hexadecimal. It
 11 identifies the beginning of a sequence header.

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

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

14 **horizontal_size** -- The horizontal_size is a 14 bit unsigned integer, the 12 least significant bits are
 15 defined in horizontal_size_value, the 2 most significant bits are defined in horizontal_size_extension.
 16 The horizontal_size_value is the width of the displayable part of each luminance component in pixels.
 17 The width of the encoded luminance component in macroblocks, mb_width, is
 18 $(\text{horizontal_size} + 15)/16$. The displayable part of the frame is left-aligned in the encoded pictures.

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

21 **vertical_size** -- The vertical_size is a 14 bit unsigned integer, the 12 least significant bits are defined in
 22 vertical_size_value, the 2 most significant bits are defined in vertical_size_extension. The
 23 vertical_size_value is the height of the displayable part of each luminance component of the frame in
 24 pixels.

25 In the case that progressive_sequence is "1" the height of the encoded luminance component of the
 26 frame in macroblocks, mb_height, is $(\text{vertical_size} + 15)/16$.

27 In the case that progressive_sequence is "0" the height of the encoded luminance component of the
 28 frame in macroblocks, mb_height, is $2 * ((\text{vertical_size} + 31)/32)$.

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

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

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

1 aspect_ratio_information either specifies that the “pel aspect ratio” (PAR) of the reconstructed frame is
2 1.0 (square pels) or alternatively it gives the “display aspect ratio” (DAR).

3 If aspect_ratio_information specified that the pel aspect ratio is 1.0:

- 4 • Image aspect ratio (IAR) can be deduced from the horizontal_size and the vertical_size;

$$IAR = \frac{vertical_size}{horizontal_size}$$

- 6 • If the sequence_display_extension() is present, then DAR can be deduced from the
7 display_vertical_size and the display_horizontal_size;

$$DAR = \frac{display_vertical_size}{display_horizontal_size}$$

- 9 • If the sequence_display_extension() is not present, then DAR is undefined.

10 If aspect_ratio_information specifies the DAR:

- 11 • If sequence_display_extension() is not present then it is intended that the entire reconstructed
12 frame is intended to be mapped to the entire active region of the display. Thus:

$$PAR = DAR \times \frac{horizontal_size}{vertical_size}$$

- 14 • If sequence_display_extension() is present then IAR and DAR may differ and IAR can
15 derived as follows:

$$IAR = DAR \times \frac{display_horizontal_size}{display_vertical_size} \times \frac{vertical_size}{horizontal_size}$$

17 Pel aspect ratio can be derived as follows:

$$PAR = DAR \times \frac{display_horizontal_size}{display_vertical_size}$$

20 **Table 6-3--- aspect_ratio_information**

aspect_ratio_information	Pel Aspect Ratio	DAR
0000	forbidden	forbidden
0001	1.0 (Square Pel)	-
0010	-	3÷4
0011	-	9÷16
0100	-	reserved
...		...
1111	-	reserved

21
22 **frame_rate_code** -- This is a four-bit integer used to define frame_rate_value as shown in table 6-4.
23 frame_rate may be derived from frame_rate_value, frame_rate_extension_n and
24 frame_rate_extension_d as follows;

$$25 \quad frame_rate = frame_rate_value * (frame_rate_extension_n + 1) \div (frame_rate_extension_d + 1)$$

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

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

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

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

8

9

Table 6-4 --- frame_rate_value

frame_rate_code	frame_rate_value
0000	forbidden
0001	24000÷1001 (23.976)
0010	24
0011	25
0100	30000÷1001 (29.97)
0101	30
0110	50
0111	60000÷1001 (59.94)
1000	60
...	reserved
1111	reserved

10

11 **bit_rate_value** -- The lower 18 bits of bit_rate.

12 **bit_rate** -- This is a 30 bit integer. The lower 18 bits of the integer are in bit_rate and the upper 12 bits
13 are in bit_rate_extension. The 30 bit integer specifies the bit rate of the bitstream measured in units of
14 400 bits/second, rounded upwards. The value zero is forbidden.

15 If the bit-stream is a constant bit-rate stream, the bit rate specified is the actual rate of operation of the
16 VBV specified in annex C. If the bit-stream is a variable bit-rate stream, the STD specifications in
17 ISO/IEC 13818-1 supersede the VBV, and the bit-rate specified here is used to dimension the transport
18 stream STD (2.4.2 in ISO/IEC 13818-1).

19 The bit-stream is a variable rate bit-stream if and only if the vbv_delay field has the value 0xFFFF.

20 Given the value encoded in the bit-rate field, the bit-stream shall be generated so that the video
21 encoding and the worst case multiplex jitter do not cause STD buffer overflow or underflow.

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

23 **vbv_buffer_size_value** -- the lower 10 bits of vbv_buffer_size.

24 **vbv_buffer_size** -- vbv_buffer_size is an 18-bit integer. The lower 10 bits of the integer are in
25 vbv_buffer_size_value and the upper 8 bits are in vbv_buffer_size_extension. The integer defines the
26 size of the VBV (Video Buffering Verifier, see Annex C) buffer needed to decode the sequence. It is
27 defined as:

$$28 \quad B = 16 * 1024 * vbv_buffer_size$$

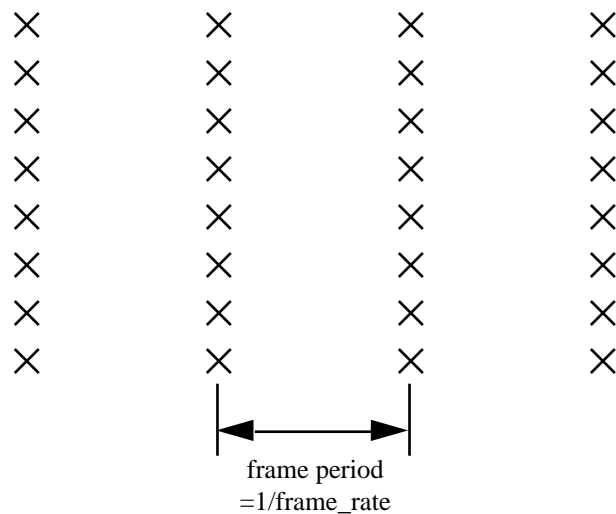
29 where B is the minimum VBV buffer size in bits required to decode the sequence (see Annex C).

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

32 **load_intra_quantiser_matrix** -- See clause 6.3.6 "Quant matrix extension"

33 **intra_quantiser_matrix** -- See clause 6.3.6 "Quant matrix extension"

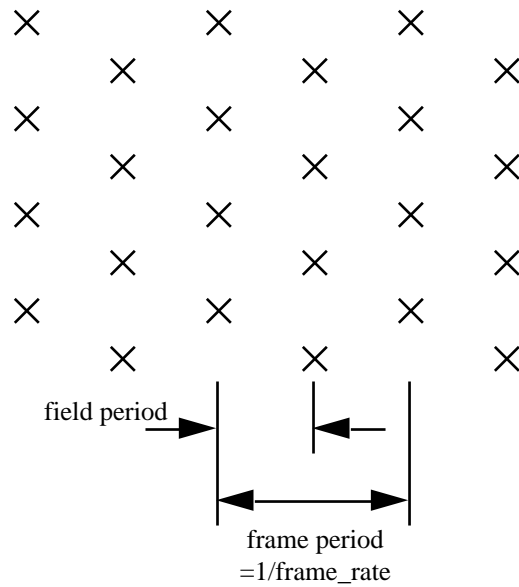
- 1 **load_non_intra_quantiser_matrix** -- See clause 6.3.6 “Quant matrix extension”
- 2 **non_intra_quantiser_matrix** -- See clause 6.3.6 “Quant matrix extension”
- 3 **6.3.4 Extension and user data**
- 4 **extension_start_code** -- The extension_start_code is the bit string 000001B5 in hexadecimal. It
- 5 identifies the beginning of extensions defined by this specification.
- 6 **user_data_start_code** -- The user_data_start_code is the bit string 000001B2 in hexadecimal. It
- 7 identifies the beginning of user data. The user data continues until receipt of another start code.
- 8 **user_data** -- The user_data is defined by the users for their specific applications. The user data shall
- 9 not contain a string of 23 or more zero bits.
- 10 **6.3.5 Sequence extension**
- 11 **profile_and_level_indication** – This is 8 bit integer used to signal the profile and level identification.
- 12 The meaning of the bits is given in clause 8.
- 13 Note in a scalable hierarchy the bitstreams of each layer may set profile_and_level_indication
- 14 to a different value as specified in clause 8.
- 15 **progressive_sequence** – When set to “1” the coded video sequence contains only frame-pictures which
- 16 have progressive_frame set to “1”. In this case the output of the decoding process are progressive
- 17 frames. Figure 6-15 illustrates this. Since the output of the decoding process is not interlaced,
- 18 top_field_first and repeat_first_field are not applicable.



19
20

Figure 6-15. progressive_sequence == 1

- 21 This sampling structure may be particularly applicable to applications that require automatic frame-rate
- 22 conversion.
- 23 When set to “0” the coded video sequence may contain both frame-pictures and field-pictures. In
- 24 frame-pictures progressive_frame may be zero or one (and different in different pictures). In this case
- 25 the output of the decoding process are interlaced fields. Figure 6-16 illustrates this. It is a requirement
- 26 on the bitstream that the fields at the output of the decoding process shall always be alternately top and
- 27 bottom (note that the very first field may be either top or bottom).

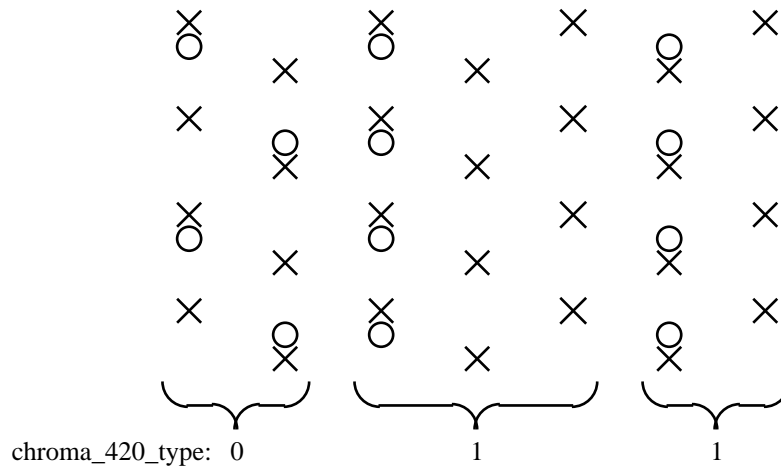


1
2

Figure 6-16. progressive_sequence == 0

3 When a progressive frame is decoded, the output of the decoding process is two or three consecutive
4 fields. The information that these fields originate from the same progressive frame in the bitstream can
5 be conveyed to the display process.

6 In the case of 4:2:0 chroma the chrominance data may be intended for upsampling independently
7 within the fields or, alternatively, within the frame. This is indicated by chroma_420_type. If
8 chroma_420_type is set to 1 then it is desirable to consider that all of the chrominance samples for the
9 frame are transferred from the decoding process to the display process at the same time as the first field
10 of luminance samples. This is illustrated in Figure 6-16a.



11
12

Figure 6-16a. progressive_sequence == 0 with 4:2:0 chrominance.

13 **chroma_format** - This is a two bit integer indicating the chrominance format as defined in the Table 6-
14 5.

1

Table 6-5. Meaning of chroma_format

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

2

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

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

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

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

7 **low_delay** - This flag, when set to "1", indicates that the sequence does not contain any B-pictures, that
8 the frame reordering delay is not present in the VBV description and that skipped pictures (VBV buffer
9 underflow) may occur.

10 When set to "0", it indicates that the sequence may contain B-pictures, that the frame reordering delay
11 is present in the VBV description and that skipped pictures (VBV buffer underflow) shall not occur.

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

14 **frame_rate_extension_n** -- See frame_rate_code.

15 **frame_rate_extension_d** -- See frame_rate_code.

16 6.3.6 Sequence display extension

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

19

Table 6-6. Meaning of video_format

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

20

- 1 **colour_description** -- A flag which if set to "1" indicates the presence of colour_primaries,
 2 transfer_characteristics and matrix_coefficients in the bitstream.
- 3 **colour_primaries** -- This 8-bit integer describes the chromaticity coordinates of the source primaries,
 4 and is defined in Table 6-7.

5 **Table 6-7. Colour Primaries**

Value	Primaries
0	(forbidden)
1	ITU-R Recommendation 709 (1990) primary x y green 0.300 0.600 blue 0.150 0.060 red 0.640 0.330 white D65 0.3127 0.3290
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	ITU-R Recommendation 624-4 System M primary x y green 0.21 0.71 blue 0.14 0.08 red 0.67 0.33 white C 0.310 0.316
5	ITU-R Recommendation 624-4 System B, G primary x y green 0.29 0.60 blue 0.15 0.06 red 0.64 0.33 white D65 0.313 0.329
6	SMPTE 170M primary x y green 0.310 0.595 blue 0.155 0.070 red 0.630 0.340 white D65 0.3127 0.3290
7	SMPTE 240M (1987) primary x y green 0.310 0.595 blue 0.155 0.070 red 0.630 0.340 white D65 0.3127 0.3291
8-255	reserved

- 6 In the case that sequence_display_extension() is not present in the bitstream or colour_description is
 7 zero the chromaticity is assumed to be that corresponding to colour_primaries having the value 1.
- 8

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

3

Table 6-8. Transfer Characteristics

Value	Transfer Characteristic
0	(forbidden)
1	ITU-R Recommendation 709 (1990) $V = 1.099 L_c^{0.45} - 0.099$ for $L_c \geq 0.018$ $V = 4.500 L_c$ for $0.018 > L_c \geq 0$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	ITU-R Recommendation 624-4 System M Assumed display gamma 2.2
5	ITU-R Recommendation 624-4 System B, G Assumed display gamma 2.8
6	SMPTE 170M $V = 1.099 L_c^{0.45} - 0.099$ for $L_c \geq 0.018$ $V = 4.500 L_c$ for $0.018 > L_c \geq 0$
7	SMPTE 240M (1987) $V = 1.1115 L_c^{0.45} - 0.1115$ for $L_c \geq 0.0228$ $V = 4.0 L_c$ for $0.0228 > L_c$
8-255	reserved

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

7

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

3

Table 6-9. Matrix Coefficients

Value	Matrix
0	(forbidden)
1	ITU-R Recommendation 709 (1990). $E'_Y = 0.7154 E'_G + 0.0721 E'_B + 0.2125 E'_R$ $E'_{PB} = -0.386 E'_G + 0.500 E'_B + -0.115 E'_R$ $E'_{PR} = -0.454 E'_G - 0.046 E'_B + 0.500 E'_R$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	FCC $E'_Y = 0.59 E'_G + 0.11 E'_B + 0.30 E'_R$ $E'_{PB} = -0.331 E'_G + 0.500 E'_B + -0.169 E'_R$ $E'_{PR} = -0.421 E'_G - 0.079 E'_B + 0.500 E'_R$
5	ITU-R Recommendation 624-4 System B, G $E'_Y = 0.587 E'_G + 0.114 E'_B + 0.299 E'_R$ $E'_{PB} = -0.331 E'_G + 0.500 E'_B + -0.169 E'_R$ $E'_{PR} = -0.419 E'_G - 0.081 E'_B + 0.500 E'_R$
6	SMPTE 170M $E'_Y = 0.587 E'_G + 0.114 E'_B + 0.299 E'_R$ $E'_{PB} = -0.331 E'_G + 0.500 E'_B + -0.169 E'_R$ $E'_{PR} = -0.419 E'_G - 0.081 E'_B + 0.500 E'_R$
7	SMPTE 240M (1987) $E'_Y = 0.701 E'_G + 0.087 E'_B + 0.212 E'_R$ $E'_{PB} = -0.384 E'_G + 0.500 E'_B - 0.116 E'_R$ $E'_{PR} = -0.445 E'_G - 0.055 E'_B + 0.500 E'_R$
8-255	reserved

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

7 **display_horizontal_size** - See `display_vertical_size`.

8 **display_vertical_size** - `display_horizontal_size` and `display_vertical_size` together define a rectangle
9 which may be considered as the "intended display's" active region. If this rectangle is smaller than the
10 encoded frame size then the display process may be expected to display only a portion of the encoded
11 frame. Conversely if the display rectangle is larger than the encoded frame size then the display
12 process may be expected to display the reconstructed frames on a portion of the display device rather
13 than on the whole display device.

14 `display_horizontal_size` shall be in the same units as `horizontal_size` (pels of the encoded frames).

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

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

1 **6.3.7 Quant matrix extension**

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

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

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

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

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

10

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

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

12

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

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

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

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

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

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

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

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

11 6.3.8 Sequence scalable extension

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

16 **lower_layer_prediction_horizontal_size** -- this a 14 bit integer indicating the horizontal size of the
 17 lower layer frame which is used for prediction.

18 **lower_layer_prediction_vertical_size** -- this a 14 bit integer indicating the vertical size of the lower
 19 layer frame which is used for prediction.

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

26 **Table 6-10. Definition of scalable_mode**

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

27

28 **layer_id** - This is an integer which identifies the layers in a scalable hierarchy. The base layer always
 29 has layer_id = 0. However, except in the case of data partitioning, sequence_scalable_extension(), and
 30 hence level_id, is not present in the bitstream for the base layer, and is assumed to be zero. Each
 31 successive layer has a layer_id which is one greater than the layer for which it is an enhancement.

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

34 **horizontal_subsampling_factor_m** — This affects the spatial scalable upsampling process, as defined
 35 in clause 7.7. The value shall be non-zero.

36 **horizontal_subsampling_factor_n** — This affects the spatial scalable upsampling process, as defined
 37 in clause 7.7. The value shall be non-zero.

38 **vertical_subsampling_factor_m** -- This affects the spatial scalable upsampling process, as defined in
 39 clause 7.7. The value shall be non-zero.

40 **vertical_subsampling_factor_n** — This affects the spatial scalable upsampling process, as defined in
 41 clause 7.7. The value shall be non-zero.

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

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

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

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

14 6.3.9 Group of pictures header

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

17 **time_code** -- This is a 25-bit field containing the following: drop_frame_flag, time_code_hours,
18 time_code_minutes, marker_bit, time_code_seconds and time_code_pictures as shown in Table 6-11.
19 The fields correspond to the fields defined in the IEC standard for "time and control codes for video
20 tape recorders" (see Bibliography, Annex E). The code refers to the first picture after the group of
21 pictures header that has a temporal_reference of zero. The drop_frame_flag can be set to either "0" or
22 "1". It may be set to "1" only if the frame rate is 29.97Hz. If it is "0" then pictures are counted
23 assuming rounding to the nearest integral number of pictures per second, for example 29.97Hz would
24 be rounded to and counted as 30Hz. If it is "1" then picture numbers 0 and 1 at the start of each minute,
25 except minutes 0, 10, 20, 30, 40, 50 are omitted from the count.

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

27 **Table 6-11 --- time_code**

time_code	range of value	No. of bits	Mnemonic
drop_frame_flag		1	
time_code_hours	0 - 23	5	uimsbf
time_code_minutes	0 - 59	6	uimsbf
marker_bit	1	1	"1"
time_code_seconds	0 - 59	6	uimsbf
time_code_pictures	0 - 59	6	uimsbf

28 **closed_gop** -- This is a one-bit flag which is set to "1" if the B-pictures (if any) immediately following
29 the group of picture header have been encoded without prediction vectors pointing to previous
30 pictures.

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

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

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

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

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

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

8 **extra_information_picture** -- Reserved. A decoder conforming to this specification that encounters
9 extra_information_picture in a bitstream shall ignore it (i.e. parse from bitstream and discard).

10 **6.3.11 Picture Coding Extension**

11 **forward_horizontal_f_code** -- An unsigned integer taking values 1 through 9 (or 15 if unused). The
12 value zero is forbidden. In an I-picture in which concealment_motion_vectors is zero (in which the
13 value is not used in the decoding process) forward_horizontal_f_code shall take the value 15 (all ones).

14 **forward_vertical_f_code** -- An unsigned integer taking values 1 through 9 (or 15 if unused). The
15 value zero is forbidden. In an I-picture in which concealment_motion_vectors is zero (in which the
16 value is not used in the decoding process) forward_vertical_f_code shall take the value 15 (all ones).

17 **backward_horizontal_f_code** -- An unsigned integer taking values 1 through 9 (or 15 if unused). The
18 value zero is forbidden. In an I-picture or a P-picture (in which the value is not used in the decoding
19 process) backward_horizontal_f_code shall take the value 15 (all ones).

20 **backward_vertical_f_code** -- An unsigned integer taking values 1 through 9 (or 15 if unused). The
21 value zero is forbidden. In an I-picture or a P-picture (in which the value is not used in the decoding
22 process) backward_vertical_f_code shall take the value 15 (all ones).

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

24

Table 6-13 Intra DC precision

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

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

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

28

Table 6-14 Meaning of picture_structure

picture_structure	Meaning
00	reserved
01	Top Field
10	Bottom Field
11	Frame-Picture

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

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

- 1 When a frame is encoded in the form of two field pictures the following syntax elements may be set
2 independently in each field picture:
- 3 • forward_horizontal_f_code, forward_vertical_f_code
 - 4 • backward_horizontal_f_code, backward_vertical_f_code
 - 5 • intra_dc_precision, concealment_motion_vectors, q_scale_type
 - 6 • intra_vlc_format, alternate_scan
- 7
- 8 **top_field_first** — The meaning of this element depends upon picture_structure. In a frame picture
9 top_field_first being set to “1” indicates that the top field of the frame is the earlier field (“0” if the
10 bottom field is earlier). In a field picture (or when progressive_sequence is set to “1”) top_field_first
11 shall have the value “0”.
- 12 **frame_pred_frame_dct** - This flag shall be set to “1” to indicate that only frame-DCT shall be used
13 and only frame prediction. In a field picture it shall be “0”. frame_pred_frame_dct shall be “1” if
14 progressive_frame is “1”.
- 15 **concealment_motion_vectors** - This flag has the value “1” to indicate that motion vectors are coded
16 for intra macroblocks.
- 17 **q_scale_type** - This affects the quantisation process as described in clause 7.4.2.2.
- 18 **intra_vlc_format** -- This affects the decoding of transform coefficient data as described in
19 clause 7.2.2.1.
- 20 **alternate_scan** -- This affects the decoding of transform coefficient data as described in clause 7.3.
- 21 **repeat_first_field** -- This flag is applicable only in a frame picture, in a field picture it shall be set to
22 zero and does not affect the decoding process.
- 23 If this flag is set to 0, the output of the decoding process, corresponding to this frame, consists of two
24 fields. The first field (top or bottom field as identified by top_field_first) is followed by the other field.
- 25 If it is set to 1, the output of the decoding process, corresponding to this frame, consists of three fields.
26 The first field (top or bottom field as identified by top_field_first) is followed by the other field, then
27 the first field is repeated.
- 28 It shall be zero if progressive_frame is equal to 0.
- 29 **chroma_420_type** - This is a 1-bit integer that indicates the way in which chroma samples in 4:2:0
30 picture were derived. This allows the display process to upsample the chroma in an appropriate
31 manner. (Note that the display process is not standardised in this specification.)
- 32 If it is zero it indicates that each field may be upsampled independently. If it is one then the chroma
33 data from the two fields should be combined into a single frame before upsampling.
- 34 The following semantic rules shall be obeyed in all bitstreams: In a field picture chroma_420_type
35 shall be zero. With 4:2:2 and 4:4:4 the value of chroma_420_type has no meaning and it shall be set to
36 the value 0.
- 37 **progressive_frame** — If progressive_frame is set to 0 it indicates that the two fields of the frame are
38 conventional interlaced fields in which an interval of time of the field period exists between
39 (corresponding spatial samples) of the two fields. In this case the following restriction applies:
- 40 • repeat_first_field shall be zero (two field duration).
- 41 If progressive_frame is set to 1 it indicates that the two fields (of the frame) are actually from the same
42 time instant as one another. In this case a number of restrictions to other parameters and flags in the
43 bitstream apply:
- 44 • picture_structure shall be “Frame”
 - 45 • frame_pred_frame_dct shall be 1
 - 46 • if chroma_format is “4:2:0”, then chroma_420_type shall be 1 (frame based).

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

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

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

10 **v_axis** -- 1 bit integer used in PAL. v-axis is set to 1 on a positive sign, v-axis is set to 0 otherwise.

11 **field_sequence** -- 3 bits integer which is defined in the table 6-15.

12 **Table 6-15 Definition of field_sequence.**

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

13

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

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

19 **sub_carrier_phase** -- This is a 8 bits integer defining the sub-carrier phase (for PAL and NTSC only).
20 Phase of reference sub-carrier at the field-synchronisation datum with respect, to field start as defined
21 in ITU-R Recommendation 470. See Table 6-16.

22 **Table 6-16 Definition of sub_carrier_phase.**

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

23

24 6.3.12 Picture display extension

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

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

1 **frame_centre_vertical_offset** — This is a 16 bit, signed, integer giving the vertical offset in units of
 2 1/16th pel. A positive value shall indicate that the centre of the reconstructed frame lies below the
 3 centre of the display rectangle.

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

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

```

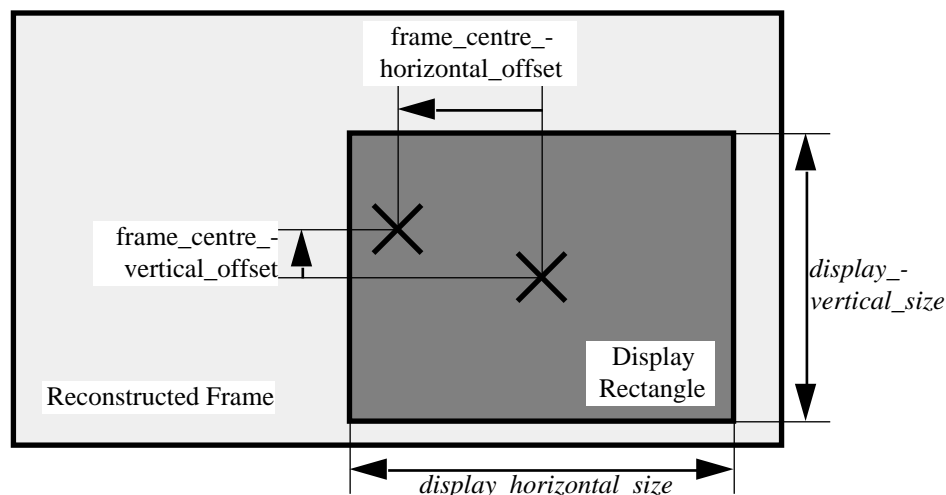
  9     if (( progressive_sequence == 1) || (picture_structure == "field")) {
 10         number_of_frame_centre_offsets = 1
 11     } else {
 12         if (repeat_first_field == "1")
 13             number_of_frame_centre_offsets = 3
 14         else
 15             number_of_frame_centre_offsets = 2
 16     }
 17
  
```

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

20 In the case that a given picture does not have a `picture_display_extension()` then the most recently
 21 decoded frame centre offset shall be used. Note that if more than one frame centre offset would have
 22 been present in the `picture_display_extension()` had it been present then all of these offsets take the
 23 same value. Following a `sequence_header()` the value zero shall be used for all frame centre offsets
 24 until a `picture_display_extension()` defines non-zero values.

25 Figure 6-17 illustrates the picture display parameters. As shown the frame centre offsets contained in
 26 the `picture_display_extension()` shall specify the position of the centre of the reconstructed frame from
 27 the centre of the display rectangle.

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



29

30

Figure 6-17. Frame centre offset parameters

31 The frame centre offsets are specified to an accuracy of 1/16th pel.

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

34 6.3.12.1 Pan-scan

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

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

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

7 **6.3.13 Picture spatial scalable extension**

8 **lower_layer_temporal_reference** - An unsigned integer value which indicates the temporal reference
 9 of the lower layer picture to be used to provide the prediction. If the lower level indicates the temporal
 10 reference with more than 10 bits, the least significant bits are indicated here. If the lower layer indicates
 11 the temporal reference with less than 10 bits, all bits are indicated here, with the extra most significant
 12 bits being set to zero.

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

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

21 **spatial_temporal_weight_code_table_index** -- this indicates which table of spatial temporal weight
 22 codes is to be used as defined in clause 7.7. `spatial_temporal_weight_code_table_index` must be "00"
 23 when `picture_structure` is "Top Field" or "Bottom Field" and when `progressive_frame` is 0.

24 **lower_layer_progressive_frame** -- this flag is set to 0 if the lower layer picture is an interlaced frame
 25 and to "1" if the lower layer picture is a progressive frame. The use of this flag in the spatial scalable
 26 upsampling process is defined in clause 7.7.

27 **lower_layer_deinterlaced_field_select** -- This affects the spatial scalable upsampling process, as
 28 defined in clause 7.7.

29 **6.3.14 Picture temporal scalable extension**

30 **reference_select_code** -- This is a 2 bit code that identifies reference picture(s) for prediction
 31 depending on the picture type.

32 **forward_temporal_reference** -- It indicates the temporal reference of the lower layer picture used for
 33 forward prediction.

34 **backward_temporal_reference** -- It indicates the temporal reference of the lower layer picture used
 35 for backward prediction. When `backward_temporal_reference` is not used it shall carry a value of 0.

36 **6.3.15 Slice header**

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

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

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

44 The macroblock row may be calculated as follows:

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

50 The `slice_vertical_position` of the first row of macroblocks is one. Some slices may have the same
 51 `slice_vertical_position`, since slices may start and finish anywhere. The maximum value of

1 slice_vertical_position is 175 unless slice_vertical_position_extension is present in which case
2 slice_vertical_position shall be in the range [128:1].

3 **priority_breakpoint** — This is a 7 bit integer that indicates the point in the syntax where the bitstream
4 is supposed to be partitioned. The allowed values and their semantic interpretation is given in Table 7-
5 26 Note that when MSB of priority_breakpoint is set, the remaining 6 bits indicate the number of
6 (run, level) DCT codewords present in the base partition, unless the scan is ended earlier by an EOB.
7 The value of priority_breakpoint can only take the value zero in partition 1.

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

11 **intra_slice** - This shall be set to "0" if any of the macroblocks in the slice are non-intra macroblocks.
12 If all of the macroblocks are intra macroblocks then intra_slice may be set to "1". intra_slice may be
13 omitted from the bitstream in which case it shall be assumed to have the value zero.

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

16 **reserved_bits** - These seven bits shall have the value zero.

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

19 **extra_information_slice** -- Reserved. A decoder conforming to this specification that encounters
20 extra_information_slice in a bitstream shall ignore it (i.e. parse from bitstream and discard).

21 **6.3.16 Macroblock**

22 Note "macroblock_stuffing" which was supported in ISO/IEC11172-2 shall not be used in a
23 bitstream conforming to this specification.

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

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

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

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

37 The previous_macroblock_address is a variable defining the absolute position of the last non-skipped
38 macroblock (see clause 7.6.6 for the definition of skipped macroblocks) except at the start of a slice. At
39 the start of a slice previous_macroblock_address is reset as follows:

40
$$\text{previous_macroblock_address} = (\text{mb_row}) * \text{mb_width} - 1;$$

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

43
$$\text{mb_column} = \text{macroblock_address} \% \text{mb_width}$$

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

45 Except at the start of a slice, if the value of macroblock_address recovered from
46 **macroblock_address_increment** and the **macroblock_escape** codes (if any) differs from the
47 previous_macroblock_address by more than one then some macroblocks have been skipped. The
48 bitstream shall comply with the following;

- 1 • There shall be no skipped macroblocks in I-pictures.
- 2 • In a B-picture there shall be no skipped macroblocks following a macroblock in which
3 macroblock_intra is one.
- 4 **macroblock_type** -- Variable length coded indicator which indicates the method of coding and content
5 of the macroblock according to the tables B-2 through B-8, selected by picture type and scalable mode.
- 6 **macroblock_quant** -- Derived from macroblock_type.
- 7 **macroblock_motion_forward** -- Derived from macroblock_type.
- 8 **macroblock_motion_backward** -- Derived from macroblock_type.
- 9 **macroblock_pattern** -- Derived from macroblock_type.
- 10 **macroblock_intra** -- Derived from macroblock_type.
- 11 **spatial_temporal_weight_code_flag** -- Derived from the macroblock_type. This indicates whether the
12 spatial_temporal_weight_code is present in the bitstream.
- 13 **spatial_temporal_weight_code** -- This is a two bit code which indicates, in the case of spatial
14 scalability, how the spatial and temporal predictions shall be combined to form the prediction for the
15 macroblock. A one bit code is used if spatial_temporal_weight_code_table_index, defined in
16 picture_spatial_scalable_extension(), is "00", otherwise a two bit code is used. A full description of
17 how to form the spatial scalable prediction is given in clause 7.7.
- 18 **spatial_temporal_weight_class** -- Derived from the spatial_temporal_weight_code if present,
19 otherwise from the macroblock_type. This class is used to indicate the number of motion vectors for
20 the macroblock in the coded bitstream and indicates how the motion vector predictors are updated as
21 defined in a combined table in clause 7.7.4
- 22 **frame_motion_type** - This is a two bit code indicating the macroblock motion prediction, defined in
23 Table 6-17.
- 24 If frame_pred_frame_dct is equal to 1 then frame_motion_type is omitted from the bitstream. In this
25 case vector decoding and prediction formation shall be performed as if frame_motion_type had
26 indicated "Frame-based prediction".
- 27 In the case of intra macroblocks (in a frame picture) when concealment_motion_vectors is equal to 1
28 frame_motion_type is not present in the bitstream. In this case vector decoding shall be performed as if
29 frame_motion_type had indicated "Frame-based prediction". See clause 7.6.3.9.

30 **Table 6-17 Meaning of frame_motion_type**

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

- 31
- 32 **field_motion_type** - This is a two bit code indicating the macroblock motion prediction, defined in
33 Table 6-18.
- 34 In the case of intra macroblocks (in a field picture) when concealment_motion_vectors is equal to 1
35 field_motion_type is not present in the bitstream. In this case vector decoding shall be performed as if
36 field_motion_type had indicated "Field-based prediction". See clause 7.6.3.9.

1

Table 6-18 Meaning of field_motion_type

code	prediction type	spatial_temporal_weight_class	motion_vector_count	mv_format	dmv
00	reserved				
01	Field-based prediction	0,1	1	field	0
10	16x8 MC	0,1	2	field	0
11	Dual-Prime	0	1	field	1

2

3 **dct_type** - This is a one-bit integer indicating whether the macroblock is frame DCT coded or field
4 DCT coded. If this is set to "1", the macroblock is field DCT coded. `dct_type` is only included in the
5 bitstream if `decode_dct_type` is non-zero. `decode_dct_type` is derived as follows:

```
6         if ( (picture_structure == "frame") && (frame_pred_frame_dct == 0) &&
7             (macroblock_intra || macroblock_pattern) ) {
8             decode_dct_type = 1;
9         } else
10            decode_dct_type = 0;
```

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

13

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

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

14

15 `motion_vector_count` is derived from `field_motion_type` or `frame_motion_type` as indicated in Table 6-
16 17 and Table 6-18.

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

20 `dmv` is derived from `field_motion_type` or `frame_motion_type` as indicated in Table 6-17 and Table 6-
21 18

22 **coded_block_pattern()** -- For 4:2:0 source format, the coded block pattern is a variable length code
23 "coded_block_pattern_420" that is used to derive the variable `cbp` according to Table B-9 in Annex B.

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

```

1      for (i=0; i<12; i++)
2          if (macroblock_intra != 0)
3              pattern_code[i] = 1;
4          else
5              pattern_code[i] = 0;
6
7      if (macroblock_intra == 0) {
8          for (i=0; i<6; i++)
9              if ( cbp & (1<<(5-i)) ) pattern_code[i] = 1;
10         if (chroma_format == "4:2:2")
11             for (i=6; i<8; i++)
12                 if ( coded_block_pattern_1 & (1<<(7-i)) ) pattern_code[i] = 1;
13         if (chroma_format == "4:4:4")
14             for (i=6; i<12; i++)
15                 if ( coded_block_pattern_2 & (1<<(11-i)) ) pattern_code[i] = 1;
16     }
17

```

18 If pattern_code[i] equals to 1, i=0 to (block_count-1), then the block number i defined in Figures 6-9,
19 6-10 and 6-11 is contained in this macroblock.

20 The number "block_count" which determines the number of blocks in the macroblock is derived from
21 the chroma format as shown in Table 6-20.

22 **Table 6-20 block_count as a function of chroma format**

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

23

24 **6.3.17 Block**

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

1 7 The video decoding process

2 This clause specifies the decoding process that a decoder shall perform to recover picture data from the
3 coded bit-stream.

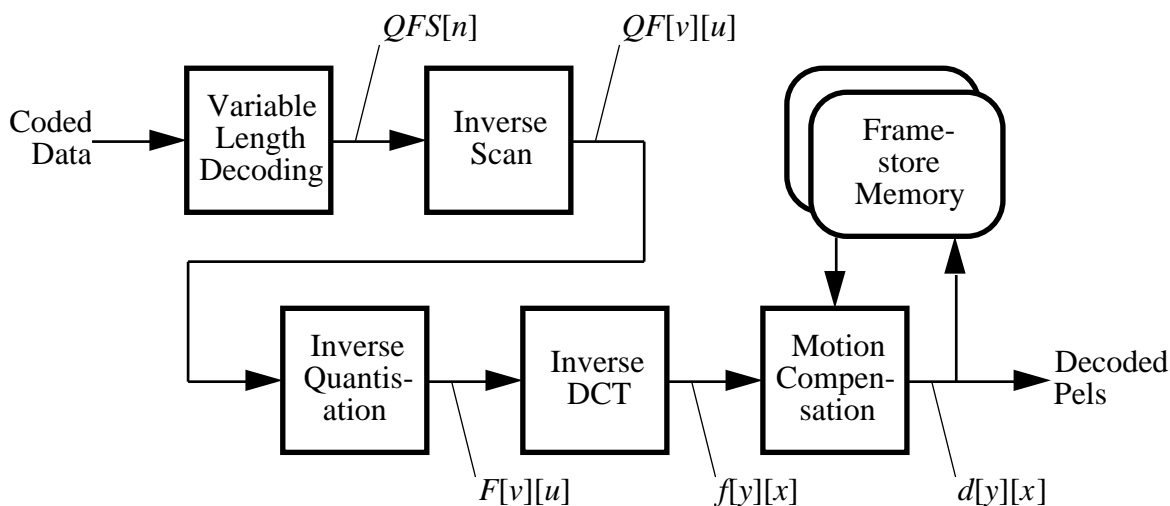
4 With the exception of the Inverse Discrete Cosine Transform (IDCT) the decoding process is defined
5 such that all decoders shall produce numerically identical results. Any decoding process that produces
6 identical results to the process described here, by definition, complies with this specification.

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

9 In clauses 7.1 through 7.6 the simplest decoding process is specified in which no scalability features are
10 used. Clauses 7.7 to 7.10 document the decoding process when scalable extensions are used.

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

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



16
17

18

Figure 7-1. Simplified Video Decoding Process

19 7.1 Higher syntactic structures

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

24 Reconstructed field pictures shall be associated together in pairs to form reconstructed frames. (See
25 "picture_structure" in clause 6.3.10.)

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

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

29 If progressive_sequence == 0 the reconstructed frames shall be broken into a sequence of fields which
30 shall be output from the decoding process at regular intervals of the field period as shown in Figure 6-
31 16. In the case that a frame picture has repeat_first_field == 1 the first field of the frame shall be
32 repeated after the second field. (See "repeat_first_field" in clause 6.3.10.)

33

1 7.2 Variable length decoding

2 Clause 7.2.1 specifies the decoding process used for the DC coefficient ($n=0$) in an intra coded block.
 3 Clause 7.2.2 specifies the decoding process for all other coefficients; AC coefficients ($n \neq 0$) and DC
 4 coefficients in non-intra coded blocks.

5 Let cc denote the colour component. It is related to the block number as specified in Table 7-?.

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

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

7

8 7.2.1 DC intra coefficients

9 DC intra coefficients are encoded as a variable length code denoting dct_dc_size as defined in Table B-
 10 12 and B-13. If dct_dc_size is not equal to zero then this shall be followed by a fixed length code,
 11 $dc_dct_differential$, of dct_dc_size bits. A differential value is first calculated which is added to a
 12 predictor in order to recover the final decoded coefficient.

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

15 Three predictors are maintained, one for each of the colour components, cc . Each time a DC intra
 16 coefficient is decoded the predictor is added to the differential to recover the actual coefficient. Then
 17 the predictor shall be set to the value of the coefficient just decoded. At various times the predictors
 18 shall be reset. The reset value is related to the parameter $intra_dc_precision$ that is encoded in the
 19 picture coding extension.

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

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

21

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

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

26 The predictors are denoted $dc_dct_pred[cc]$.

```

1  QFS[0] shall be calculated from dc_dct_size and dc_dct_differential by any process equivalent to:
2      if ( dc_dct_size == 0 ) {
3          dct_diff = 0;
4      } else {
5          half_range = 2 ^ ( dc_dct_size - 1 );           Note ^ denotes power (not XOR)
6          if ( dc_dct_differential >= half_range )
7              dct_diff = dc_dct_differential;
8          else
9              dct_diff = (dc_dct_differential + 1) - (2 * half_range);
10     }
11     QFS[0] = dc_dct_pred[cc] + dct_diff;
12     dc_dct_pred[cc] = QFS[0]
13

```

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

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

17 0 to $((2^{(8 + \text{intra_dc_precision})} - 1)$

18 7.2.2 Other coefficients

19 All coefficients with the exception of the DC intra coefficients shall be encoded using Tables B-14, B-
20 15 and B-16.

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

- 23 1 End of Block. In this case there are no more coefficients in the block in which case the
24 remainder of the coefficients in the block (those for which no value has yet been decoded)
25 shall be set to zero. This is denoted by “End of block” in the syntax specification of clause
26 6.??.
- 27 2 A “normal” coefficient in which a value of *run* and *level* is decoded followed by a single bit, *s*,
28 giving the sign of the coefficient *signed_level* is computed from *level* and *s* as shown below.
29 *run* coefficients shall be set to zero and the subsequent coefficient shall have the value
30 *signed_level*.
31 if (*s*)
32 *signed_level* = *level*;
33 else
34 *signed_level* = (-*level*);
35
- 36 3 An “Escape” coded coefficient. In which the values of *run* and *signed_level* are fixed length
37 coded as described below.

38 7.2.2.1 Table selection

39 Table 7-? indicates which table shall be used for decoding the DCT coefficients.

40 **Table 7-3. Selection of DCT coefficient VLC tables**

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

41

1 7.2.2.2 First coefficient of a non-intra block

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

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

7 The positions in the syntax that use this modified table are denoted by “First DCT coefficient” in the
8 syntax specification of clause 6.?.?. The remainder of the coefficients are denoted by “Subsequent
9 DCT coefficients”.

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

13 7.2.2.3 Escape coding

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

16 Table B-15 shows the escape coding method. The Escape VLC is followed by a 6 bit fixed length code
17 giving “run”. This is followed by a 12 bit fixed length code giving the values of “signed_level”.

18 If a variable length code exists for a specific combination of *run* and *signed_level* then the escape
19 coding method shall not be used.

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

22 7.2.2.4 Summary

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

```

25     eob_not_read = 1;
26     while ( eob_not_read )
27     {
28         <decode VLC, decode Escape coded coefficient if required>
29         if ( <decoded VLC indicates End-of-block> ) {
30             eob_not_read = 0;
31             while ( n < 64 ) {
32                 QFS[n] = 0;
33                 n = n + 1;
34             }
35         } else {
36             for ( m = 0; m < run; m++ ) {
37                 QFS[n] = 0;
38                 n = n + 1;
39             }
40             QFS[n] = signed_level
41             n = n + 1;
42         }
43     }

```

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

45 7.3 Inverse scan

46 Let the data at the output of the variable length decoder be denoted by $QFS[n]$. *n* is in the range 0 to
47 63.

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

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

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

3

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

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

5

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

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

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

8 for ($v=0; v<8; v++$)

9 for ($u=0; u<8; u++$)

10 $QF[v][u] = QFS[scan[alternate_scan][v][u]]$

11

12 **7.3.1 Inverse scan for matrix download**

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

16 In this case the scan defined by Figure 7-1 (i.e. scan zero) shall always be used.

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

20 for ($v=0; v<8; v++$)

21 for ($u=0; u<8; u++$)

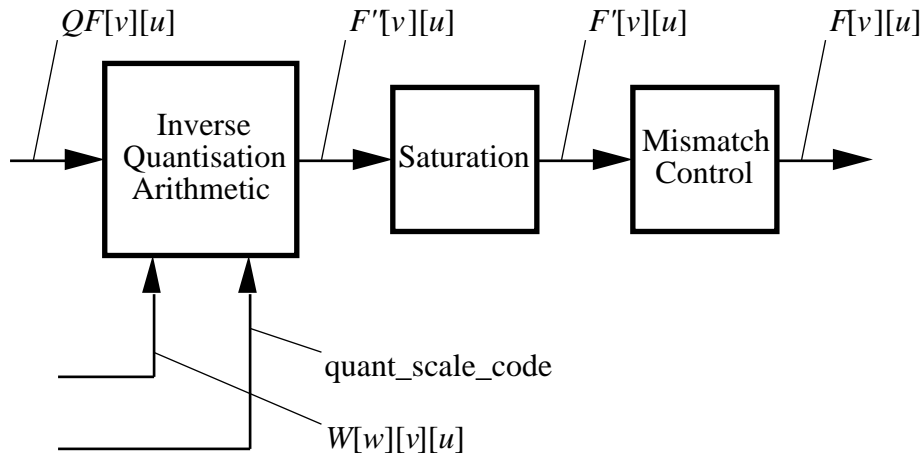
22 $W[w][v][u] = W'[w][scan[0][v][u]]$

23

24 **7.4 Inverse Quantisation**

25 The two-dimensional array of coefficients, $QF[v][u]$, is inverse quantised to produce the reconstructed
 26 DCT coefficients. This process is essentially a multiplication by the quantiser step size. The quantiser
 27 step size is modified by two mechanisms; a weighting matrix is used to modify the step size within a

1 block and a scale factor is used in order that the step size can be modified at the cost of only a few bits
 2 (as compared to encoding an entire new weighting matrix).



3
 4 **Figure 7-3. Inverse quantisation process**

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

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

10 **7.4.1 Intra DC coefficient**

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

13 In intra blocks $F''[0][0]$ shall be obtained by multiplying $QF[0][0]$ by a constant multiplier,
 14 *intra_dc_mult*, (constant in the sense that it is not modified by either the weighting matrix or the scale
 15 factor). The multiplier is related to the parameter *intra_dc_precision* that is encoded in the picture
 16 coding extension. If no picture_coding_extension has been decoded (i.e. ISO/IEC 11172-2) then
 17 *intra_dc_precision* shall be assumed to be zero (meaning eight bits). Table 7-? specifies the relation
 18 between *intra_dc_precision* *intra_dc_mult*.

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

<i>intra_dc_precision</i>	Bits of precision	<i>intra_dc_mult</i>
0	8	8
1	9	4
2	10	2
3	11	1

20

21 Thus; $F''[0][0] = \textit{intra_dc_mult} \times QF[0][0]$

22

23 **7.4.2 Other coefficients**

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

26 **7.4.2.1 Weighting matrices**

27 When 4:2:0 data is used two weighting matrices are used. One shall be used for intra macroblocks and
 28 the other for non-intra macroblocks. When 4:2:2 or 4:4:4 data is used four matrices are used allowing

- 1 different matrices to be used for luminance and chrominance data. Each matrix has a default set of
 2 values which may be overwritten by down-loading a user defined matrix as explained in clause 6.?.
 3 Let the weighting matrices be denoted by $W[w][v][u]$ where w takes the values 0 to 3 indicating which
 4 of the matrices is being used. Table 7-? summarises the rules governing the selection of w .

5 **Table 7-5. Selection of w**

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

6

7 **7.4.2.2 Quantiser scale factor**

- 8 The quantisation scale factor is encoded as a fixed length code, *quantiser_scale_code*. This indicates
 9 the appropriate *quantiser_scale* to apply in the inverse quantisation arithmetic.
 10 *q_scale_type* (encoded in the picture coding extension) indicates which of two mappings between
 11 *quantiser_scale_code* and *quantiser_scale* shall apply. Table 7-? shows the two mappings between
 12 *quantiser_scale_code* and *quantiser_scale*.

1

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

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

2

3 **7.4.2.3 Reconstruction formulae**

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

$$F'[v][u] = ((2QF[v][u] + k) \times W[w][v][u] \times \text{quantiser_scale}) / 32$$

6

where:

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

7

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

1 **7.4.3 Saturation**

2 The coefficients resulting from the mismatch control process are saturated to lie in the range
3 $[-2048;+2047]$. Thus;

$$4 \quad F'[v][u] = \begin{cases} 2047 & F'[v][u] > 2047 \\ F'[u][v] & -2048 \leq F'[v][u] \leq 2047 \\ -2048 & F'[v][u] < -2048 \end{cases}$$

5

6 **7.4.4 Mismatch control**

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

$$10 \quad sum = \sum_{v=0}^{v<8} \sum_{u=0}^{u<8} F'[v][u]$$

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

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

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

15 **7.4.5 Summary**

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

17

```

18   for (v=0; v<8;v++) {
19     for (u=0; u<8;u++) {
20       if ( (u==0) && (v==0) && (macroblock_intra) ) {
21         F''[v][u] = intra_dc_mult * QF[v][u];
22       } else {
23         if ( macroblock_intra ) {
24           F''[v][u] = ( QF[v][u] * W[w][v][u] * quantiser_scale * 2 ) / 32;
25         } else {
26           F''[v][u] = ( ( ( QF[v][u] * 2 ) + Sign(QF[v][u]) ) * W[w][v][u]
27             * quantiser_scale ) / 32;
28         }
29       }
30     }
31   }

```

32

```

1      sum = 0;
2      for (v=0; v<8;v++) {
3          for (u=0; u<8;u++) {
4              if ( F'[v][u] > 2047 ) {
5                  F'[v][u] = 2047;
6              } else {
7                  if ( F'[v][u] < -2048 ) {
8                      F'[v][u] = -2048;
9                  } else {
10                     F'[v][u] = F''[v][u];
11                 }
12                 sum = sum + F'[v][u];
13                 F[v][u] = F'[v][u];
14             }
15         }
16     }
17
18     if ((sum & 1) == 0) {
19         if ((F[7][7] & 1) != 0) {
20             F[7][7] = F'[7][7] - 1;
21         } else {
22             F[7][7] = F'[7][7] + 1;
23         }
24     }
25

```

26 7.5 Inverse DCT

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

30 7.5.1 Non-coded blocks and skipped macroblocks

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

34 However, if `pattern_code[i]` is zero, or if the macroblock is skipped, then that block contains no
 35 coefficient data. In this case it is not necessary to perform the processes; Variable length decoding,
 36 Inverse scan, Inverse quantisation or Inverse DCT for that block. The pel domain coefficients $f[y][x]$
 37 for such a block shall all take the value zero.

38 Note In many implementations a block of all zero coefficients $QF[u][v]$ will result in a block of
 39 all zero pel domain coefficients, $f[y][x]$, despite the fact that the reconstructed transform
 40 domain coefficients $F[v][u]$ will not all be zero (because $F[7][7]$ will take the value 1 due
 41 to mismatch control).

42 7.6 Motion compensation

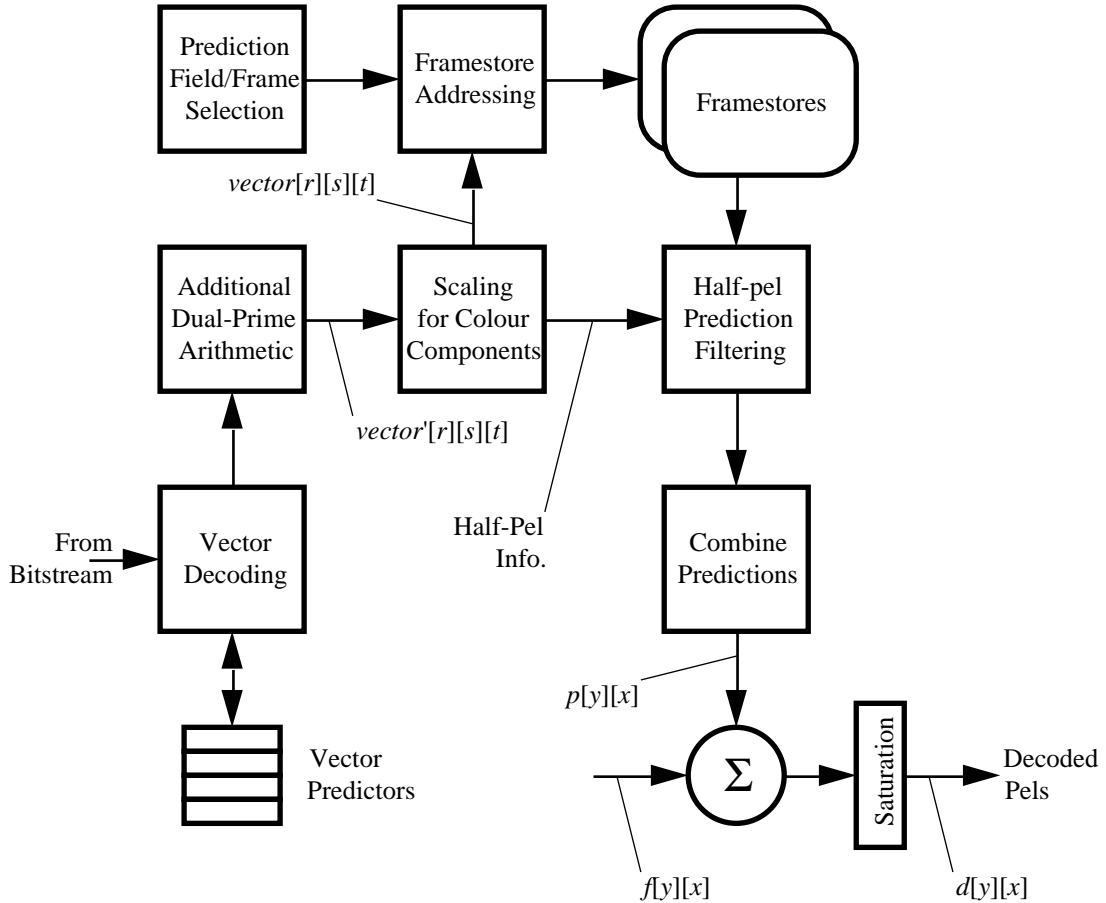
43 The motion compensation process forms predictions from previously decoded pictures which are
 44 combined with the coefficient data (from the output of the IDCT) in order to recover the final decoded
 45 pels. Figure 7-4 shows a simplified diagram of this process.

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

48 In the case of intra coded macroblocks no prediction is formed so that $p[y][x]$ will be zero. The
 49 saturation shown in figure 7-4 is still required in order to remove negative values from $f[y][x]$. Intra
 50 coded macroblocks may carry motion vectors known as "concealment vectors". Despite this no
 51 prediction is formed in the normal course of events. This vector information is intended for use in the
 52 case that bitstream errors preclude the decoding of coefficient information. The way in which a
 53 decoder shall use this information is not specified. The only requirement for these vectors is that they

1 shall have the correct syntax for motion vectors. A description of the way in which these vectors may
 2 be used can be found in clause 7.7.

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



6

7

Figure 7-4. Simplified motion compensation process

8 7.6.1 Prediction modes

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

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

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

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

- 1 • 16x8 motion compensation. In which two motion vectors are used for each macroblock. The
 2 first vector is used for the upper 16x8 region, the second for the lower 16x8 region. In the
 3 case of a bidirectionally predicted macroblock a total of four vectors will be used since there
 4 will be two for the forward prediction and two for the backward prediction. In this
 5 specification 16x8 motion compensation shall only be used with field pictures.
- 6 • Dual-prime. In which only one vector is encoded (in its full format) in the bitstream together
 7 with a small differential vector. In the case of field pictures two motion vectors are then
 8 derived from this information. These are used to form predictions from two reference fields
 9 (one top, one bottom) which are averaged to form the final prediction. In the case of frame
 10 pictures this process is repeated for the two fields so that a total of four field predictions are
 11 made. This mode shall only be used in P-pictures where there are no B-pictures between the
 12 predicted and reference fields or frames.

13

14 7.6.2 Prediction field and frame selection

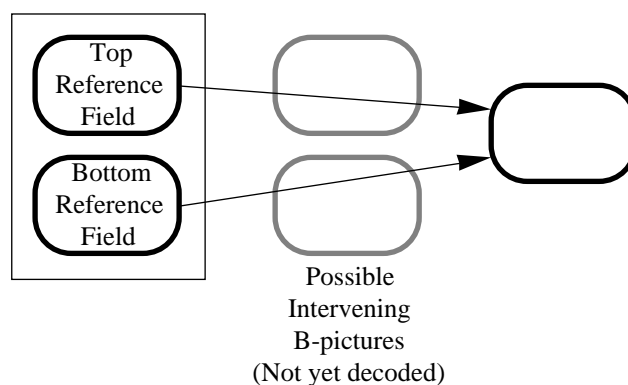
15 The selection of which fields and frames shall be used to form predictions shall be made as detailed in
 16 this clause.

17 7.6.2.1 Field prediction

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

22 Note 1: The reference fields may themselves have been two field-pictures or a single frame-
 23 picture.

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



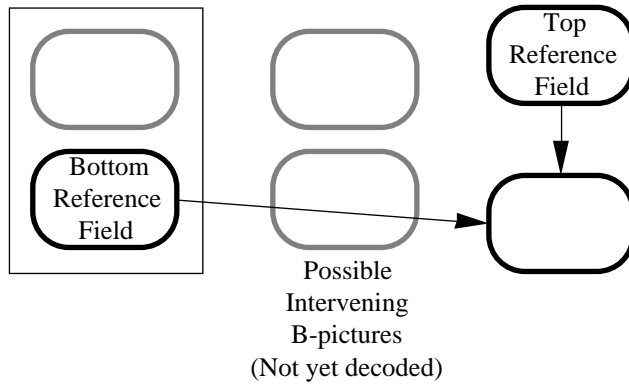
26

27

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

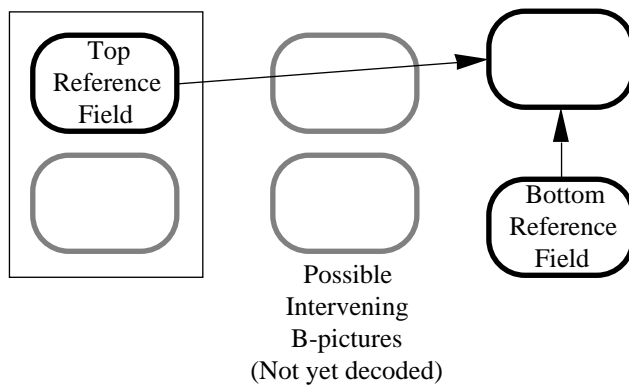
28 The case when predicting the second of two field pictures is more complicated because the two most
 29 recently decoded I field or P field pictures shall be used. Figure 7-6 illustrates the situation when this
 30 second picture is the bottom field. Figure 7-7 illustrates the situation when this second picture is the
 31 top field.

32 Note: The earlier reference field may itself have been a field picture or part of a frame picture.



1
2
3

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



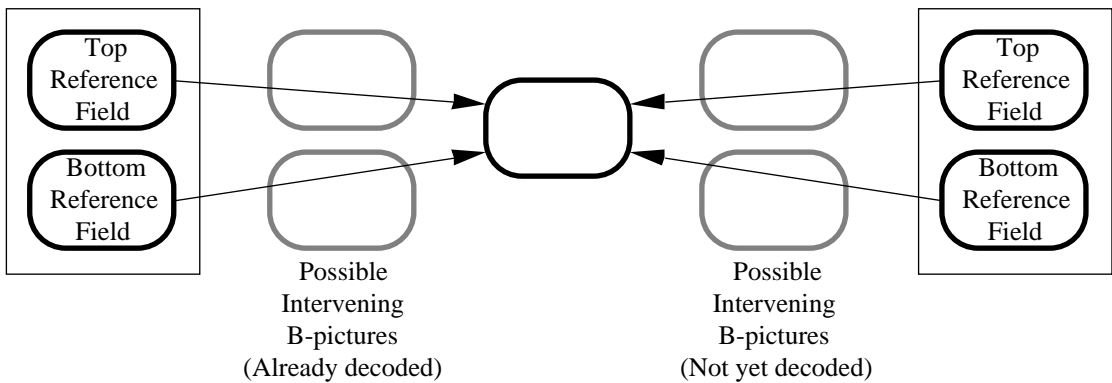
4
5

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

6 Field prediction in B-pictures shall be made from the two fields of the two most recently decoded I
7 frame pictures or P frame pictures. Figure 7-8 illustrates this situation.

8 Note 1: The reference fields may themselves have been two field-pictures or a single frame-
9 picture.

10 Note 2: In the case of predicting a field B-picture, the same reference fields shall be used to
11 predict both the top field-picture and the bottom field-picture.



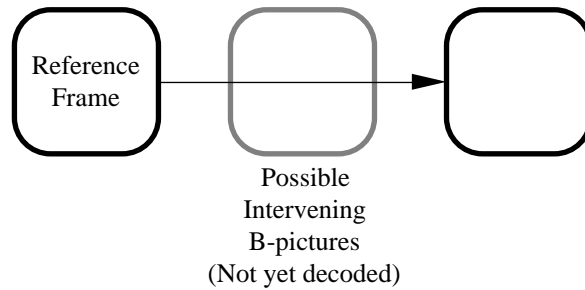
12
13
14
15

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

1 **7.6.2.2 Frame prediction**

2 In P-pictures prediction shall be made from the most recently decoded frame that was itself an I-picture
 3 and/or P-picture. This is illustrated in figure 7-9.

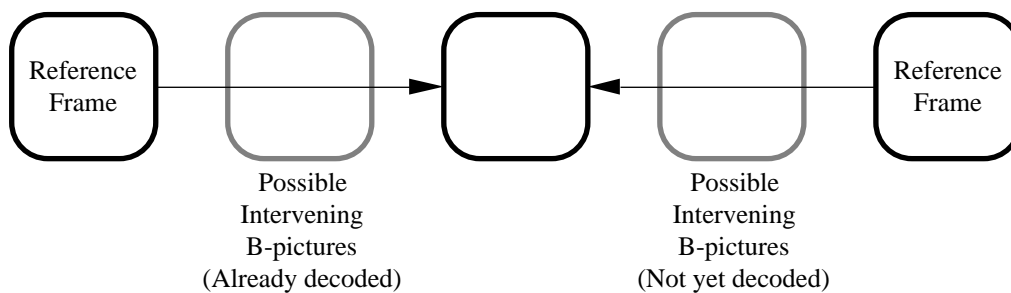
4 Note: The reference frame may itself have been two field-pictures or a single frame-picture.



5
 6 **Figure 7-9. Frame-prediction for I-pictures and P-pictures**

7 Similarly frame prediction in B-pictures shall be made from the two most recently decoded I frame
 8 pictures or P frame pictures as illustrated in figure 7-10.

9 Note: The reference frames themselves may each have been either two field-pictures or a single
 10 frame-picture.



11
 12 **Figure 7-10. Frame-prediction for B-pictures**

13 **7.6.3 Motion vectors**

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

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

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

21

1 **7.6.3.1 Decoding the motion vectors**

2 There are four values denoted $f_code[s][t]$ that are recovered from the bitstream as defined in Table 7-
3 8.

4 **Table 7-8. Recovery of $f_code[s][t]$**

$f_code[0][0]$	forward_horizontal_f_code
$f_code[1][0]$	backward_horizontal_f_code
$f_code[0][1]$	forward_vertical_f_code
$f_code[1][1]$	backward_vertical_f_code

5

6 Note Two flags in the bitstream; `full_pel_forward_vector` and `full_pel_backward_vector` are
7 used in ISO/IEC 11172-2 but play no part in the decoding process described in this
8 specification.

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

11

12 $r_size = f_code[s][t] - 1$

13 $f = 1 \ll r_size$

14 $high = (16 * f) - 1;$

15 $low = ((-16) * f);$

16 $range = (32 * f);$

17

18 if ($f == 1$ || (`motion_code` == 0))

19 $\delta = \text{motion_code};$

20 else {

21 $\delta = ((\text{Abs}(\text{motion_code}) - 1) * f) + \text{motion_residual} + 1;$

22 if (`motion_code` < 0)

23 $\delta = -\delta;$

24 }

25

26 $prediction = PMV[r][s][t];$

27 if ((`mv_format` == "field") && ($t == 1$) && (`picture_structure` == "Frame-Picture"))

28 $prediction = PMV[r][s][t] \text{ DIV } 2;$

29

30 $vector'[r][s][t] = prediction + \delta;$

31 if ($vector'[r][s][t] < low$)

32 $vector'[r][s][t] = vector'[r][s][t] + range;$

33 if ($vector'[r][s][t] > high$)

34 $vector'[r][s][t] = vector'[r][s][t] - range;$

35

36 if ((`mv_format` == "field") && ($t == 1$) && (`picture_structure` == "Frame-Picture"))

37 $PMV[r][s][t] = vector'[r][s][t] * 2;$

38 else

39 $PMV[r][s][t] = vector'[r][s][t];$

40

41 The parameters in the bitstream shall be such that the reconstructed vector, $vector'[r][s][t]$, and the
42 updated value of the motion vector predictor $PMV[r][s][t]$, shall lie in the range [$low : high$].

43 $r_size, f, \delta, high, low$ and $range$ are temporary variables that are not used in the remainder of this
44 specification.

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

46 r, s and t specify the particular motion vector component being processed as identified in table 7-?.

47 $vector'[r][s][t]$ is the final reconstructed motion vector for the luminance component of the
48 macroblock.

1 7.6.3.2 Vector restrictions

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

6

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

$f_code[s][t]$	Vertical components ($t==1$) of field vectors in frame pictures	All other cases
0	(forbidden)	
1	-4, +3.5	-8, +7.5
2	-8, +7.5	-16, +15.5
3	-16, +15.5	-32, +31.5
4	-32, +31.5	-64, +63.5
5	-64, +63.5	-128, +127.5
6	-128, +127.5	-256, +255.5
7	-256, +255.5	-512, +511.5
8	-512, +511.5	-1024, +1023.5
9	-1024, +1023.5	-2048, +2047.5
10-15	(reserved)	

8

9 7.6.3.3 Updating motion vector predictors

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

15 The motion vector predictors shall be updated as specified in table 7-? and 7-?. The rules for updating
 16 motion vector predictors in the case of skipped macroblocks are specified in clause 7.6.6.

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

1

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

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

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

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

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

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

2

3

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

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

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

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

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

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

4

1 **7.6.3.4 Resetting motion vector predictors**

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

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

8 **7.6.3.5 Prediction in P-pictures**

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

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

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

- 17 • The prediction mode shall be “Frame-based”
- 18 • The (frame) motion vector shall be zero (0,0)
- 19 • The motion vector predictors shall be reset to zero

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

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

29 **7.6.3.6 Dual prime additional arithmetic**

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

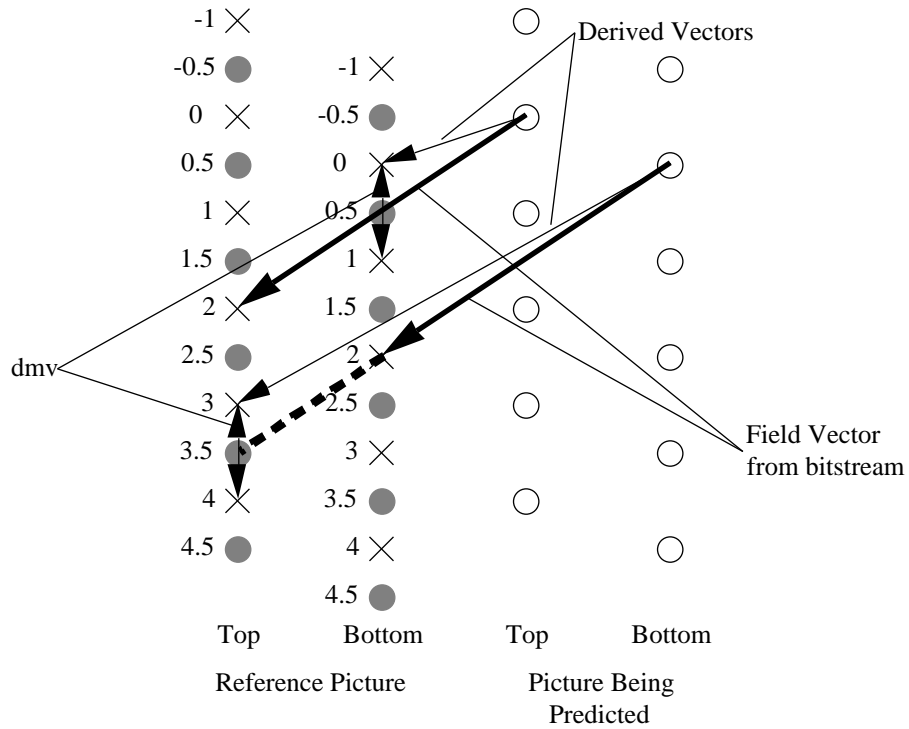


Figure 7-11. Scaling of vectors for dual prime prediction

1
2

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

8 $dmvector[0]$ is the horizontal component of the differential motion vector and $dmvector[1]$ the vertical
 9 component. The two components of the differential motion vector shall be decoded directly using
 10 Table B-8a and shall take only one of the values -1, 0, +1.

11 $m[parity_ref][parity_pred]$ is the field distance between the predicted field and the reference field as
 12 defined in Table 7-?. " $parity_ref$ " is the parity of the reference field for which the new vector is being
 13 computed. " $parity_pred$ " is the parity of the field that shall be predicted.

14 $e[parity_ref][parity_pred]$ is the adjustment necessary to reflect the vertical shift between the lines of
 15 top field and bottom field as defined in Table 7-?.

16

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

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

17

Table 7-13. Definition of $e[\text{parity_ref}][\text{parity_pred}]$

<i>parity_ref</i>	<i>parity_pred</i>	$e[\text{parity_ref}][\text{parity_pred}]$
0	0	0
0	1	+1
1	0	-1
1	1	0

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

$$\begin{aligned} \text{vector}'[r][0][0] &= ((\text{vector}'[0][0][0] * m[\text{parity_ref}][\text{parity_pred}])//2) + dm\text{vector}[0]; \\ \text{vector}'[r][0][1] &= ((\text{vector}'[0][0][1] * m[\text{parity_ref}][\text{parity_pred}])//2) \\ &\quad + e[\text{parity_ref}][\text{parity_pred}] + dm\text{vector}[1]; \end{aligned}$$

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

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

7.6.3.7 Vectors for colour difference components

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

$$\text{vector}[r][s][t] = \text{vector}'[r][s][t] \quad (\text{for all } r, s \text{ and } t)$$

For each of the two colour difference components the vectors shall be scaled as follows:

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

$$\begin{aligned} \text{vector}[r][s][0] &= \text{vector}'[r][s][0] / 2; \\ \text{vector}[r][s][1] &= \text{vector}'[r][s][1] / 2; \end{aligned}$$

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

$$\begin{aligned} \text{vector}[r][s][0] &= \text{vector}'[r][s][0] / 2; \\ \text{vector}[r][s][1] &= \text{vector}'[r][s][1]; \end{aligned}$$

4:4:4 The vector is unmodified:

$$\begin{aligned} \text{vector}[r][s][0] &= \text{vector}'[r][s][0]; \\ \text{vector}[r][s][1] &= \text{vector}'[r][s][1]; \end{aligned}$$

7.6.3.8 Semantic restrictions concerning predictions

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

As explained in clause 6.1.3 it is, in general, not necessary for the slices to cover the entire picture. However in most defined levels of defined profiles the "restricted slice structure" is used in which case the slices do cover the entire picture. In this case the semantic rule may be simplified to say that it is a restriction of the bitstream that reconstructed motion vectors shall not refer to pels outside the boundary of the coded picture.

7.6.3.9 Concealment motion vectors

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

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

- 6 • In a field picture; `field_motion_type = "Field-based prediction"`
- 7 • In a frame picture; `frame_motion_type = "Frame-based prediction"`

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

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

- 15 • For all macroblocks except those in the bottom row of macroblocks concealment motion
 16 vectors should be appropriate for use in the macroblock that lies vertically below the
 17 macroblock in which the vector occurs.
- 18 • When the motion vector is used with respect to the macroblock identified in the previous rule
 19 then it a decoder must assume that the vector may refer to pels outside of the slices encoded in
 20 the reference picture.

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

24

25 **7.6.4 Forming predictions**

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

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

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

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

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

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

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

41 For each prediction block the integer pel vectors `int_vec[t]` and the half pel flags `half_flag[t]` shall be
 42 formed as follows;

```

43     for (t=0; t<2; t++) {
44         int_vec[t] = vector[r][s][t] DIV 2;
45         if ((vector[r][s][t] - (2 * int_vec[t]) != 0)
46             half_flag[t] = 1;
47         else
48             half_flag[t] = 0;
49     }
  
```

50 Then for each pel in the prediction block the pels are read and the half pel prediction applied as
 51 follows;

```

1      if ( (! half_flag[0] ) && (! half_flag[1] )
2          pel_pred[y][x] = pel_ref[y + int_vec[1]][x + int_vec[0]] ;
3
4      if ( (! half_flag[0] ) && half_flag[1] )
5          pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
6                          pel_ref[y + int_vec[1]+1][x + int_vec[0]] ) // 2;
7
8      if ( half_flag[0] && (! half_flag[1] ) )
9          pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
10                          pel_ref[y + int_vec[1]][x + int_vec[0]+1] ) // 2;
11
12     if ( half_flag[0] && half_flag[1] )
13         pel_pred[y][x] = ( pel_ref[y + int_vec[1]][x + int_vec[0]] +
14                             pel_ref[y + int_vec[1]][x + int_vec[0]+1] +
15                             pel_ref[y + int_vec[1]+1][x + int_vec[0]] +
16                             pel_ref[y + int_vec[1]+1][x + int_vec[0]+1] ) // 4;
17

```

18 where $pel_pred[y][x]$ is the prediction pel being formed and $pel_ref[y][x]$ are pels in the reference field
19 or frame.

20 7.6.5 Motion vector selection

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

1

Table 7-14. Predictions and vectors in field pictures

field_ motion_ type	macroblock_motion_		macro- block_ intra	Vector	Prediction formed for
	forward	backward			
Field-based [‡]	-	-	1	$vector'[0][0][1:0]$ [◇]	None (vector is for concealment)
Field-based	1	1	0	$vector'[0][0][1:0]$ $vector'[0][1][1:0]$	whole field, forward whole field, backward
Field-based	1	0	0	$vector'[0][0][1:0]$	whole field, forward
Field-based	0	1	0	$vector'[0][1][1:0]$	whole field, backward
Field-based [‡]	0	0	0	$vector'[0][0][1:0]$ ^{*§}	whole field, forward
16x8 MC	1	1	0	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$ $vector'[0][1][1:0]$ $vector'[1][1][1:0]$	upper 16x8 field, forward lower 16x8 field, forward upper 16x8 field, backward lower 16x8 field, backward
16x8 MC	1	0	0	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$	upper 16x8 field, forward lower 16x8 field, forward
16x8 MC	0	1	0	$vector'[0][1][1:0]$ $vector'[1][1][1:0]$	upper 16x8 field, backward lower 16x8 field, backward
Dual prime	1	0	0	$vector'[0][0][1:0]$ $vector'[2][0][1:0]$ ^{*†}	same parity whole field, forward opposite parity whole field, forward
Note: Vectors are listed in the order they appear in the bitstream ◇ the vector is only present if concealment_motion_vectors is one ‡ field_motion_type is not present in the bitstream but is assumed to be Field-based * These vectors are not present in the bitstream † These vectors are derived from $vector'[0][0][1:0]$ as described in clause 7-? § The vector is taken to be (zero, zero) as explained in clause 7-?					

2

1

Table 7-15. Predictions and vectors in frame pictures

frame_ motion_ type	macroblock_motion_		macro- block_ intra	Vector	Prediction formed for
	forward	backward			
Frame-based [‡]	-	-	1	$vector'[0][0][1:0]^\diamond$	None (vector is for concealment)
Frame-based	1	1	0	$vector'[0][0][1:0]$	frame, forward
				$vector'[0][1][1:0]$	frame, backward
Frame-based	1	0	0	$vector'[0][0][1:0]$	frame, forward
Frame-based	0	1	0	$vector'[0][1][1:0]$	frame, backward
Frame-based [‡]	0	0	0	$vector'[0][0][1:0]^{*\S}$	frame, forward
Field-based	1	1	0	$vector'[0][0][1:0]$	top field, forward
				$vector'[1][0][1:0]$	bottom field, forward
				$vector'[0][1][1:0]$	top field, backward
				$vector'[1][1][1:0]$	bottom field, backward
Field-based	1	0	0	$vector'[0][0][1:0]$	top field, forward
				$vector'[1][0][1:0]$	bottom field, forward
Field-based	0	1	0	$vector'[0][1][1:0]$	top field, backward
				$vector'[1][1][1:0]$	bottom field, backward
Dual prime	1	0	0	$vector'[0][0][1:0]$	same parity top field, forward
				$vector'[0][0][1:0]^*$	same parity bottom field, forward
				$vector'[2][0][1:0]^{*\dagger}$	opposite parity top field, forward
				$vector'[3][0][1:0]^{*\dagger}$	opposite parity bottom field, forward

Note: Vectors are listed in the order they appear in the bitstream

\diamond the vector is only present if **concealment_motion_vectors** is one

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

*

† These vectors are derived from $vector'[0][0][1:0]$ as described in clause 7-?

\S The vector is taken to be (zero, zero) as explained in clause 7-?

2

3 7.6.6 Skipped Macroblocks

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

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

9 There are no skipped macroblocks in I-pictures.

10 7.6.6.1 P field-picture

- 11 • The prediction shall be made as if field_motion_type is "Field-based"
- 12 • The prediction shall be made from the field of the same parity as the field being predicted.
- 13 • Motion vector predictors shall be reset to zero.
- 14 • The motion vector shall be zero.

15

1 **7.6.6.2 P frame-picture**

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

5

6 **7.6.6.3 B field-picture**

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

14

15 **7.6.6.4 B frame-picture**

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

22

23 **7.6.7 Combining predictions**

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

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

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

29 **7.6.7.1 Simple frame predictions**

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

34
$$pel_pred[y][x] = (pel_pred_forward[y][x] + pel_pred_backward[y][x])/2;$$

35 **7.6.7.2 Simple field predictions**

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

39 **7.6.7.3 16x8 Motion compensation**

40 In this prediction mode separate predictions are formed for the upper 16x8 region of the macroblock
 41 and the lower 16x8 region of the macroblock. Note that in the case of 4:2:0 data that the formation of
 42 predictions of size 8 pels by 4 lines for colour difference components is required.

43 **7.6.7.4 Dual prime**

44 In dual prime mode two predictions are formed for each field in an analogous manner to the backward
 45 and forward predictions in B-pictures. If `pel_pred_same_parity[y][x]` is the prediction pel from the

1 same parity field and $pel_pred_opposite_parity[y][x]$ is the corresponding pel from the opposite
2 prediction then the final prediction pel shall be formed as;

3 $pel_pred[y][x] = (pel_pred_same_parity[y][x] + pel_pred_opposite_parity[y][x]) // 2;$

4 **7.6.8 Adding prediction and coefficient data**

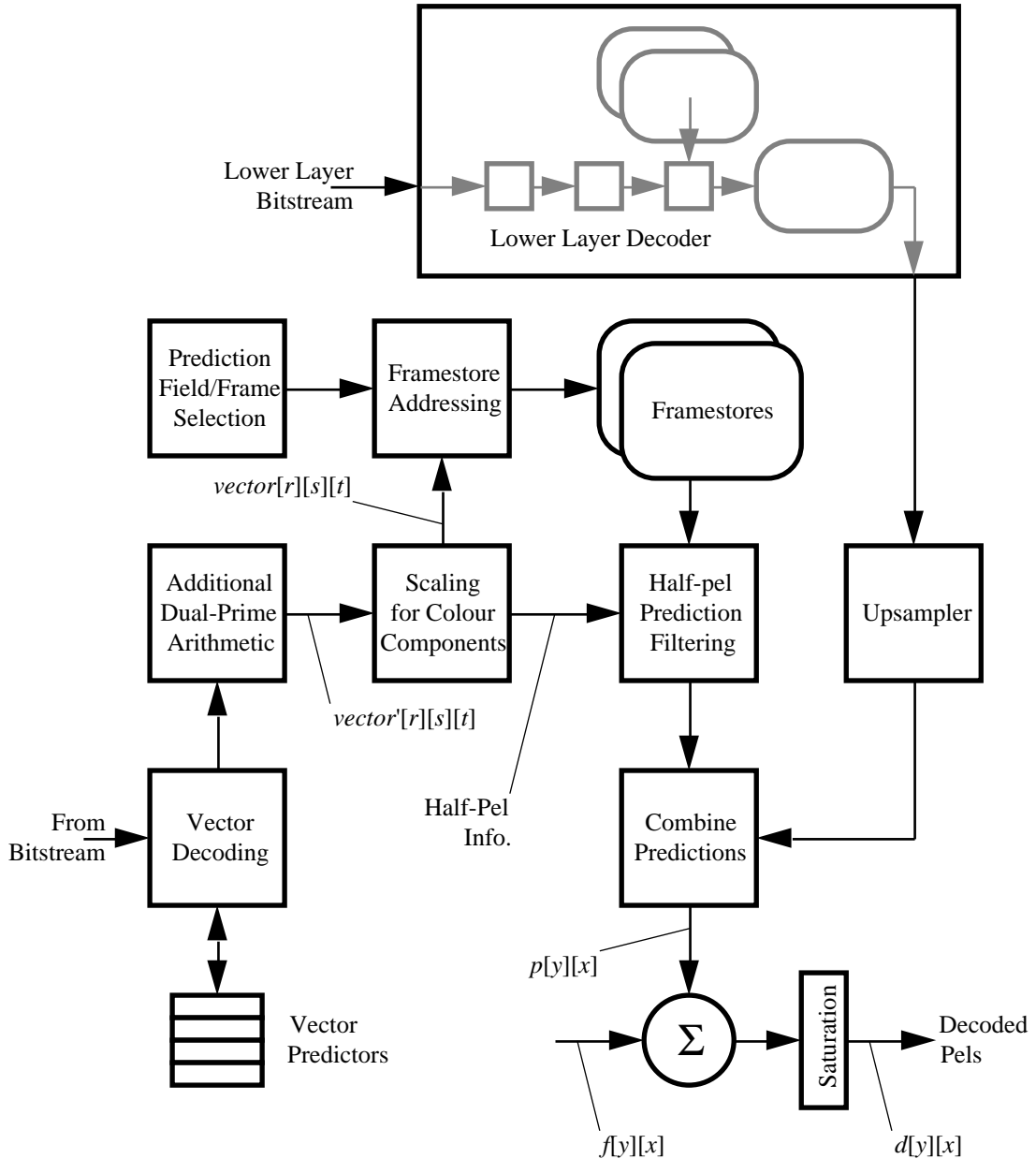
5 The prediction blocks have been formed and reorganised into blocks of prediction pels $p[y][x]$ which
6 match the field/frame structure used by the transform data blocks.

7 The transform data $f[y][x]$ shall be added to the prediction data and saturated to form the final decoded
8 pels $d[y][x]$ as follows;

```
9     for (y=0; y<8; y++) {  
10         for (x=0; x<8; x++) {  
11              $d[y][x] = f[y][x] + p[y][x];$   
12             if ( $d[y][x] < 0$ )  $d[y][x] = 0;$   
13             if ( $d[y][x] > 255$ )  $d[y][x] = 255;$   
14         }  
15     }
```

1 **7.7 Spatial Scalability**

2 This clause specifies the additional decoding process required for the spatial scalable extensions. Figure
 3 7-12 is a diagram of the video decoding process with spatial scalability The diagram is simplified for
 4 clarity.



5
 6

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

7 **7.7.1 Prediction in scalable layer**

8 A motion compensated 'temporal' prediction is made from previously decoded pictures in the
 9 enhancement layer as described in clause 7.6. In addition, a 'spatial' prediction is formed, which is an
 10 upsampled version of the lower layer decoded picture, as described in clause 7.7.2. These predictions
 11 are selected individually or combined to form the actual prediction.

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

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

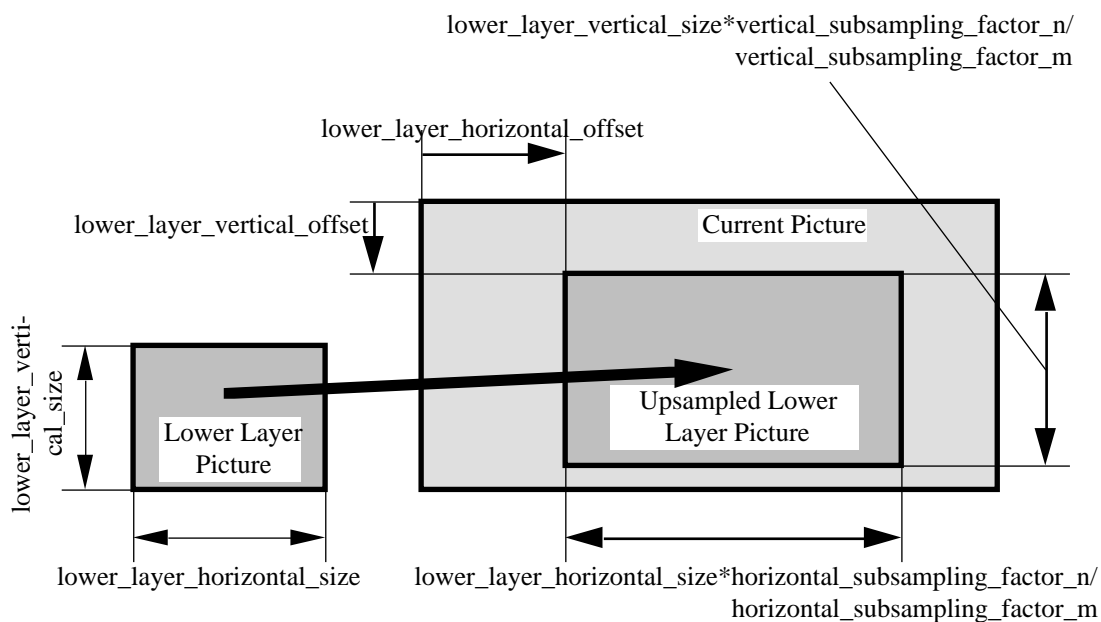
4 7.7.2 Formation of 'spatial' prediction

5 The spatial prediction is made from the decoded picture of the lower layer referenced by the
 6 lower_layer_temporal_reference. Note that spatial scalability will only work efficiently when
 7 predictions are formed from pictures in the lower layer which are coincident (in display time) with the
 8 predicted picture in the enhancement layer (or very close in time). In addition, in order to avoid
 9 additional frame memories in the decoder, predictions shall only be made from pictures in the lower
 10 layer that has most recently been decoded.

11 It is a requirement, on the bitstream, that a spatial prediction shall only be derived from the coincident
 12 decoded lower layer picture (or, if there is no appropriate coincident decoded picture, the most recently
 13 decoded picture). Timing information from the system layer shall be used (in addition to
 14 lower_layer_temporal_reference) to indicate the appropriate coincident, or most recently decoded,
 15 picture. If group_of_pictures_header() occurs often in the lower layer bitstream then the temporal
 16 reference in the lower layer may be ambiguous (because temporal_reference is reset after a
 17 group_of_pictures_header()). Thus without system layer timing information it may be difficult to align
 18 the two layers.

19 The prediction is made by scaling the lower layer to the same pel grid as the enhancement layer. A
 20 16x16 region is chosen, which corresponds to the position of the current macroblock which is being
 21 decoded.

22 This interpolation process is described in this section illustrated in Figure 7-13.



23

24 **Figure 7-13. Formation of the 'spatial' prediction by interpolation of the lower layer picture**

25 Spatial predictions shall only be made for macroblocks in the enhancement layer that lie wholly within
 26 the upsampled lower layer picture.

27 7.7.2.1 General

28 The process for resampling depends on whether the lower layer picture is interlaced or progressive, as
 29 indicated by lower_layer_progressive_frame and whether the enhancement layer picture is interlaced or
 30 progressive, as indicated by progressive_frame.

31 When lower_layer_progressive_frame is "1", the lower layer picture (input_prog_field) is resampled
 32 vertically as described in section 7.7.2.3. The resulting frame is considered to be progressive if
 33 progressive_frame is "1" and interlaced if progressive_frame is "0". The resulting frame is resampled

1 horizontally as described in section 7.7.2.4. Lower_layer_deinterlaced_field_select shall have the value
2 "1".

3 When lower_layer_progressive_frame is "0" and progressive_frame is "0", the lower layer picture is
4 deinterlaced as described in section 7.7.2.2, to produce two progressive fields (input_prog_field). Both
5 of these fields are resampled vertically, taking into account the lower_layer_vertical_offset, as
6 described in section 7.7.2.3. The resulting fields are subsampled, again taking into account the
7 lower_layer_vertical_offset, to produce two interlaced fields. The resulting frame is resampled
8 horizontally as described in section 7.7.2.4. Lower_layer_deinterlaced_field_select shall have the value
9 "1".

10 When lower_layer_progressive_frame is "0" and progressive_frame is "1", the lower layer picture is
11 deinterlaced as described in section 7.7.2.2, to produce two progressive fields (input_prog_field). Only
12 one of these fields is required. When lower_layer_deinterlaced_field_select is "0" the first is used,
13 otherwise the second is used. The one that is used is resampled vertically as described in section
14 7.7.2.3. The resulting frame is resampled horizontally as described in section 7.7.2.4.

15 The lower layer offsets are limited to even values when the chroma in the enhancement layer is
16 subsampled in that dimension in order that the chrominance pels are aligned between the two layers.

17 The upsampling process is summarised Table 7-16.

18 **Table 7-16 Upsampling process**

lower_layer_ field_select	lower_layer_ progressive_frame	progressive_ frame	spatial_temporal_ weight_code_ table_index	Apply deinterlace process	Entity used for prediction
0	0	1	00	yes	top field
1	0	1	00	yes	bottom field
1	1	1	00	no	frame
1	1	0	00,01,10,11	no	frame
1	0	0	00,01,10,11	yes	both fields

19 **7.7.2.2 Deinterlacing**

20 First, the lower layer frame is padded with zeros to form a progressive grid at a frame rate equal to the
21 field rate of the lower layer, and with the same number of lines and pels per line as the lower layer
22 frame. The luminance component is then filtered using the relevant two field aperture filter from the
23 table below. The chrominance component is then filtered using the relevant one field aperture filter
24 from the table below. The temporal and vertical columns of the table indicate the relative spatial and
25 temporal coordinates of the pels to which the filter taps defined in the other two columns apply. An
26 intermediate sum is formed by adding the multiplied coefficients together.

27 **Table 7-17. Deinterlacing Filter**

Temporal	Vertical	luminance		chrominance
		Filter for first field	Filter for second field	Filter (both fields)
-1	-2	0	-1	0
-1	0	0	2	0
-1	2	0	-1	0
0	-1	8	8	8
0	0	16	16	16
0	1	8	8	8
1	-2	-1	0	0
1	0	2	0	0
1	+2	-1	0	0

28

1 The output of the filter (sum) is then scaled according to the following formula:

$$2 \quad \text{prog_field}[y][x] = \text{sum} // 16$$

3 The filter aperture can extend outside the coded picture size. In this case the pels of the lines outside the
4 active picture shall take the value of the closest neighbouring existing pel (below or above) of the same
5 field as defined below.

6 For all pels [y][x]:

```
7     if (y<0 && (y&1 == 1))
8         y=1
9     if (y<0 && (y&1 == 0))
10        y=0
11    if (y >= lower_layer_prediction_vertical_size &&
12        (y-lower_layer_prediction_vertical_size)&1 == 1))
13        y = lower_layer_prediction_vertical_size - 1
14    if (y >= lower_layer_prediction_vertical_size &&
15        ((y-lower_layer_prediction_vertical_size)&1 == 0))
16        y = lower_layer_prediction_vertical_size - 2
```

17

18 **7.7.2.3 Vertical resampling**

19 The frame subject to vertical resampling, input_field, is resampled to the enhancement layer vertical
20 sampling grid using linear interpolation between the sample sites according to the following formula,
21 where output_field is the resulting field:

$$22 \quad \text{mid_field}[y_h + y_offset][x] = (16 - \text{phase}) * \text{input_prog_field}[y1][x] + \text{phase} * \text{input_prog_field}[y2][x]$$

23 where y_h = output sample co-ordinate in output field for the individual component

$$24 \quad y1 = (y_h * m) / n$$

$$25 \quad y2 = y1 + 1 \quad \text{if } y1 < \text{lower_layer_prediction_vertical_size} - 1$$

$$26 \quad y1 \quad \text{otherwise}$$

$$27 \quad \text{phase} = (16 * ((y_h * m) \% n)) // n$$

$$28 \quad m = \text{vertical_subsampling_factor_m}$$

$$29 \quad n = \text{vertical_subsampling_factor_n}$$

30 If different chroma formats are used in the two layers then “m” and “n” will need to be
31 modified for the chrominance samples: if the ratio of vertical chrominance subsampling
32 factor between enhancement and lower layers is $p \div q$, m and n become $m * p$ and $n * q$
33 respectively.

$$34 \quad y_offset = \text{lower_layer_vertical_offset} \text{ (scaled for the chrominance components in a} \\ 35 \quad \text{similar manner to the vectors - clause 7.6.3.7)}$$

36 Pels which lie outside the lower layer picture which are required for upsampling are obtained by border
37 extension of the lower layer picture.

38 **7.7.2.4 Horizontal resampling**

39 The frame subject to horizontal resampling, input_field, is resampled to the enhancement layer
40 horizontal sampling grid using linear interpolation between the sample sites according to the following
41 formula, where output_field is the resulting field:

$$42 \quad \text{output_field}[y][x_h + x_offset] = ((16 - \text{phase}) * \text{mid_field}[y][x1] + \text{phase} * \text{mid_field}[y][x2]) // 256$$

43 where x_h = output sample co-ordinate in output field for the individual component

$$44 \quad x1 = (x_h * m) / n$$

$$45 \quad x2 = x1 + 1 \quad \text{if } y1 < \text{lower_layer_prediction_horizontal_size} - 1$$

1 x1 otherwise
 2 phase = (16 * ((x_h * m) % n)) // n
 3 m = horizontal_subsampling_factor_m
 4 n = horizontal_subsampling_factor_n

5 If different chroma formats are used in the two layers then “m” and “n” will need to be
 6 modified for the chrominance samples: if the ratio of horizontal chrominance subsampling
 7 factor between enhancement and lower layers is p÷q, m and n become m*p and n*q
 8 respectively.

9 x_offset = lower_layer_horizontal_offset (scaled for the chrominance components in a
 10 similar manner to the vectors - clause 7.6.3.7)

11 Pels which lie outside the lower layer picture which are required for upsampling are obtained by border
 12 extension of the lower layer picture.

13 **7.7.2.5 Chroma formats**

14 It is permissible for the chroma formats in the lower layer and the enhancement layer to be different
 15 from one another. If the formats are different then the chrominance resampling process will be
 16 different to the luminance resampling.

17 **7.7.2.6 Generalised slice structure in the lower layer**

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

20 **7.7.3 Selection and combination of spatial and temporal predictions**

21 The spatial and temporal predictions can be selected or combined to form the actual prediction. The
 22 macroblock_type (tables B.2-5, B.2-6 and B.2-7) indicates, by use of the
 23 spatial_temporal_weight_class, which can take the values 0, 1, 2, 3, 4, whether the prediction is
 24 temporal-only, spatial-only or a weighted combination of temporal and spatial predictions. A fuller
 25 description of spatial_temporal_weight_class is given in Section 7.7.4.

26 In intra pictures, if spatial_temporal_weight_class is 0, normal intra coding is performed, otherwise the
 27 prediction is spatial-only. In predicted and interpolated pictures, if the spatial_temporal_weight_class is
 28 0, prediction is temporal-only, if the spatial_temporal_weight_class is 4, prediction is spatial-only,
 29 otherwise one or a pair of prediction weights is used to combine the spatial and temporal predictions.

30 The possible prediction weights are given in a weight table which is selected in the picture scalable
 31 extension. Up to four different weight tables are available for use depending on whether the current
 32 and lower layers are interlaced or progressive. In macroblock_modes(), a two bit code,
 33 spatial_temporal_weight_code, is used to describe the prediction for each field (or frame), as shown in
 34 the table below.

35

1 **Table 7-18. spatial_temporal_weights and spatial_temporal_weight_classes for the**
 2 **lower_layer_picture_formats and spatial_temporal_weight_codes**

spatial_temporal_weight_code_table_index	spatial_temporal_weight_code	spatial_temporal_weight (s)	spatial_temporal_weight_class	spatial_temporal_integer_weight
00*	-	(0.5)	1	0
01	00	(0, 1)	3	1
	01	(0, 0.5)	1	0
	10	(0.5, 1)	3	0
	11	(0.5, 0.5)	1	0
10	00	(1, 0)	2	1
	01	(0.5, 0)	1	0
	10	(1, 0.5)	2	0
	11	(0.5, 0.5)	1	0
11	00	(1, 0)	2	1
	01	(1, 0.5)	2	0
	10	(0.5, 1)	3	0
	11	(0.5, 0.5)	1	0
* For spatial_temporal_weight_code_table_index == 0 a spatial_temporal_weight of 1 (spatial_temporal_weight_class == 4) is signalled in the macroblock_type tables.				

3

4 When the prediction weight combination is given in the form (a, b), "a" gives the proportion of the
 5 prediction for the top field which is derived from the spatial prediction and "b" gives the proportion of
 6 the prediction for the bottom field which is derived from the spatial prediction for that field.

7 When the prediction weight combination is given in the form (a), "a" gives the proportion of the
 8 prediction for the frame which is derived from the spatial prediction for that frame. The precise
 9 method for predictor calculation is as follows:

10 If $pel_pred_temp[y][x]$ is used to denote the temporal prediction (formed within the enhancement
 11 layer) as defined for $pel_pred[y][x]$ in clause 7.6. $pel_pred_lower[y][x]$ is used to denote the prediction
 12 formed from the lower layer, then:

13 If the weight code is zero then no prediction is made from the lower layer. Therefore;

$$14 \quad pel_pred[y][x] = pel_pred_temp[y][x];$$

15 If the weight code is one then no prediction is made from the enhancement layer. Therefore;

$$16 \quad pel_pred[y][x] = pel_pred_lower[y][x];$$

17 If the weight code is one half then the prediction is the average of the temporal and spatial predictions.
 18 Therefore;

$$19 \quad pel_pred[y][x] = (pel_pred_temp[y][x] + pel_pred_lower[y][x])/2;$$

20 When chrominance is sampled 4:2:0, it is treated as interlaced, in accordance with the other prediction
 21 modes, that is, the first weight code is used for the top field chrominance lines and the second weight
 22 is used for the bottom field chrominance lines.

23 Note Each field has its own weight code.

24 It is intended that the different weight tables are used in the following circumstances (although this is
 25 not mandatory)

1

Table 7-19. Intended spatial_temporal_weight_code_table_index values

Lower layer format	Enhancement layer format	spatial_temporal_weight_code_table_index
Progressive or interlaced	Progressive	00
Progressive coincident with enhancement layer top fields	Interlaced	10
Progressive coincident with enhancement layer from bottom fields	Interlaced	01
Interlaced	Interlaced	00 or 11

2

3 **7.7.4 Updating motion vector predictors and Motion vector selection**

4 In frame pictures where field prediction is used the possibility exists that one of the fields is predicted
5 using spatial-only prediction. In this case no motion vector is present in the bitstream for the field
6 which has spatial-only prediction. For the case where both fields of a frame have spatial-only
7 prediction, the macroblock_type is such that no motion vectors are present in the bitstream for that
8 macroblock.

9 The class also indicates the number of motion vectors which are present in the coded bitstream and
10 how the prediction vectors are updated and this is described in the Table below. Classes are defined in
11 the following way:

12 Class 0 indicates temporal-only prediction

13 Class 1 indicates that neither field has spatial-only prediction

14 Class 2 indicates that the top field is spatial-only prediction

15 Class 3 indicates that the bottom field is spatial-only prediction

16 Class 4 indicates spatial-only prediction.

17 Updating of motion vector predictors is defined in tables 7-20 and 7-21.

1

Table 7-20. Updating of motion vector predictors in Field Pictures

frame_motion_type	macroblock_motion_forward				Predictors to update
	macroblock_motion_backward				
	macroblock_intra				
	spatial_temporal_weight_class				
Field-based [‡]	-	-	1	0	$PMV[1][0][1:0] = PMV[0][0][1:0]$ [◇]
Field-based	1	1	0	0	$PMV[1][0][1:0] = PMV[0][0][1:0]$ $PMV[1][1][1:0] = PMV[0][1][1:0]$
Field-based	1	0	0	0,1	$PMV[1][0][1:0] = PMV[0][0][1:0]$
Field-based	0	1	0	0,1	$PMV[1][1][1:0] = PMV[0][1][1:0]$
Field-based [‡]	0	0	0	0,1,4	$PMV[r][s][t] = 0$ §
16x8 MC	1	1	0	0	(none)
16x8 MC	1	0	0	0,1	(none)
16x8 MC	0	1	0	0,1	(none)
Dual prime	1	0	0	0	$PMV[1][0][1:0] = PMV[0][0][1:0]$

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

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

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

§ $PMV[r][s][t]$ is set to zero (for all r , s and t). See clause 7.6.3.4.

2

1

Table 7-21. Updating of motion vector predictors in Frame Pictures

frame_motion_type	macroblock_motion_forward				Predictors to update
	macroblock_motion_backward			spatial_temporal_weight_class	
	macroblock_intra		0		
	0				
	1				
Frame-based [‡]	-	-	1	0	$PMV[1][0][1:0] = PMV[0][0][1:0]$ [◇]
Frame-based	1	1	0	0	$PMV[1][0][1:0] = PMV[0][0][1:0]$ $PMV[1][1][1:0] = PMV[0][1][1:0]$
Frame-based	1	0	0	0,1,2,3	$PMV[1][0][1:0] = PMV[0][0][1:0]$
Frame-based	0	1	0	0,1,2,3	$PMV[1][1][1:0] = PMV[0][1][1:0]$
Frame-based [‡]	0	0	0	0,1,2,3,4	$PMV[r][s][t] = 0$ [§]
Field-based	1	1	0	0	(none)
Field-based	1	0	0	0,1	(none)
Field-based	1	0	0	2	$PMV[1][0][1:0] = PMV[0][0][1:0]$
Field-based	1	0	0	3	$PMV[1][0][1:0] = PMV[0][0][1:0]$
Field-based	0	1	0	0,1	(none)
Field-based	0	1	0	2	$PMV[1][1][1:0] = PMV[0][1][1:0]$
Field-based	0	1	0	3	$PMV[1][1][1:0] = PMV[0][1][1:0]$
Dual prime [@]	1	0	0	0,2,3	$PMV[1][0][1:0] = PMV[0][0][1:0]$

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

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

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

§ $PMV[r][s][t]$ is set to zero (for all r , s and t). See clause 7.6.3.4.

@ Dual prime can not be used when **spatial_temporal_integer_weight** = "0".

2

3 7.7.4.1 Resetting motion vector predictors

4 In addition to the cases identified in clause 7.6.3.4 the motion vector predictors shall be reset in the
5 following cases;

- 6 • In a P-picture when a macroblock is purely spatially predicted
7 (spatial_temporal_weight_class == 4)
- 8 • In a B-picture when a macroblock is purely spatially predicted
9 (spatial_temporal_weight_class == 4)

10 In case that spatial_temporal_weight_class == 2 in a frame picture when field-based prediction is used
11 only one vector is sent. This is vector[0][s][1:0] which is used for the bottom field. $PMV[1][s][1:0]$ is
12 then updated as shown in Table 7-21.

1

Table 7-22. Predictions and vectors in field pictures

field_motion_type	macroblock_motion_forward				Vector	Prediction formed for
	macroblock_motion_backward					
	macroblock_intra					
	spatial_temporal_weight_class					
Field-based [‡]	-	-	1	0	$vector'[0][0][1:0]$ [◇]	None (vector is for concealment)
Field-based	1	1	0	0	$vector'[0][0][1:0]$ $vector'[0][1][1:0]$	whole field, forward whole field, backward
Field-based	1	0	0	0,1	$vector'[0][0][1:0]$	whole field, forward
Field-based	0	1	0	0,1	$vector'[0][1][1:0]$	whole field, backward
Field-based [‡]	0	0	0	0,1,4	$vector'[0][0][1:0]$ ^{*§}	whole field, forward
16x8 MC	1	1	0	0	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$ $vector'[0][1][1:0]$ $vector'[1][1][1:0]$	upper 16x8 field, forward lower 16x8 field, forward upper 16x8 field, backward lower 16x8 field, backward
16x8 MC	1	0	0	0,1	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$	upper 16x8 field, forward lower 16x8 field, forward
16x8 MC	0	1	0	0,1	$vector'[0][1][1:0]$ $vector'[1][1][1:0]$	upper 16x8 field, backward lower 16x8 field, backward
Dual prime	1	0	0	0	$vector'[0][0][1:0]$ $vector'[2][0][1:0]$ ^{*†}	same parity whole field, forward opposite parity whole field, forward
Note:	Vectors are listed in the order they appear in the bitstream					
◇	the vector is only present if concealment_motion_vectors is one					
‡	field_motion_type is not present in the bitstream but is assumed to be Field-based					
*	These vectors are not present in the bitstream					
†	These vectors are derived from $vector'[0][0][1:0]$ as described in clause 7-?					
§	The vector is taken to be (zero, zero) as explained in clause 7-?					

2

3

1

Table 7-23. Predictions and vectors in frame pictures

frame_motion_type	macroblock_motion_forward				Vector	Prediction formed for
	macroblock_motion_backward					
	macroblock_intra					
	spatial_temporal_weight_class					
Frame-based [‡]	-	-	1	0	$vector'[0][0][1:0]$ [◇]	None (vector is for concealment)
Frame-based	1	1	0	0	$vector'[0][0][1:0]$ $vector'[0][1][1:0]$	frame, forward frame, backward
Frame-based	1	0	0	0,1,2,3	$vector'[0][0][1:0]$	frame, forward
Frame-based	0	1	0	0,1,2,3	$vector'[0][1][1:0]$	frame, backward
Frame-based [‡]	0	0	0	0,1,2,3,4	$vector'[0][0][1:0]$ ^{*§}	frame, forward
Field-based	1	1	0	0	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$ $vector'[0][1][1:0]$ $vector'[1][1][1:0]$	top field, forward bottom field, forward top field, backward bottom field, backward
Field-based	1	0	0	0,1	$vector'[0][0][1:0]$ $vector'[1][0][1:0]$	top field, forward bottom field, forward
Field-based	1	0	0	2	$vector'[0][0][1:0]$	top field, spatial bottom field, forward
Field-based	1	0	0	3	$vector'[0][0][1:0]$	top field, forward bottom field, spatial
Field-based	0	1	0	0,1	$vector'[0][1][1:0]$ $vector'[1][1][1:0]$	top field, backward bottom field, backward
Field-based	0	1	0	2	$vector'[0][1][1:0]$	top field, spatial bottom field, backward
Field-based	0	1	0	3	$vector'[0][1][1:0]$	top field, backward bottom field, spatial
Dual prime [@]	1	0	0	0,2,3	$vector'[0][0][1:0]$ $vector'[0][0][1:0]$ [*] $vector'[2][0][1:0]$ ^{*†} $vector'[3][0][1:0]$ ^{*†}	same parity top field, forward same parity bottom field, forward opposite parity top field, forward opposite parity bottom field, forward
Note:	<p>Vectors are listed in the order they appear in the bitstream</p> <p>◇ the vector is only present if concealment_motion_vectors is one</p> <p>‡ frame_motion_type is not present in the bitstream but is assumed to be Frame-based</p> <p>* These vectors are not present in the bitstream</p> <p>† These vectors are derived from $vector'[0][0][1:0]$ as described in clause 7-?</p> <p>§ The vector is taken to be (zero, zero) as explained in clause 7-?</p> <p>@ Dual prime can not be used when spatial_temporal_integer_weight = "0".</p>					

2

1 **7.7.5 Skipped macroblocks**

2 In all cases, a skipped macroblock is only the result of a prediction, and all the DCT coefficients are
3 considered to be zero.

4 The following rules apply for non-scalable bitstreams; i.e. if `sequence_scalable_extension` is not
5 present (or if `sequence_scalable_extension()` is present and `scalable_mode = "Data partitioning"`).

6 In I-frame pictures or field pictures, if `sequence_scalable_extension` is not present, then all
7 macroblocks are coded and there are no skipped macroblocks; i.e. The syntax element
8 `macroblock_address_increment` is always equal to "1", except for the first macroblock of a slice, in
9 which case it indicates the horizontal position of the slice.

10 In P frame pictures a skipped macroblock is defined to be a frame-based predicted macroblock with a
11 reconstructed frame motion vector equal to zero.

12 In P field pictures, a skipped macroblock is defined to be a field-based predicted macroblock with a
13 reconstructed field motion vector equal to zero. The reference field for the prediction is the same parity
14 field.

15 In B frame pictures, a skipped macroblock is defined to be a frame-based predicted macroblock with
16 differential frame motion vector(s) equal to zero. The type of prediction (forward, backward or
17 averaged) is the same as the prior macroblock.

18 In B field pictures, a skipped macroblock is defined to be a field-based predicted macroblock with
19 differential field motion vector(s) equal to zero. The type of prediction (forward, backward or
20 averaged) is the same as the prior macroblock. A reference field(s) for the prediction is (are) the same
21 parity field(s).

22 In B-frame or B-field pictures, a skipped macroblock shall not follow an intra-coded macroblock.

23

24 If `sequence_scalable_extension` is present and `scalable_mode = "spatial scalability"`, the following rules
25 apply in addition to those given above.

26 In I-frame or I-field pictures, skipped macroblocks are allowed. These are defined as spatial-only
27 predicted.

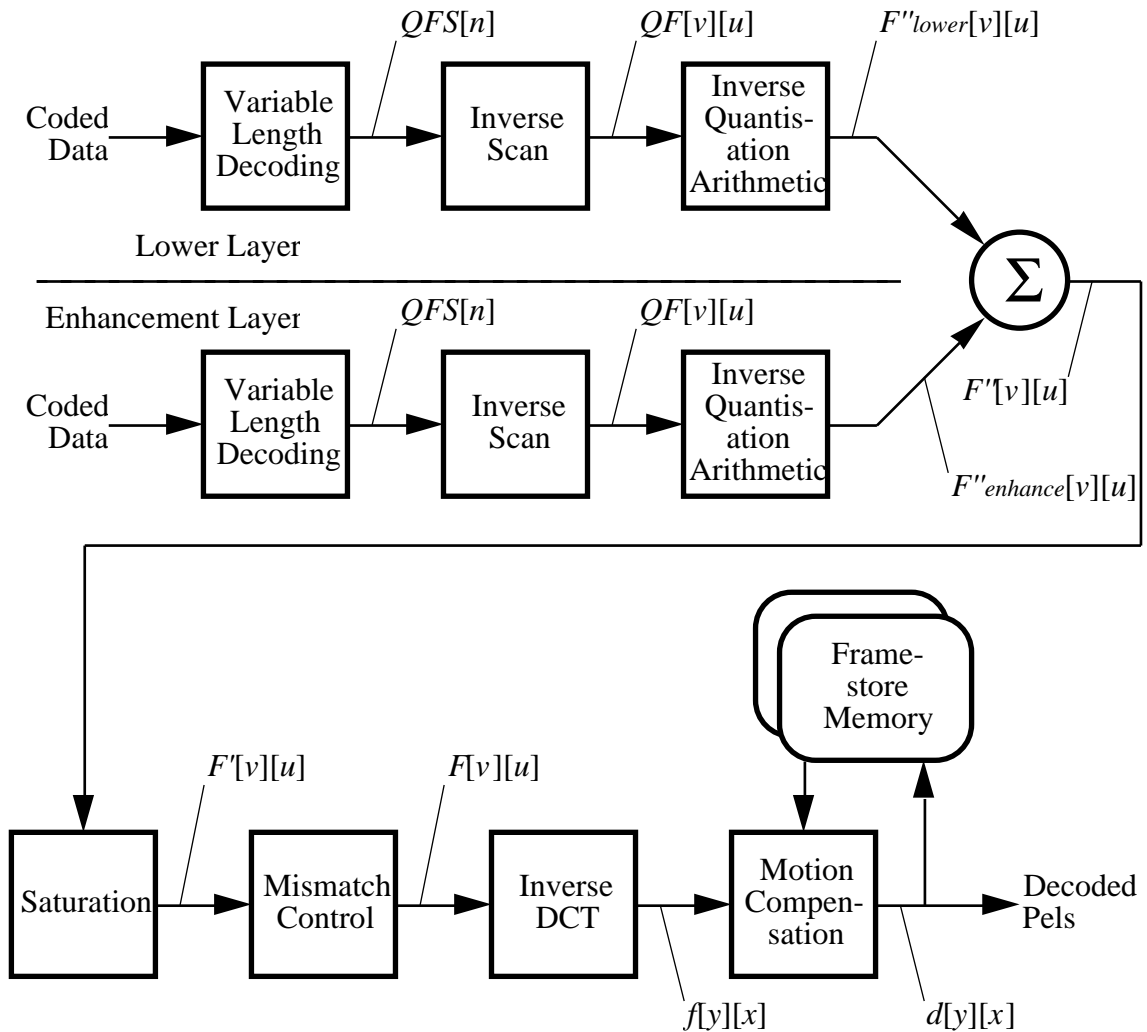
28 In P-pictures and B-pictures, the skipped macroblock is temporal-only predicted.

29 **7.7.6 Skipped pictures in the lower layer**

30 In the case of spatial scalability, skipped pictures in the lower layer may cause problems. This is
31 because of possible uncertainty in precisely which frames will be repeated by a particular decoder.

32 Hence the lower layer shall not use skipped pictures.

1 7.8 SNR scalability



2
3

4 **Figure 7-14. Illustration of decoding process for SNR scalability**

5

6 This clause describes the additional decoding process required for the SNR scalable extensions.

7 SNR scalability defines a mechanism to refine the DCT coefficients encoded in another layer of a
8 stream. As illustrated in Figure 7-? data from two bitstreams has to be combined after the inverse
9 quantisation processes by adding the DCT coefficients.

10 Clause 7.8.1 defines how to identify these bitstreams in a multilayer set of streams, however they can
11 be classified as follows.

12 The "lower layer", derived from the first bitstream, can itself be either non-scalable, or require the
13 spatial or temporal scalability decoding process to be applied.

14 The "enhancement" layer, derived from the second bitstream, contains mainly coded DCT coefficients
15 and a small overhead. The decoding process for this layer and the combination of the two layers are
16 described in this clause.

17 Note that all information regarding spatio-/temporal prediction is contained in the lower layer only.
18 Therefore it is not possible to reconstruct an enhancement layer without decoding the lower layer data
19 in parallel.

1 Furthermore prediction and reconstruction of the pictures as described in clauses 7.6, 7.7 and 7.9 for
2 the combined lower and enhancement layer is identical to the respective steps for decoding of the lower
3 layer only.

4 Semantics and decoding process described in this clause include a mechanism for “chroma simulcast”.
5 This may be used (for instance) to enhance a 4:2:0 signal in the lower layer to a 4:2:2 signal after
6 processing the enhancement layer data. While the luminance data is processed as described before, in
7 this case the chrominance information retrieved from the lower layer (with exception of intra-DC
8 values, see 7.8.3.4) shall be discarded and replaced by the new information with higher chrominance
9 resolution decoded from the enhancement layer.

10 It is inherent in SNR scalability that the two layers are very tightly coupled to one another. It is a
11 requirement that corresponding pictures in each layer shall be decoded at the same time as one another.

12 In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then two
13 different IDCT mismatch control schemes are being used in decoding. Care must be taken in the
14 encoder to take account of this.

15 **7.8.1 Higher syntactic structures**

16 The two bitstreams layers in this clause are identified by their `layer_id`, decoded from the
17 `sequence_scalable_extension`.

18 The two bitstreams shall have consecutive layer ids, with enhancement layer having
19 `layer_id = idenhance` and the lower layer having `layer_id = idenhance-1`.

20 The syntax and semantics of the enhancement layer are as defined in clauses 6.2 and 6.3, respectively.

21 Semantic restrictions apply to several values in the headers and extensions of the enhancement layer as
22 follows

23 In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then both this
24 lower and the enhancement layer shall use the “restricted slice structure” defined in this specification.

25

26 **Sequence header**

27 This header shall be identical to the one in the lower layer except for the values of `bit_rate`,
28 `vbv_buffer_size`, `load_intra_quantiser_matrix`, `intra_quantiser_matrix`,
29 `load_non_intra_quantiser_matrix` and `non_intra_quantiser_matrix`. These can be selected independently
30 except for `load_intra_quantiser_matrix` which shall be zero.

31 **Sequence extension**

32 This extension shall be identical to the one in the lower layer except for the values of
33 `profile_and_level_indication`, `chroma_format`, `bit_rate_extension` and `vbv_buffer_size_extension`.
34 Those can be selected independently.

35 A different value of `chroma_format` in each layer will cause the `chroma_simulcast` flag to be set as
36 specified by Table 7-?.

37 Note that the `chroma_format` of the enhancement layer shall be higher or equal to the `chroma_format` of
38 the lower layer.

1

Table 7-24 chroma_simulcast flag

chroma_format (lower layer)	chroma_format (enhancement layer)	chroma_simulcast
4:2:0	4:2:0	0
4:2:0	4:2:2	1
4:2:0	4:4:4	1
4:2:2	4:2:2	0
4:2:2	4:4:4	1
4:4:4	4:4:4	0

2

3 In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification),
 4 sequence_extension() is not present in the lower layer, and the following values shall be assumed for
 5 the decoding process.

6 progressive_sequence = 1
 7 chroma_format = "4:2:0"
 8 horizontal_size_extension = 0
 9 vertical_size_extension = 0
 10 bit_rate_extension = 0
 11 vbv_buffer_size_extension = 0
 12 low_delay = 0
 13 frame_rate_extension_n = 0
 14 frame_rate_extension_d = 0

15

16 The sequence_extension() in the enhancement layer shall have the values shown above.

17 Sequence display extension

18 This extension shall not be present as there is no separate display process for the enhancement layer.

19 Sequence scalable extension

20 This extension shall be present with scalable_mode = "SNR scalable".

21 GOP header

22 This header (if present) shall be identical to the one in the lower layer.

23 Picture header

24 This header shall be identical to the one in the lower layer except for the value of vbv_delay. This can
 25 be selected independently.

26 Picture coding extension

27 This extension shall be identical to the one in the lower layer except for the value of q_scale_type and
 28 alternate_scan. These can be selected independently.

29 chroma_420_type shall be set to "0" if chroma_simulcast is set. Else it shall have the same value as in
 30 the lower layer.

1 In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then
 2 picture_coding_extension() is not present in the lower layer and the following values shall be assumed
 3 for the decoding process:

4	forward_horizontal_f_code	=	forward_f_code in the lower layer or 15
5	forward_vertical_f_code	=	forward_f_code in the lower layer or 15
6	backward_horizontal_f_code	=	backward_f_code in the lower layer or 15
7	backward_vertical_f_code	=	backward_f_code in the lower layer or 15
8	intra_dc_precision	=	0
9	picture_structure	=	“Frame Picture”
10	top_field_first	=	0
11	frame_pred_frame_dct	=	1
12	concealment_motion_vectors	=	0
13	intra_vlc_format	=	0
14	repeat_first_field	=	0
15	chroma_420_type	=	1
16	progressive_frame	=	1
17	composite_display_flag	=	0

18 The picture_coding_extension() in the enhancement layer shall have the values shown above.

19 For the lower layer q_scale_type and alternate_scan shall be assumed to have the value zero.

20 Note q_scale_type and alternate_scan can be set independently in the enhancement layer.

21 **Quant matrix extension**

22 This extension is optional. Semantics are described in clause 6.3.9.

23 Note that only the non-intra matrices will be used in the subsequent decoding process.

24 load_intra_quantiser_matrix and load_chroma_intra_quantiser_matrix shall both be zero.

25 **Pan-scan extension**

26 This extension shall not be present.

27 Note: There is no separate display process for the enhancement layer. If pan-scan functionality
 28 is desired it can be accomplished already by using the information conveyed by the pan-
 29 scan extension of the lower layer.

30 **Slice header**

31 Slices shall be coincident with those in the lower layer. The value of quantiser_scale_code can be set
 32 independently from the lower layer.

33 **7.8.2 Macroblock**

34 Subsequently the "current macroblock" denotes the currently processed macroblock. The "current
 35 macroblock of the lower layer" denotes the macroblock identified by having the same
 36 macroblock_address as the current macroblock.

37 The decoding of the macroblock header information is done according to semantics in clause 6.3.14.

38 Note: Table B-8 which is used if scalable_mode == "SNR scalability" will never set the
 39 macroblock_intra, macroblock_motion_forward or macroblock_motion_backward flags,
 40 since a macroblock in the enhancement layer contains only refinement data for the current
 41 macroblock of the lower layer.

42 However the corresponding syntax elements and flags of the current macroblock in the
 43 lower layer are relevant for the combined decoding process of lower and enhancement
 44 layer following the inverse DCT as detailed below.

45 **7.8.2.1 dct_type**

46 The syntax element dct_type may be present in none, one or both of the lower and enhancement layer
 47 macroblock_modes(), as indicated by the semantics in clause 6.3.15.

48 If dct_type is present in the macroblock_modes() in both layers it shall have identical values.

1 **7.8.2.2 Skipped Macroblocks**

2 Macroblocks can be skipped in the enhancement layer, meaning that no coefficient enhancement is
3 done. In that case the decoding process is exactly as specified in clause 7.

4 Macroblocks can also be skipped in the lower layer, while still being coded in the enhancement layer.
5 In that case the decoding process detailed in the following has to be applied.

6 **7.8.3 Block**

7 The first part of the decoding process of the enhancement layer block is independent from the lower
8 layer.

9 The second part of the decoding process of the enhancement layer block has to be done jointly with the
10 decoding process of the coincident lower layer block.

11 Two sets of inverse quantised coefficients F''_{lower} and $F''_{enhance}$ are added to form F'' (see Figure
12 7.?).

13 F''_{lower} is derived from the lower layer exactly as defined in clauses 7.1 to 7.4.2.3.

14 $F''_{enhance}$ is derived as is defined in the sub-clauses below.

15 The resulting signal F'' is further processed, starting with saturation, as defined in clauses 7.4.4 to 7.6
16 (7.7, 7.9)

17 **7.8.3.1 VLC decoding**

18 In an enhancement-layer block the VLC decoding shall be performed according to clause 7.2., as for a
19 non-intra block (as indicated by `macroblock_intra = 0`).

20 **7.8.3.2 Inverse scan**

21 Inverse scan shall be done exactly as defined in clause 7.3

22 **7.8.3.3 Inverse quantisation**

23 In an enhancement-layer block the inverse quantisation shall be performed according to clause 7.4.2 as
24 for a non-intra block.

25 In the case that the lower layer conforms to ISO/IEC 11172-2 (and not this specification) then the
26 “inverse quantisation arithmetic” used to derive $F''_{lower}[v][u]$ (see Figure 7-14) shall include the
27 IDCT mismatch control (oddification) and saturation specified in ISO/IEC 11172-2.

28 **7.8.3.4 Addition of coefficients from the two layers**

29 Corresponding coefficients from the blocks of each layer shall be added together to form the signal F''
30 (see Figure 7.?).

31
$$F''[v][u] = F''_{lower}[v][u] + F''_{enhance}[v][u], \text{ for all } u, v$$

32 If `chroma_simulcast = 1` is set only the luminance blocks are treated as described above.

33 For chrominance blocks the DC coefficient of the base layer is used as a prediction of the DC
34 coefficient in the coincident block in the enhancement layer, whereas the AC coefficients of the base
35 layer are discarded and AC coefficients of the enhancement layer form the signal F'' in Figure 7-?
36 according to the following formulae:

37
$$F''[0][0] = F''_{lower}[0][0] + F''_{enhance}[0][0]$$

38
$$F''[v][u] = F''_{enhance}[v][u], \text{ for all } u, v \text{ except } u = v = 0$$

39 Note Chroma simulcast blocks are inverse quantised like non-intra blocks and use the
40 chrominance non-intra matrix.

41 Table 7-25 gives the index of the chrominance block whose DC coefficient ($F''_{lower}[0][0]$) is to be
42 used to predict the DC coefficient in the coincident chrominance block of the enhancement layer
43 ($F''_{enhance}[0][0]$).

1

Table 7-25. block index

chroma_format	block index							
	4	5	6	7	8	9	10	11
base: 4:2:0 upper: 4:2:2	4	5	4	5	-	-	-	-
base: 4:2:0 upper: 4:4:4	4	5	4	5	4	5	4	5
base: 4:2:2 upper: 4:4:4	4	5	6	7	4	5	6	7

2

3 **7.8.3.5 Remaining macroblock decoding steps**

4 After addition of coefficients from the two layers, the remainder of the macroblock decoding steps is
 5 exactly as described in clauses 7.4.4 to 7.6 (7.7, 7.9, if applicable), since there is now only one data
 6 stream $F''[v][u]$ to be processed.

7 In this process, the spatio/temporal prediction signal $p[y][x]$ is derived according to the macroblock
 8 type syntax elements and flags for the current macroblock known from the lower layer.

9

1 7.9 Temporal scalability

2 Temporal scalability involves two layers, a lower layer and an enhancement layer. Both the lower and
3 the enhancement layers process the same spatial resolution. The enhancement layer enhances the
4 temporal resolution of the lower layer and if temporally remultiplexed with the lower layer signal
5 provides full temporal rate. This is the frame rate indicated in the enhancement layer. The decoding
6 process for enhancement layer pictures is similar to the normal decoding process described in clauses
7 7.1 to 7.6. The only difference is in the "Prediction field and frame selection" described in 7.6.2.

8 The reference pictures for prediction are selected by `reference_select_code` as described in Tables 7-26
9 and 7-27. In P pictures, the forward reference picture can be one of the following three: most recent
10 enhancement picture, most recent lower layer frame, or next lower layer frame in display order. Note
11 that in the latter case, the reference frame in lower layer used for prediction is backward in time.

12 In B-pictures, the forward reference can be one of the following two: most recent the enhancement
13 pictures or most recent (or temporally coincident) lower layer frame whereas the backward reference
14 can be one of the following two: most recent lower layer picture including temporally coincident
15 picture in display order or next lower layer frame in display order. Note that in this case, the backward
16 reference frame in lower layer used for prediction is forward in time.

17 Backward prediction cannot be made from a picture in the enhancement layer. This avoids the need for
18 frame reordering in the enhancement layer. Motion compensation process forms predictions using
19 lower layer decoded pictures and/or previous temporal prediction from the enhancement layer.

20 The enhancement layer can contain I-, P- or B-pictures, but B-pictures in enhancement layer behave
21 more like P-pictures in the sense that a decoded B-picture can be used to predict the following P- or B-
22 pictures in the enhancement layer.

23 When the most recent frame in the lower layer is used as the reference, this includes the frame that is
24 temporally coincident with the frame or the first field (in case of field pictures) in the enhancement
25 layer. The prediction references used for P- and B- pictures are shown in Table 7-26 and Table 7-27
26 respectively.

27 The lower and enhancement layers shall use a restricted slice structure.

28 **Table 7-26 Prediction references selection in P-pictures**

<code>reference_select_code</code>	forward prediction reference
00	Most recent decoded enhancement picture(s)
01	Most recent lower layer frame in display order
10	Next lower layer frame in display order
11	forbidden

29

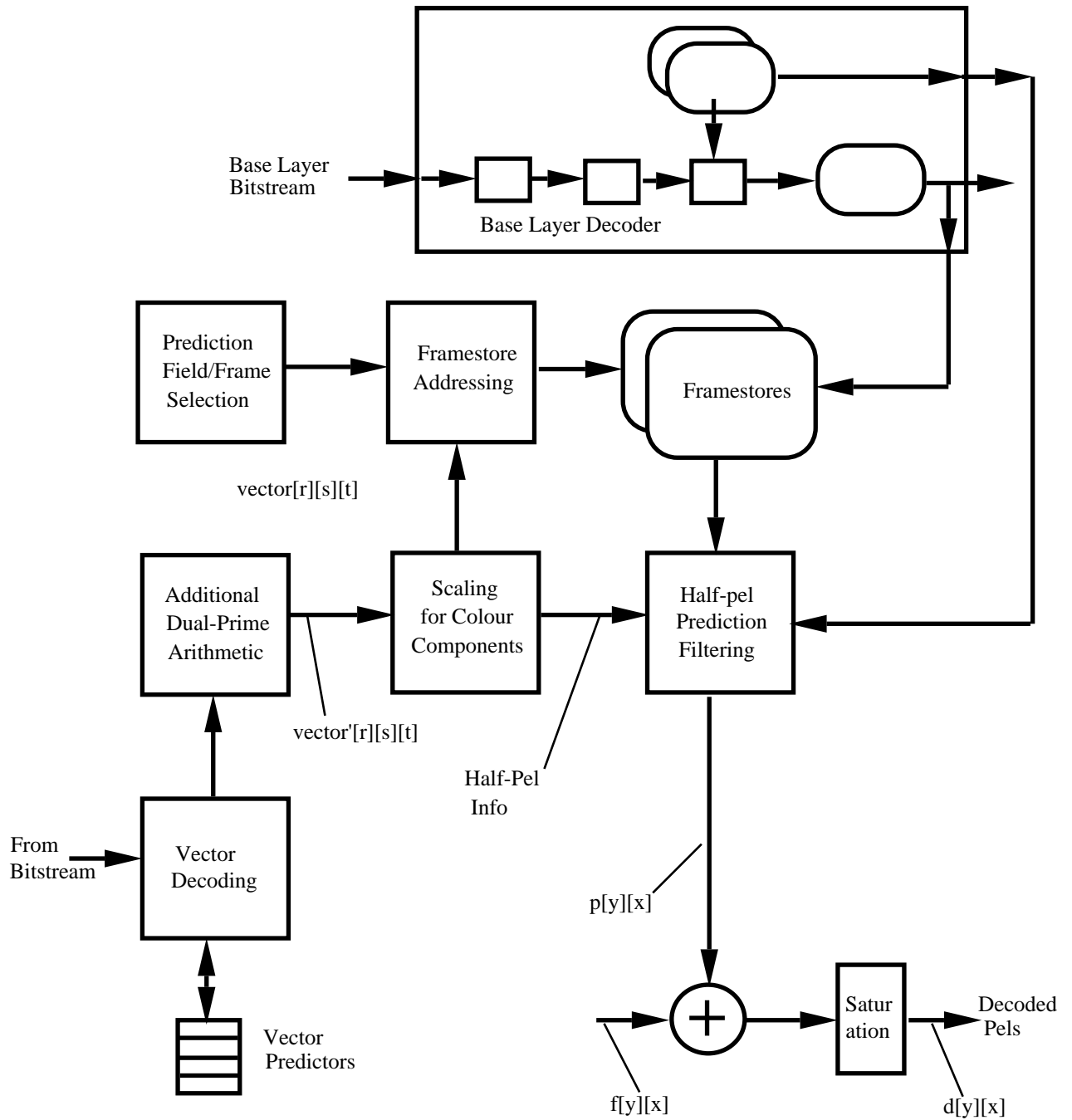
30 **Table 7-27 Prediction references selection in B-pictures**

<code>reference_select_code</code>	forward prediction reference	backward prediction reference
00	forbidden	forbidden
01	Most recent decoded enhancement picture(s)	Most recent lower layer picture in display order
10	Most recent decoded enhancement picture(s)	Next lower layer picture in display order
11	Most recent lower layer picture in display order	Next lower layer picture in display order

31

32 Figure 7.15 shows a simplified diagram of the decoding process for temporal scalability.

1



2
3
4

Figure 7-15 Decoding process for temporal scalability

1 **7.10 Data Partitioning**

2 Data partitioning is a technique that splits a video bitstream into two layers, called partitions. A priority
 3 breakpoint indicates which syntax elements are placed in partition 0, which is the base partition (also
 4 called high priority partition). The remainder of the bitstream is placed in partition 1 (which is also
 5 called low priority partition). Sequence, GOP, and picture headers are redundantly copied in partition 1
 6 to facilitate error recovery. All fields in the redundant headers must be identical to the original ones.
 7 The only extensions allowed (and required) in partition 1 are `sequence_extension()`,
 8 `picture_coding_extension()` and `sequence_scalable_extension()`.

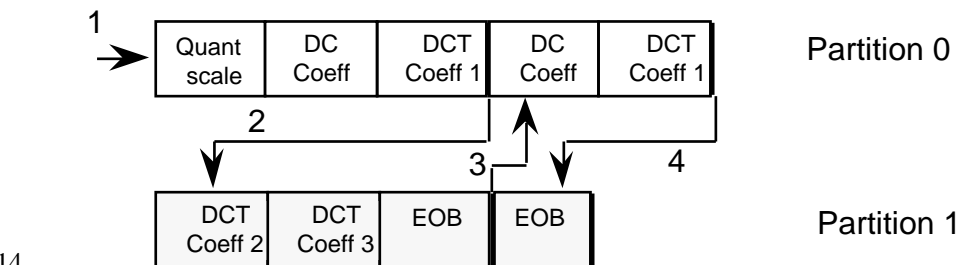
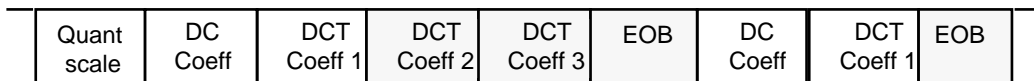
9 Note that the `slice()` syntax given in clause 6.2.4 is followed in both partitions up to (an including) the
 10 syntax element `extra_bit_slice`.

11 The interpretation of `priority_breakpoint` is given in Table 7-28.

12 **Table 7-28 Priority breakpoint values and associated semantics**

priority_break point	Syntax elements included in partition zero
0	This value is reserved for partition 1. All slices in partition 1 must have a <code>priority_breakpoint</code> equal to 0.
1	All data at the sequence, GOP, picture and <code>slice()</code> down to <code>extra_bit_slice</code> in <code>slice()</code> .
2	All data included above, plus macroblock syntax elements up to and including <code>macroblock_address_increment</code> .
3	All data included above, plus macroblock syntax elements up to but not including <code>coded_block_pattern()</code> .
4 ... 63	Reserved.
64	All syntax elements up to and including <code>coded_block_pattern()</code> or DC coefficient (<code>dct_dc_differential</code>), and the first (run, level) DCT coefficient pair (or EOB). [†]
65	All syntax elements above, plus up to 2 (run, level) DCT coefficient pairs.
...	
63+j	All syntax elements above, plus up to <i>j</i> (run, level) DCT coefficient pairs.
...	
127	All syntax elements above, plus up to 64 (run, level) DCT coefficient pairs.

13



[†] Note that a `priority_breakpoint` immediately following the DC coefficient is disallowed since it might cause start code emulation.

1 **Figure 7-16 A segment from a bitstream with two partitions, with *priority_breakpoint* set to 64**
 2 **(one (run, level) pair). The two partitions are shown, with arrows indicating how the decoder**
 3 **needs to switch between partitions.**

4
5

6 Semantics of VBV remains unchanged, i.e. the VBV refers to the sum of two partitions, not any single
7 one.

8 The decoding process is modified in the following manner:

9 Set *current_partition* to 0, and start decoding from bitstream that contains the
10 sequence_scalable_extension (partition 0).

11 If *current_partition* = 0, check to see if the current point in the bitstream is a priority
12 breakpoint.

13 If yes, set *current_partition* to 1. Next item will be decoded from partition. 1

14 Otherwise, continue decoding from partition 0. Remove sequence, GOP, and picture
15 headers from both partitions.

16 If *current_partition* = 1, check the priority breakpoint to see if the next item to be decoded is
17 expected in partition 0.

18 If yes, set *current_partition* to 0. Next item will be decoded from partition 0.

19 Otherwise, continue decoding from partition 1.

20 An example is shown in Figure 7-16 where the priority breakpoint is set at 64 (one (run, level) pair).

21 **7.11 Hybrid scalability**

22 Hybrid scalability is the combination of two different types of scalability. The types of scalability that
23 can be combined are SNR scalability, spatial scalability and temporal scalability. When two types of
24 scalability are combined, there are three bitstreams that have to be decoded. The layers to which these
25 bitstreams belong are named in Table 7-29.

26

Table 7-29 Names of layers

layer_id	name
0	base layer
1	enhancement layer 1
2	enhancement layer 2
...	...

27

28 For the scalability between the enhancement layers 1 and 2, the enhancement layer 1 is its base layer,
29 and the enhancement layer 2 is its enhancement layer. No layer can be omitted from the hierarchical
30 ladder. E.g., if there is SNR scalability between enhancement layer 1 and enhancement layer 2, the
31 prediction types in layer 2 are exactly the same as in enhancement layer 1, which is the base layer for
32 SNR scalability and which carries the prediction type.

33 The coupling of layers is more loose with spatial and temporal scalability than with SNR scalability.
34 Therefore, in these kinds of scalability, first the base layer has to be decoded and upconverted before it
35 can be used in the enhancement layer. In SNR scalability, both layers are decoded simultaneously.
36 The decoding order can be summarised as follows :

37

- 1 case 1 :
- 2 base layer
- 3 <*spatial or temporal scalability*>
- 4 enhancement layer 1
- 5 <*SNR scalability*>
- 6 enhancement layer 2
- 7
- 8 First decode the base layer, and then decode both enhancement layers simultaneously.
- 9
- 10 case 2 :
- 11 base layer
- 12 <*SNR scalability*>
- 13 enhancement layer 1
- 14 <*spatial or temporal scalability*>
- 15 enhancement layer 2
- 16
- 17 First decode the base layer and the enhancement layer 1 simultaneously, and then decode the
- 18 enhancement layer 2.
- 19
- 20 case 3 :
- 21 base layer
- 22 <*spatial or temporal scalability*>
- 23 enhancement layer 1
- 24 <*spatial or temporal scalability*>
- 25 enhancement layer 2
- 26
- 27 First decode the base layer, then decode the enhancement layer 1, and finally decode enhancement
- 28 layer 2.

1 8 Profiles and levels

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

5 Profiles and levels provide a means of defining subsets of the syntax and semantics of this specification
6 and thereby the decoder capabilities required to decode a particular bitstream. A profile is a defined
7 sub-set of the entire bitstream syntax that is defined by this specification. A level is a defined set of
8 constraints imposed on parameters in the bitstream. Conformance tests will be carried out against
9 defined profiles at defined levels.

10 The purpose of defining conformance points in the form of profiles and levels is to facilitate bitstream
11 interchange among different applications. Implementers of this specification are encouraged to
12 produce decoders and bitstreams which correspond to those defined conformance regions. The
13 discretely defined profiles and levels are the means of bitstream interchange between applications of
14 this specification.

15 In subsequent clauses the constrained parts of the defined profiles and levels will be described. All
16 syntactic elements and parameter values which are not explicitly constrained may take any of the
17 possible values that are allowed by this specification. A decoder shall be deemed to be conformant to a
18 given profile at a given level if it is able to properly decode all allowed values of all syntactic elements
19 as specified by that profile at that level. A bitstream shall be deemed to be conformant if it does not
20 exceed the allowed range of allowed values and does not include disallowed syntactic elements.

21 Attention is drawn to clause 5.5 which defines the convention for specifying a range of numbers. This
22 is used throughout to specify the constrained range of values and parameters.

23 The `profile_and_level_indication` in the `sequence_extension` indicates the profile and level to which the
24 bitstream complies. The meaning of the bits in this parameter is defined in Table 8-1.

25 **Table 8-1. Meaning of bits in `profile_and_level_indication`.**

Bits	Field Size (bits)	Meaning
[7:7]	1	Escape bit
[6:4]	3	Profile identification
[3:0]	4	Level identification

26

27 Table 8-2 specifies the profile identification codes and Table 8-3 the level identification codes. When
28 the escape bit equals zero a profile with a numerically larger identification value will be a subset of a
29 profile with a numerically smaller identification value. Similarly, whenever the escape bit equals zero,
30 a level with a numerically larger identification value will be a subset of a level with a numerically
31 smaller identification value.

1

Table 8-2. Profile identification.

Profile identification	Profile
110 to 111	(reserved)
101	Simple
100	Main
011	SNR Scalable
010	Spatially Scalable
001	High
000	(reserved)

2

Table 8-3. Level identification.

Level identification	Level
1011 to 1111	(reserved)
1010	Low
1001	(reserved)
1000	Main
0111	(reserved)
0110	High 1440
0101	(reserved)
0100	High
0000 to 0011	(reserved)

3

4 All decoders which can decode bitstreams which comply with the Simple, Main, SNR Scalable,
5 Spatially Scalable, or High profiles, shall be able to decode bitstreams which comply with the
6 "constrained parameter set" of ISO/IEC 11172-2. Additionally, these decoders shall be able to decode
7 D-pictures-only bitstreams of ISO/IEC 11172-2 which are within the level constraints of the decoder.

8 Table 8-4 describes profiles and levels when the escape bit equals 1. For these profiles and levels there
9 is no implied hierarchy from the assignment of profile_and_level_indication and profiles and levels are
10 not necessarily subsets of others.

11

Table 8-4. Escape profile_and_level_indication identification.

profile_and_level_indication	Name
10000000 to 11111111	(reserved)

12

8.1 Simple profile

13 Bitstreams that are compliant with the Simple profile shall meet the restrictions set out in clause 8.1.1.

14 In addition, bitstreams that are compliant with the defined levels of the Simple profile shall meet the
15 restrictions set out in the corresponding clause.

Defined level	Clause specifying restrictions
Main	8.1.2

16

8.1.1 Simple profile syntax

17 Bitstreams complying to the simple profile shall conform to the restrictions in clause 8.2.1 with the
18 single additional restriction that:

19

8.1.1.1 Picture coding type

20 The picture_coding_type shall not take the value "Bidirectionally predictive coding".

1 **8.1.1.2 Chroma sampling structure**

2 The chroma sampling structure (defined by chroma_format) shall be 4:2:0.

3 **8.1.1.3 Scalability**

4 The following extensions shall not be present in the bitstream;

- 5 • sequence_scalable_extension()
- 6 • picture_spatial_scalable_extension()
- 7 • picture_temporal_extension()

8 **8.1.1.4 Slice structure**

9 The “restricted slice structure” defined in clause 6.1.5.2 shall be used.

10 **8.1.2 Main level**

11 **8.1.2.1 Frame dimensions**

12 The horizontal size of the frame (defined by horizontal_size_value and horizontal_size_extension) shall
13 be in the range <0:720] pel.

14 The vertical size of the frame (defined by vertical_size_value and vertical_size_extension) shall be in
15 the range <0:576] pels. In the case of interlaced pictures this constraint refers to the number of lines in
16 the frame; there will be half this number in each field.

17 The frame rate (defined by frame_rate) shall be in the range <0:30] frames per second (and is
18 constrained to the discrete values that frame_rate can represent). In addition the product of the
19 horizontal size, the vertical size and the frame rate shall be limited to lie in the range <0:10 368 000]
20 pels per second. The frame_rate_extension value shall be zero.

21 **8.1.2.2 Coded data rate and VBV buffer size**

22 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
23 R_{max} for variable bit rate operation, both of which are specified in bit_rate, shall be limited to the
24 range <0:15 000 000] bits per second.

25 The ratio of bit rate specified in bit_rate to frame rate specified in frame_rate shall lie in the range
26 <0:626] kbits.

27 The VBV buffer size (vbv_buffer_size) shall be limited to the range <0:1 835 008] bits.

28 **8.1.2.3 Vector range**

29 The reconstructed vertical motion vectors shall be limited to the range [-128:+127.5] in frame pictures.
30 The reconstructed vertical motion vectors shall be limited to the range [-64:+63.5] in field pictures.
31 Note that this rule applies to the final reconstructed motion vector after scaling (for dual-prime) has
32 been performed.

33 The values of forward_horizontal_f_code and backward_horizontal_f_code shall be limited to the
34 range [1:8].

35 The values of forward_vertical_f_code and backward_vertical_f_code shall be limited to the range
36 [1:5].

37 **8.1.2.4 intra_dc_precision**

38 The intra_dc_precision shall be 8 bit, 9 bit or 10 bit.

39 **8.2 Main profile**

40 Bitstreams that are compliant with the Main profile shall meet the restrictions set out in clause 8.2.1.

41 In addition bitstreams that are compliant with the defined levels of the Main profile shall meet the
42 restrictions set out in the corresponding clause.

Defined level	Clause specifying restrictions
Low	8.2.2
Main	8.2.3
High-1440	8.2.4
High	8.2.5

- 1 **8.2.1 Main profile syntax**
- 2 **8.2.1.1 Chroma sampling structure**
- 3 The chroma sampling structure (defined by `chroma_format`) shall be 4:2:0.
- 4 **8.2.1.2 Scalability**
- 5 The following extensions shall not be present in the bitstream:
- 6 • `sequence_scalable_extension()`
- 7 • `picture_spatial_scalable_extension()`
- 8 • `picture_temporal_extension()`.
- 9 **8.2.1.3 Slice structure**
- 10 The “restricted slice structure” defined in clause 6.1.5.2 shall be used.
- 11 **8.2.2 Low level**
- 12 **8.2.2.1 Frame dimensions**
- 13 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
- 14 be in the range $<0:352]$ pels.
- 15 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
- 16 the range $<0:288]$ pels.
- 17 The frame rate (defined by `frame_rate`) shall be in the range $<0:30]$ frames per second (and is
- 18 constrained to the discrete values that `frame_rate` can represent). In addition the product of the
- 19 horizontal size, the vertical size and the frame rate shall be limited to lie in the range $<0:3\ 041\ 280]$
- 20 pels per second. The `frame_rate_extension` value shall be zero.
- 21 **8.2.2.2 Coded data rate and VBV buffer size**
- 22 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
- 23 R_{max} for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the
- 24 range $<0:4\ 000\ 000]$ bits per second.
- 25 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range
- 26 $<0:167]$ kbits.
- 27 The VBV buffer size (`vbv_buffer_size`) shall be limited to the range $<0:489\ 472]$ bits.
- 28 **8.2.2.3 Vector range**
- 29 The reconstructed vertical motion vectors shall be limited to the range $[-128:+127.5]$ in frame pictures.
- 30 **8.2.2.4 intra_dc_precision**
- 31 The `intra_dc_precision` is shall be 8 bit, 9 bit or 10 bit.
- 32 **8.2.3 Main level**
- 33 **8.2.3.1 Frame dimensions**
- 34 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
- 35 be in the range $<0:720]$ pel.
- 36 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
- 37 the range $<0:576]$ pels. In the case of interlaced pictures this constraint refers to the number of lines in
- 38 the frame; there will be half this number in each field.
- 39 The frame rate (defined by `frame_rate`) shall be in the range $<0:30]$ frames per second (and is
- 40 constrained to the discrete values that `frame_rate` can represent). The `frame_rate_extension` value shall

1 be zero. In addition the product of the horizontal size, the vertical size and the frame rate shall be
2 limited to lie in the range $<0:10\ 368\ 000]$ pels per second.

3 **8.2.3.2 Coded data rate and VBV buffer size**

4 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
5 R_{\max} for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the
6 range $<0:15\ 000\ 000]$ bits per second.

7 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range
8 $<0:626]$ kbits.

9 The VBV buffer size (`vbv_buffer_size`) shall be limited to the range $<0:1\ 835\ 008]$ bits.

10 **8.2.3.3 Vector range**

11 The reconstructed vertical motion vectors shall be limited to the range $[-128:+127.5]$ in frame pictures.
12 The reconstructed vertical motion vectors shall be limited to the range $[-64:+63.5]$ in field pictures.
13 Note that this rule applies to the final reconstructed motion vector after scaling (for dual-prime) has
14 been performed.

15 The values of `forward_horizontal_f_code` and `backward_horizontal_f_code` shall be limited to the
16 range $[1:8]$.

17 The values of `forward_vertical_f_code` and `backward_vertical_f_code` shall be limited to the range
18 $[1:5]$.

19 **8.2.3.4 intra_dc_precision**

20 The `intra_dc_precision` shall be 8 bit, 9 bit or 10 bit.

21 **8.2.4 High-1440 level**

22 **8.2.4.1 Frame dimensions**

23 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
24 be in the range $<0:1440]$ pels.

25 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
26 the range $<0:1152]$ pels. In the case of interlaced pictures this constraint refers to the number of lines
27 in the frame; there will be half this number in each field.

28 The frame rate (defined by `frame_rate`) shall be in the range $<0:60]$ frames per second (and is
29 constrained to the discrete values that `frame_rate` can represent). The `frame_rate_extension` value shall
30 be zero. In addition the product of the horizontal size, the vertical size and the frame rate shall be
31 limited to lie in the range $<0:47\ 001\ 600]$ pels per second.

32 **8.2.4.2 Coded data rate and VBV buffer size**

33 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
34 R_{\max} for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the
35 range $<0:60\ 000\ 000]$ bits per second.

36 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range $<0:2$
37 $503]$ kbits.

38 The VBV buffer size (`vbv_buffer_size`) shall be limited to the range $<0:7\ 340\ 032]$ bits.

39 **8.2.5 High level**

40 **8.2.5.1 Frame dimensions**

41 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
42 be in the range $<0:1920]$ pels.

43 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
44 the range $<0:1152]$ pels. In the case of interlaced pictures this constraint refers to the number of lines
45 in the frame; there will be half this number in each field.

46 The frame rate (defined by `frame_rate`) shall be in the range $<0:60]$ frames per second (and is
47 constrained to the discrete values that `frame_rate` can represent). The `frame_rate_extension` value shall

1 be zero. In addition the product of the horizontal size, the vertical size and the frame rate shall be
2 limited to lie in the range <0:62 668 800] pels per second.

3 **8.2.5.2 Coded data rate and VBV buffer size**

4 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
5 Rmax for variable bit rate operation, both of which are specified in bit_rate, shall be limited to the
6 range <0:80 000 000] bits per second.

7 The ratio of bit rate specified in bit_rate to frame rate specified in frame_rate shall lie in the range <0:3
8 337] kbits.

9 The VBV buffer size (vbv_buffer_size) shall be limited to the range <0:9 787 392] bits.

10 **8.3 SNR Scalable Profile**

11 Bitstreams that are compliant with the SNR Scalable profile shall meet the restrictions set out in clause
12 8.3.1.

13 In addition bitstreams that are compliant with the defined levels of the SNR Scalable profile shall meet
14 the restrictions set out in the corresponding clause.

Defined level	Clause specifying restrictions
Low	8.3.2
Main	8.3.3

15

16 **8.3.1 SNR Scalable profile syntax**

17 **8.3.1.1 Chroma sampling structure**

18 The chroma sampling structure (defined by chroma_format) shall be 4:2:0.

19 **8.3.1.2 Slice structure**

20 The “restricted slice structure” defined in clause 6.1.5.2 shall be used.

21 **8.3.2 Low level**

22 The Low Level of the SNR Scalable Profile may contain up to 2 layers of SNR scalability. We refer to
23 the upper SNR scalability layer as enhancement layer 1, and the lower SNR scalability layer as the
24 base layer.

25 **8.3.2.1 Frame dimensions**

26 The horizontal size of the frame (defined by horizontal_size_value and horizontal_size_extension) shall
27 be in the range <0:352] pels.

28 The vertical size of the frame (defined by vertical_size_value and vertical_size_extension) shall be in
29 the range <0:288] pels.

30 The frame rate (defined by frame_rate) shall be in the range <0:30] frames per second (and is
31 constrained to the discrete values that frame_rate can represent). The frame_rate_extension value shall
32 be zero. In addition, the product of the horizontal size, the vertical size and the frame rate shall be
33 limited to lie in the range <0:3 041 280] pels per second.

34 **8.3.2.2 Coded data rate and VBV buffer size**

35 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
36 Rmax for variable bit rate operation, both of which are specified in bit_rate, shall be limited to the
37 range <0:4 000 000] bits per second for all layers and <0:3 000 000] bits per second for the base layer.

38 The ratio of bit rate specified in bit_rate to frame rate specified in frame_rate shall lie in the range
39 <0:167] kbits for all layers and <0:126] kbits for the base layer.

40 The VBV buffer size shall be limited to the range <0:489 472] bits for all layers and <0:367 616] for
41 the base layer.

42 **8.3.2.3 Vector range**

43 See clause 8.2.3.3 “Vector range”

1 **8.3.2.4 intra_dc_precision**

2 See clause 8.2.3.4 “intra_dc_precision”

3 **8.3.3 Main level**

4 The Main Level of the SNR Scalable Profile may contain up to 2 layers of SNR scalability. We refer
5 to the upper SNR scalability layer as enhancement layer 1, and the lower SNR scalability layer as the
6 base layer.

7 **8.3.3.1 Frame dimensions**

8 The horizontal size of the frame (defined by horizontal_size_value and horizontal_size_extension) shall
9 be in the range <0:720] pels.

10 The vertical size of the frame (defined by vertical_size_value and vertical_size_extension) shall be in
11 the range <0:576] pels. In the case of interlaced pictures this constraint refers to the number of lines in
12 the frame; there will be half this number in each field.

13 The frame rate (defined by frame_rate) shall be in the range <0:30] frames per second (and is
14 constrained to the discrete values that frame_rate can represent). The frame_rate_extension value shall
15 be zero. In addition, the product of the horizontal size, the vertical size and the frame rate shall be
16 limited to lie in the range <0:10 368 000] pels per second.

17 **8.3.3.2 Coded data rate and VBV buffer size**

18 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
19 Rmax for variable bit rate operation, both of which are specified in bit_rate, shall be limited to the
20 range <0:15 000 000] bits per second for all layers and <0:10 000 000] bits per second for the base
21 layer.

22 The ratio of bit rate specified in bit_rate to frame rate specified in frame_rate shall lie in the range
23 <0:626] kbits for all layers and <0:418] kbits for the base layer.

24 The VBV buffer size shall be limited to the range <0:1 835 008] bits for all layers and <0:1 223 680]
25 for the base layer.

26 **8.3.3.3 Vector range**

27 See clause 8.2.3.3 “Vector range”

28 **8.3.3.4 intra_dc_precision**

29 See clause 8.2.3.4 “intra_dc_precision”

30 **8.4 Spatially Scalable Profile**

31 Bitstreams that are compliant with the Spatially Scalable profile shall meet the restrictions set out in
32 clause 8.4.1.

33 In addition bitstreams that are compliant with the defined level of the Spatially Scalable profile shall
34 meet the restrictions set out in the corresponding clause.

Defined level	Clause specifying restrictions
High-1440	8.4.2

35

36 **8.4.1 Spatially Scalable profile syntax**

37 **8.4.1.1 Chroma sampling structure**

38 The chroma sampling structure (defined by chroma_format) shall be 4:2:0.

39 **8.4.1.2 Slice structure**

40 The “restricted slice structure” defined in clause 6.1.5.2 shall be used.

41 **8.4.2 High-1440 level**

42 The High-1440 Level of the Spatially Scalable Profile may contain up to 3 layers of scalability. Of the
43 3 possible layers, the number of SNR scalable layers is limited to 2. Likewise, the number of spatially
44 scalable layers is limited to 2. We refer to the lower scalability layer as the base layer. The next layer

1 is referred to as enhancement layer 1. This is the top layer in a 2 layer bitstream. In case of 3 total
2 layers, we refer to the middle layer as enhancement layer 1 and the top layer as enhancement layer 2.

3 **8.4.2.1 Frame dimensions**

4 The horizontal size of the frame (defined by horizontal_size_value and horizontal_size_extension) shall
5 be in the range <0:1440] pels for the enhancement layers and in the range <0:720] for the base layer.

6 The vertical size of the frame (defined by vertical_size_value and vertical_size_extension) shall be in
7 the range <0:1152] pels for the enhancement layers and in the range <0:576] for the base layer. In the
8 case of interlaced pictures this constraint refers to the number of lines in the frame; there will be half
9 this number in each field.

10 The frame rate (defined by frame_rate) shall be in the range <0:60] frames per second for the
11 enhancement layers and <0:30] frames per second for the base layer (and is constrained to the discrete
12 values that frame_rate can represent). The frame_rate_extension value shall be zero. In addition, the
13 product of the horizontal size, the vertical size and the frame rate shall be limited to lie in the range
14 <0:47 001 600] pels per second for the enhancement layers and in the range <0:10 368 000] pels per
15 second for the base layer.

16 **8.4.2.2 Coded data rate and VBV buffer size**

17 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
18 Rmax for variable bit rate operation, both of which are specified in bit_rate, shall be limited to the
19 range <0:60 000 000] bits per second for all layers, <0:40 000 000] bits per second for the middle (i.e.
20 enhancement layer 1 given three layers) and base layers and <0:15 000 000] bits per second for the
21 base layer.

22 The ratio of bit rate specified in bit_rate to frame rate specified in frame_rate shall lie in the range
23 <0:2 503] kbits for all layers, <0:1 669] kbit/s for the middle and base layers and <0:626] kbits for the
24 base layer.

25 The VBV buffer size shall be limited to the range <0:7 340 032] bits for all layers, <0:4 893 696] for
26 the middle and base layers and <0:1 835 008] for the base layer.

27 **8.4.2.3 Vector range**

28 See clause 8.1.2.3 “Vector range”

29 **8.5 High profile**

30 Bitstreams that are compliant with the High profile shall meet the restrictions set out in clause 8.5.1.

31 In addition bitstreams that are compliant with the defined levels of the High profile shall meet the
32 restrictions set out in the corresponding clause.

Defined level	Clause specifying restrictions
Main	8.5.2
High-1440	8.5.3
High	8.5.4

33 **8.5.1 High profile syntax**

34 **8.5.1.1 Chroma sampling structure**

35 The chroma sampling structure (defined by chroma_format) shall be either 4:2:0 or 4:2:2.

36 **8.5.1.2 Slice structure**

37 The “restricted slice structure” defined in clause 6.1.5.2 shall be used.

38 **8.5.1.3 Scalability**

39 The High Profile may contain up to 3 layers of scalability. Of the 3 possible layers, the number of
40 SNR scalable layers is limited to 2. Likewise, the number of spatially scalable layers is limited to 2.
41 We refer to the lower scalability layer as the base layer. The next layer is referred to as enhancement
42 layer 1. This is the top layer in a 2 layer bitstream. In case of 3 total layers, we refer to the middle
43 layer as enhancement layer 1 and the top layer as enhancement layer 2.

1 **8.5.2 Main level**

2 **8.5.2.1 Frame dimensions**

3 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
4 be in the range $<0:720]$ pels for the enhancement layers and in the range $<0:352]$ for the base layer.

5 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
6 the range $<0:576]$ pels for the enhancement layers and in the range $<0:288]$ for the base layer. In the
7 case of interlaced pictures this constraint refers to the number of lines in the frame, there will be half
8 this number in each field.

9 The frame rate (defined by `frame_rate`) shall be in the range $<0:30]$ frames per second (and is
10 constrained to the discrete values that `frame_rate` can represent). The `frame_rate_extension` value shall
11 be zero. In addition, for 4:2:2 pictures the product of the horizontal size, the vertical size and the frame
12 rate shall be limited to lie in the range $<0:11\ 059\ 200]$ pels per second for the enhancement layers and
13 in the range $<0:3\ 041\ 280]$ pels per second for the base layer. For 4:2:0 pictures the product of the
14 horizontal size, the vertical size and the frame rate shall be limited to lie in the range $<0:14\ 745\ 600]$
15 pels per second for the enhancement layers and in the range $<0:3\ 041\ 280]$ pels per second for the base
16 layer..

17 **8.5.2.2 Coded data rate and VBV buffer size**

18 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
19 R_{max} for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the
20 range $<0:20\ 000\ 000]$ bits per second for all layers, $<0:15\ 000\ 000]$ bits per second for the middle (i.e.
21 enhancement layer 1 given three layers) and base layers and $<0:4\ 000\ 000]$ bits per second for the base
22 layer.

23 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range
24 $<0:835]$ kbits for all layers, $<0:626]$ kbit/s for the middle and base layers and $<0:167]$ kbits for the
25 base layer.

26 The VBV buffer size shall be limited to the range $<0:2\ 447\ 360]$ bits for all layers, $<0:1\ 835\ 008]$ for
27 the middle and base layers and $<0:489\ 472]$ for the base layer.

28 **8.5.2.3 Vector range**

29 See clause 8.2.3.3 "Vector range".

30 **8.5.2.4 intra_dc_precision**

31 The `intra_dc_precision` is not constrained and may take any of the values 8 bit, 9 bit, 10 bit or 11 bit.

32 **8.5.3 High-1440 level**

33 **8.5.3.1 Frame dimensions**

34 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
35 be in the range $<0:1440]$ pels for the enhancement layers and in the range $<0:720]$ for the base layer.

36 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
37 the range $<0:1152]$ pels for the enhancement layers and in the range $<0:576]$ for the base layer. In the
38 case of interlaced pictures this constraint refers to the number of lines in the frame, there will be half
39 this number in each field.

40 The frame rate (defined by `frame_rate`) shall be in the range $<0:60]$ frames per second for the
41 enhancement layers and in the range $<0:30]$ frames per second for the base layer (and is constrained to
42 the discrete values that `frame_rate` can represent). The `frame_rate_extension` value shall be zero. In
43 addition, for 4:2:2 pictures the product of the horizontal size, the vertical size and the frame rate shall
44 be limited to lie in the range $<0:47\ 001\ 600]$ pels per second for the enhancement layers and in the
45 range $<0:11\ 059\ 200]$ pels per second for the base layer. For 4:2:0 pictures the product of the
46 horizontal size, the vertical size and the frame rate shall be limited to lie in the range $<0:62\ 668\ 800]$
47 pels per second for the enhancement layers and in the range $<0:14\ 745\ 600]$ pels per second for the
48 base layer..

49 **8.5.3.2 Coded data rate and VBV buffer size**

50 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
51 R_{max} for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the

- 1 range <0:80 000 000] bits per second for all layers, <0:60 000 000] bits per second for the middle (i.e.
2 enhancement layer 1 given three layers) and base layers and <0:20 000 000] bits per second for the
3 base layer.
- 4 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range <0:3
5 337] kbits for all layers, <0:2 503] kbit/s for the middle and base layers and <0:835] kbits for the base
6 layer.
- 7 The VBV buffer size shall be limited to the range <0:9 786 392] bits for all layers, <0:7 340 032] for
8 the middle and base layers and <0:2 447 360] for the base layer.
- 9 **8.5.3.3 Vector range**
- 10 See clause 8.2.3.3 “Vector range”
- 11 **8.5.4 High level**
- 12 **8.5.4.1 Frame dimensions**
- 13 The horizontal size of the frame (defined by `horizontal_size_value` and `horizontal_size_extension`) shall
14 be in the range <0:1920] pels for the enhancement layers and in the range <0:960] for the base layer.
- 15 The vertical size of the frame (defined by `vertical_size_value` and `vertical_size_extension`) shall be in
16 the range <0:1152] pels for the enhancement layers and in the range <0:576] for the base layer. In the
17 case of interlaced pictures this constraint refers to the number of lines in the frame, there will be half
18 this number in each field.
- 19 The frame rate (defined by `frame_rate`) shall be in the range <0:60] frames per second for the
20 enhancement layers and <0:30] for the base layer (and is constrained to the discrete values that
21 `frame_rate` can represent). The `frame_rate_extension` value shall be zero. In addition, for 4:2:2
22 pictures the product of the horizontal size, the vertical size and the frame rate shall be limited to lie in
23 the range <0:62 668 800] pels per second for the enhancement layers and in the range <0:14 745 600]
24 pels per second for the base layer. For 4:2:0 pictures the product of the horizontal size, the vertical size
25 and the frame rate shall be limited to lie in the range <0:83 558 400] pels per second for the
26 enhancement layers and in the range <0:19 660 800] pels per second for the base layer.
- 27 **8.5.4.2 Coded data rate and VBV buffer size**
- 28 The coded data rate for fixed bit rate operation and the maximum average bitrate over a frame period
29 `Rmax` for variable bit rate operation, both of which are specified in `bit_rate`, shall be limited to the
30 range <0:100 000 000] bits per second for all layers, <0:80 000 000] bits per second for the middle
31 (i.e. enhancement layer 1 given three layers) and base layers and <0:25 000 000] bits per second for the
32 base layer.
- 33 The ratio of bit rate specified in `bit_rate` to frame rate specified in `frame_rate` shall lie in the range
34 <0:4 171] kbits for all layers, <0:3 337] kbit/s for the middle and base layers and <0:1 043] kbits for
35 the base layer.
- 36 The VBV buffer size shall be limited to the range <0:12 233 728] bits for all layers, <0:9 787 392] for
37 the middle and base layers and <0:3 036 160] for the base layer.
- 38 **8.5.4.3 Vector range**
- 39 See clause 8.2.3.3 “Vector range”.

Annex A

Discrete cosine transform

(This annex forms an integral part of this Recommendation | International Standard)

The NxN two dimensional DCT is defined as:

$$F(u, v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

with $u, v, x, y = 0, 1, 2, \dots, N-1$

where x, y are spatial coordinates in the pixel domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x, y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}$$

The input to the forward transform and output from the inverse transform is represented with 9 bits. The coefficients are represented in 12 bits. The dynamic range of the DCT coefficients is [-2048:+2047].

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990. Note that Clause 2.3 Std 1180-1990 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. The maximum refresh period requirement for this standard shall be 132 pictures, the same as indicated in 1180-1990 for visual telephony according to ITU-T Recommendation H.261.

(see Bibliography).

1

Annex B

2

Variable length code tables

3

(This annex forms an integral part of this Recommendation | International Standard)

4

B.1 Macroblock addressing

5

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

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

6

Note: The “macroblock stuffing” entry that was available in ISO/IEC11172-2 is not available in this specification.

7

1 **B.2 Macroblock type**

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

4 **Table B-2 — Variable length codes for macroblock_type in I-pictures**

macroblock_type VLC code								
		macroblock_quant						
							macroblock_motion_forward	
						macroblock_motion_backward		
					macroblock_pattern			
				macroblock_intra				
			spatial_temporal_weight_code_flag					
			permitted spatial_temporal_weight_classes					
			Description					
1	0	0	0	0	1	0	Intra	0
01	1	0	0	0	1	0	Intra, Quant	0

5

6 **Table B-3 — Variable length codes for macroblock_type in P-pictures**

macroblock_type VLC code								
		macroblock_quant						
							macroblock_motion_forward	
						macroblock_motion_backward		
					macroblock_pattern			
				macroblock_intra				
			spatial_temporal_weight_code_flag					
			permitted spatial_temporal_weight_classes					
			Description					
1	0	1	0	1	0	0	MC, Coded	0
01	0	0	0	1	0	0	No MC, Coded	0
001	0	1	0	0	0	0	MC, Not Coded	0
0001 1	0	0	0	0	1	0	Intra	0
0001 0	1	1	0	1	0	0	MC, Coded, Quant	0
0000 1	1	0	0	1	0	0	No MC, Coded, Quant	0
0000 01	1	0	0	0	1	0	Intra, Quant	0

7

1 **Table B-4 — Variable length codes for macroblock_type in B-pictures**

macroblock_type VLC code									
	macroblock_quant								
		macroblock_motion_forward							
			macroblock_motion_backward						
				macroblock_pattern					
					macroblock_intra				
						spatial_temporal_weight_code_flag			
							permitted spatial_temporal_weight_classes		
							Description		
10	0	1	1	0	0	0	Interp, Not Coded	0	
11	0	1	1	1	0	0	Interp, Coded	0	
010	0	0	1	0	0	0	Bwd, Not Coded	0	
011	0	0	1	1	0	0	Bwd, Coded	0	
0010	0	1	0	0	0	0	Fwd, Not Coded	0	
0011	0	1	0	1	0	0	Fwd, Coded	0	
0001 1	0	0	0	0	1	0	Intra	0	
0001 0	1	1	1	1	0	0	Interp, Coded, Quant	0	
0000 11	1	1	0	1	0	0	Fwd, Coded, Quant	0	
0000 10	1	0	1	1	0	0	Bwd, Coded, Quant	0	
0000 01	1	0	0	0	1	0	Intra, Quant	0	

2

3 **Table B-5 — Variable length codes for macroblock_type in I-pictures with spatial scalability.**

macroblock_type VLC code									
	macroblock_quant								
		macroblock_motion_forward							
			macroblock_motion_backward						
				macroblock_pattern					
					macroblock_intra				
						spatial_temporal_weight_code_flag			
							permitted spatial_temporal_weight_classes		
							Description		
1	0	0	0	1	0	0	Coded, Compatible	4	
01	1	0	0	1	0	0	Coded, Compatible, Quant	4	
0011	0	0	0	0	1	0	Intra	0	
0010	1	0	0	0	1	0	Intra, Quant	0	
0001	0	0	0	0	0	0	Not Coded, Compatible	4	

4

5 Note spatial_temporal_weight_code shall not present in the bitstream in intra pictures. Skipped
6 macroblocks are spatial only predicted.

1 **Table B-6 — Variable length codes for macroblock_type in P-pictures with spatial scalability.**

macroblock_type VLC code								
macroblock_quant								
macroblock_motion_forward								
macroblock_motion_backward								
macroblock_pattern								
macroblock_intra								
spatial_temporal_weight_code_flag								
permitted spatial_temporal_weight_classes								
Description								
10	0	1	0	1	0	0	MC, Coded	0
011	0	1	0	1	0	1	MC, Coded, Compatible	1,2,3
0000 100	0	0	0	1	0	0	No MC, Coded	0
0001 11	0	0	0	1	0	1	No MC, Coded, Compatible	1,2,3
0010	0	1	0	0	0	0	MC, Not Coded	0
0000 111	0	0	0	0	1	0	Intra	0
0011	0	1	0	0	0	1	MC, Not coded, Compatible	1,2,3
010	1	1	0	1	0	0	MC, Coded, Quant	0
0001 00	1	0	0	1	0	0	No MC, Coded, Quant	0
0000 110	1	0	0	0	1	0	Intra, Quant	0
11	1	1	0	1	0	1	MC, Coded, Compatible, Quant	1,2,3
0001 01	1	0	0	1	0	1	No MC, Coded, Compatible, Quant	1,2,3
0001 10	0	0	0	0	0	1	No MC, Not Coded, Compatible	1,2,3
0000 101	0	0	0	1	0	0	Coded, Compatible	4
0000 010	1	0	0	1	0	0	Coded, Compatible, Quant	4
0000 011	0	0	0	0	0	0	Not Coded, Compatible	4

2
 3 Note spatial_temporal_weight_code shall only be present in the bitstream when the
 4 spatial_temporal_weight_code is non zero. Skipped macroblocks are temporal only
 5 predicted.
 6

1 **Table B-7 — Variable length codes for macroblock_type in B-pictures with spatial scalability.**

macroblock_type VLC code								
macroblock_quant								
macroblock_motion_forward								
macroblock_motion_backward								
macroblock_pattern								
macroblock_intra								
spatial_temporal_weight_code_flag								
permitted spatial_temporal_weight_classes								
Description								
10	0	1	1	0	0	0	Interp, Not coded	0
11	0	1	1	1	0	0	Interp, Coded	0
010	0	0	1	0	0	0	Back, Not coded	0
011	0	0	1	1	0	0	Back, Coded	0
0010	0	1	0	0	0	0	For, Not coded	0
0011	0	1	0	1	0	0	For, Coded	0
0001 10	0	0	1	0	0	1	Back, Not Coded, Compatible	1,2,3
0001 11	0	0	1	1	0	1	Back, Coded, Compatible	1,2,3
0001 00	0	1	0	0	0	1	For, Not Coded, Compatible	1,2,3
0001 01	0	1	0	1	0	1	For, Coded, Compatible	1,2,3
0000 110	0	0	0	0	1	0	Intra	0
0000 111	1	1	1	1	0	0	Interp, Coded, Quant	0
0000 100	1	1	0	1	0	0	For, Coded, Quant	0
0000 101	1	0	1	1	0	0	Back, Coded, Quant	0
0000 0100	1	0	0	0	1	0	Intra, Quant	0
0000 0101	1	1	0	1	0	1	For, Coded, Compatible, Quant	1,2,3
0000 0110 0	1	0	1	1	0	1	Back, Coded, Compatible, Quant	1,2,3
0000 0111 0	0	0	0	0	0	1	Not Coded, Compatible	4
0000 0110 1	1	0	0	1	0	1	Coded, Compatible, Quant	4
0000 0111 1	0	0	0	1	0	1	Coded, Compatible	4

2

3 Note spatial_temporal_weight_code shall only be present in the bitstream when the
 4 spatial_temporal_weight_code is non zero. Skipped macroblocks are temporal only
 5 predicted.

6

1 **Table B-8 — Variable length codes for macroblock_type in I-pictures, P-pictures and B-pictures**
 2 **with SNR scalability.**

macroblock_type VLC code										
		macroblock_quant								
					macroblock_motion_forward					
							macroblock_motion_backward			
									macroblock_pattern	
									macroblock_intra	
									spatial_temporal_weight_code_flag	
									permitted spatial_temporal_weight_classes	
									Description	
1	0	0	0	1	0	0			Coded	0
01	1	0	0	1	0	0			Coded, Quant	0
001	0	0	0	0	0	0			Not Coded	0

3
 4 Note There is no differentiation between picture types, since macroblocks are processed
 5 identically in I, P and B-pictures. The "Not coded" type is needed, since skipped
 6 macroblocks are not allowed at beginning and end of a slice.

7

1 **B.3 Macroblock pattern**2 **Table B-9 --- Variable length codes for coded_block_pattern.**

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51
0110 1	52	0001 0001	23
0110 0	56	0001 0000	43
0101 1	1	0000 1111	25
0101 0	61	0000 1110	37
0100 1	2	0000 1101	26
0100 0	62	0000 1100	38
0011 11	24	0000 1011	29
0011 10	36	0000 1010	45
0011 01	3	0000 1001	53
0011 00	63	0000 1000	57
0010 111	5	0000 0111	30
0010 110	9	0000 0110	46
0010 101	17	0000 0101	54
0010 100	33	0000 0100	58
0010 011	6	0000 0011 1	31
0010 010	10	0000 0011 0	47
0010 001	18	0000 0010 1	55
0010 000	34	0000 0010 0	59
0001 1111	7	0000 0001 1	27
0001 1110	11	0000 0001 0	39
0001 1101	19	0000 0000 1	0 (NOTE)
NOTE — This entry shall not be used with 4:2:0 chrominance structure			

3

1 **B.4 Motion vectors**2 **Table B-10 --- Variable length codes for motion_code**

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

3

1

Table B-11 — Variable length codes for dmvector[0] and dmvector[1]

code	value
11	-1
0	0
10	1

2

3 **B.5 DCT coefficients**

4

Table B-12 --- Variable length codes for dct_dc_size_luminance

Variable length code	dct_dc_size_luminance
100	0
00	1
01	2
101	3
110	4
1110	5
1111 0	6
1111 10	7
1111 110	8
1111 1110	9
1111 1111 0	10
1111 1111 1	11

5

6

Table B-13 --- Variable length codes for dct_dc_size_chrominance

Variable length code	dct_dc_size_chrominance
00	0
01	1
10	2
110	3
1110	4
1111 0	5
1111 10	6
1111 110	7
1111 1110	8
1111 1111 0	9
1111 1111 10	10
1111 1111 11	11

7

1

Table B-14 --- DCT coefficients table zero

Variable length code (NOTE1)	run	level
10	End of Block	
1 s (NOTE2)	0	1
11 s (NOTE3)	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1 s	0	3
0011 1 s	3	1
0011 0 s	4	1
0001 10 s	1	2
0001 11 s	5	1
0001 01 s	6	1
0001 00 s	7	1
0000 110 s	0	4
0000 100 s	2	2
0000 111 s	8	1
0000 101 s	9	1
0000 01	Escape	
0010 0110 s	0	5
0010 0001 s	0	6
0010 0101 s	1	3
0010 0100 s	3	2
0010 0111 s	10	1
0010 0011 s	11	1
0010 0010 s	12	1
0010 0000 s	13	1
0000 0010 10 s	0	7
0000 0011 00 s	1	4
0000 0010 11 s	2	3
0000 0011 11 s	4	2
0000 0010 01 s	5	2
0000 0011 10 s	14	1
0000 0011 01 s	15	1
0000 0010 00 s	16	1
NOTE1 - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.		
NOTE2 - This code shall be used for the first (DC) coefficient in the block		
NOTE3 - This code shall be used for all other coefficients		

2

1

Table B-14 --- DCT coefficients table zero (continued)

Variable length code (NOTE)	run	level
0000 0001 1101 s	0	8
0000 0001 1000 s	0	9
0000 0001 0011 s	0	10
0000 0001 0000 s	0	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

2

Table B-14 --- DCT coefficients table zero (continued)

Variable length code (NOTE)	run	level
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

2

Table B-14 --- DCT coefficients table zero (concluded)

Variable length code (NOTE)	run	level
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

3

1

Table B-15 --- DCT coefficients table one

Variable length code (NOTE)	run	level
0110	End of Block	
10s	0	1
010 s	1	1
110 s	0	2
00101 s	2	1
0111 s	0	3
0011 1 s	3	1
0001 10 s	4	1
0011 0 s	1	2
0001 11 s	5	1
0000 110 s	6	1
0000 100 s	7	1
1110 0 s	0	4
0000 111 s	2	2
0000 101 s	8	1
1111 000 s	9	1
0000 01	Escape	
1110 1 s	0	5
0001 01 s	0	6
1111 001 s	1	3
0010 0110 s	3	2
1111 010 s	10	1
0010 0001 s	11	1
0010 0101 s	12	1
0010 0100 s	13	1
0001 00 s	0	7
0010 0111 s	1	4
1111 1100 s	2	3
1111 1101 s	4	2
0000 0010 0 s	5	2
0000 0010 1 s	14	1
0000 0011 1 s	15	1
0000 0011 01 s	16	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.		

2

1

Table B-15 --- DCT coefficients table one (continued)

Variable length code (NOTE)	run	level
1111 011 s	0	8
1111 100 s	0	9
0010 0011 s	0	10
0010 0010 s	0	11
0010 0000 s	1	5
0000 0011 00 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
1111 1010 s	0	12
1111 1011 s	0	13
1111 1110 s	0	14
1111 1111 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

2

Table B-15 --- DCT coefficients table one (continued)

Variable length code (NOTE)	run	level
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

2

Table B-15 --- DCT coefficients table one (concluded)

Variable length code (NOTE)	run	level
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

3

4

Table B-16 --- Encoding of run and level following an ESCAPE code

fixed length code	run	fixed length code	signed_level
0000 00	0	100000000001	-2047
0000 01	1	100000000010	-2046
0000 10	2
...	...	111111111111	-1
...	...	000000000000	forbidden
...	...	000000000001	+1
...
1111 11	63	011111111111	+2047

5

Annex C

Video buffering verifier

(This annex forms an integral part of this Recommendation | International Standard)

Constant rate coded video bitstreams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in Clause C.1. In variable bit-rate operation this annex is superseded by the STD models defined in ISO/IEC 13818-1. If the system coding layer defined is ISO/IEC 13818-1 is present, the STD model supercedes the VBV model including the definition of skipped pictures.

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

C.1 Video buffering verifier

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

C.1.2 The VBV has a receiving buffer of size B, where B is the **vbv_buffer_size** coded in the sequence header and sequence extension if any.

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

C.1.4 Starting at this time, the VBV buffer is examined at successive times defined in Clauses C.1.9 to C.1.11. Clauses C.1.5 to C.1.8 defines the actions to be taken at each time the VBV is examined.

C.1.5 This clause defines a requirement on all video bitstreams.

At the time the VBV buffer is examined *before* removing any picture data and immediately *after* this picture data is removed, when this applies (Cf. Clauses C.1.6 and C.1.7), the number of bits in the buffer shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by **vbv_buffer_size**.

For the purpose of this annex, picture data includes picture_data() as well as any header(s), user data and stuffing that immediately precedes it, and any trailing stuffing bits or bytes and user data immediately following picture_data() if they are present in the VBV buffer at the time of examination. For the first coded picture in the video sequence, any zero bit or byte stuffing immediately preceding the sequence_header() is also included in the picture data.

To meet this requirement the following inequality must be satisfied:

$$d_{n+1} > B_n + (t_{n+2} - t_n) * R - B$$

Real-valued arithmetic is used in this inequality.

where:

d_n is the picture data of n'th coded picture, measured in bits

B_n is the buffer occupancy (measured in bits) just *after* the n'th coded picture has been removed from the buffer

R = is the bit rate measured in bits/s. The full precision of the bitrate rather than the rounded value encoded by the **bit_rate** and **bit_rate_extension** fields shall be used by the encoder in the VBV model.

t_n is the time when the n'th coded picture is removed from the VBV buffer (measured in seconds, with full precision)

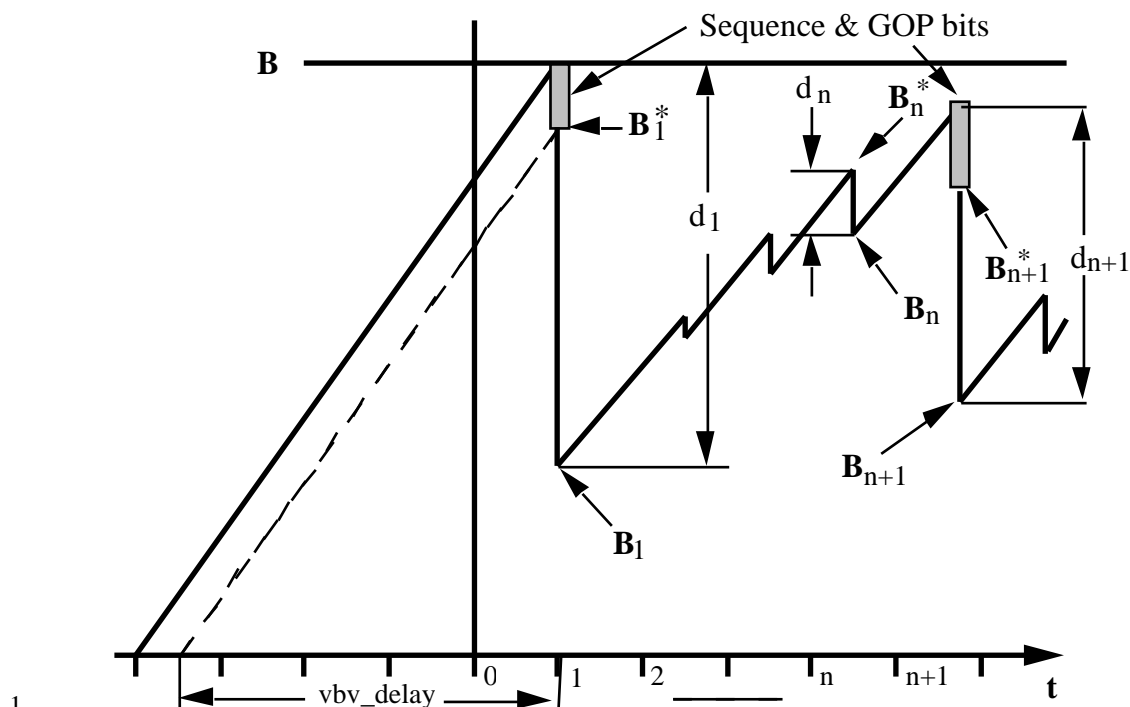


Figure C-1 VBV Buffer Occupancy

- 1
2
3 C.1.6 This clause defines a requirement on the video bitstreams when the **low_delay** flag is equal to
4 "0".
5
6 At each time the VBV buffer is examined and before any bits are removed, all of the data for
7 the picture which (at that time) has been in the buffer longest shall be present in the buffer.
8
9 Skipped pictures (that would happen if all the data of the picture were not present) shall not
10 occur when the **low_delay** flag is equal to "0".
11
12 C.1.7 This clause only applies when the **low_delay** flag is equal to "1".
13
14 This Clause describes the VBV action in case underflow would occur, i.e. when the VBV
15 buffer does not contain a complete picture at the time it is examined.
16
17 The following procedure applies in this case:
18
19 The buffer is re-examined at intervals of 2 field-periods, until the complete picture data is
20 present in the VBV input buffer. When this is the case, the number of bits in the buffer must
21 be less than B. At that point normal operation of the VBV resumes and clause C.1.5 applies.
22
23 The value of **temporal_reference** is not affected when this clause applies. For example,
24 when no B frame picture are present, **temporal_reference** is incremented by 1 for each frame
25 whether VBV buffer would underflow or not.
26
27 C.1.8 This clause is informative only.
28
29 The situation where VBV buffer would underflow (Clause C.1.7) can happen when low-delay
30 applications must transmit occasionally large pictures, for example in case of scene-cuts.
31
32 Decoding such bitstreams will cause the display process associated with a decoder to repeat a
33 previously decoded field or frame until normal operation of the VBV can resume. This
34 process is sometimes referred to as the occurrence of "skipped pictures". Note that his
35 situation should normally not occur except occasionally and only in low-delay bitstreams (i.e.,
36 when **low_delay** is equal to "1").
37
38 C.1.9 This clause defines the time intervals between successive examination of the VBV buffer
39 when **progressive_sequence** is equal to "1".

1 The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is
2 equal to the inverse of the frame rate.

3 C.1.10 This clause defines the time intervals between successive examination of the VBV buffer in
4 the case where **progressive_sequence** equals to "0" and **low_delay** equals to "0". In this
5 case, the frame reordering delay always exists and B frame pictures can occur.

6 The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a
7 multiple of T, where T is the inverse of two times the frame rate.

8 If the n'th picture is a *frame-structure* B frame picture with **repeat_first_field** equals to "0",
9 then $t_{n+1} - t_n$ is equal to $2 * T$.

10 If the n'th picture is a *frame-structure* B frame picture with **repeat_first_field** equals to "1",
11 then $t_{n+1} - t_n$ is equal to $3 * T$.

12 If the n'th picture is a *field-structure* B frame picture (B field picture), then $t_{n+1} - t_n$ is equal
13 to T.

14 If the n'th picture is a *frame-structure* P frame picture or I frame picture and if the previous P
15 frame picture or I frame picture has **repeat_first_field** equals to "0", then $t_{n+1} - t_n$ is equal to
16 $2 * T$.

17 If the n'th picture is a *frame-structure* P frame picture or I frame picture and if the previous P
18 frame picture or I frame picture has **repeat_first_field** equals to "1", then $t_{n+1} - t_n$ is equal to
19 $3 * T$.

20 If the n'th picture is the *first* field of a *field-structure* P frame picture or I frame picture, then
21 $t_{n+1} - t_n$ is equal to T.

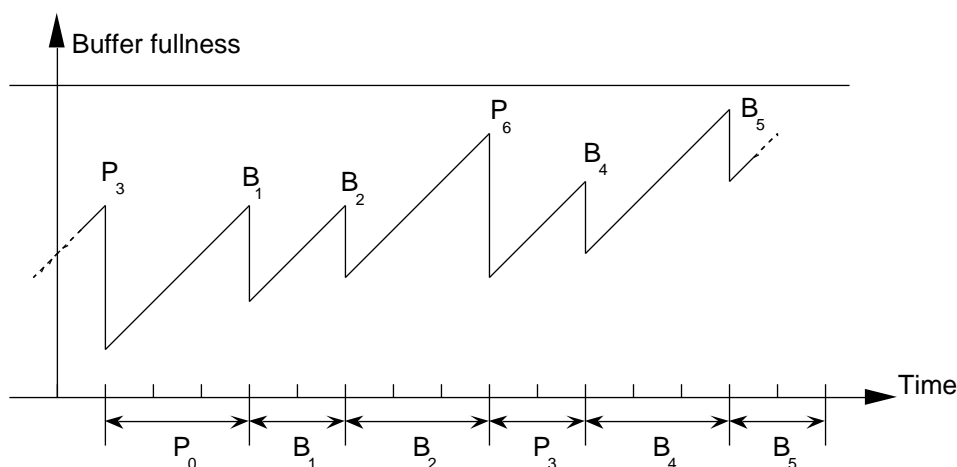
22 If the n'th picture is the *second* field of a *field-structure* P frame picture or I frame picture and
23 if the previous P frame picture or I frame picture is using field-structure or has
24 **repeat_first_field** equals to "0", then $t_{n+1} - t_n$ is equal to $(2 * T - T)$.

25 If the n'th picture is the *second* field of a *field-structure* P frame picture or I frame picture and
26 if the previous P frame picture or I frame picture is using frame-structure and has
27 **repeat_first_field** equals to "1", then $t_{n+1} - t_n$ is equal to $(3 * T - T)$.

28 If $t_{n+1} - t_n$ cannot be determined with any of the previous paragraphs because the previous P- or I frame
29 picture does not exist (which can occur at the beginning of a sequence), then the time interval is
30 arbitrary with the following restrictions:

31 The time interval between removing one frame (or the first field of a frame) and removing the next
32 frame(or the first field of a frame) can be arbitrarily defined equal to $2 * T$ or $3 * T$. The value of
33 **vbv_delay** can be used to determine which value was used.

34



35

36

Figure C-2

1 Figure C-2 shows the VBV in a simple case with only frame-pictures. Frames P₃, B₂ and B₄ have a
2 display duration of 3 fields.

3

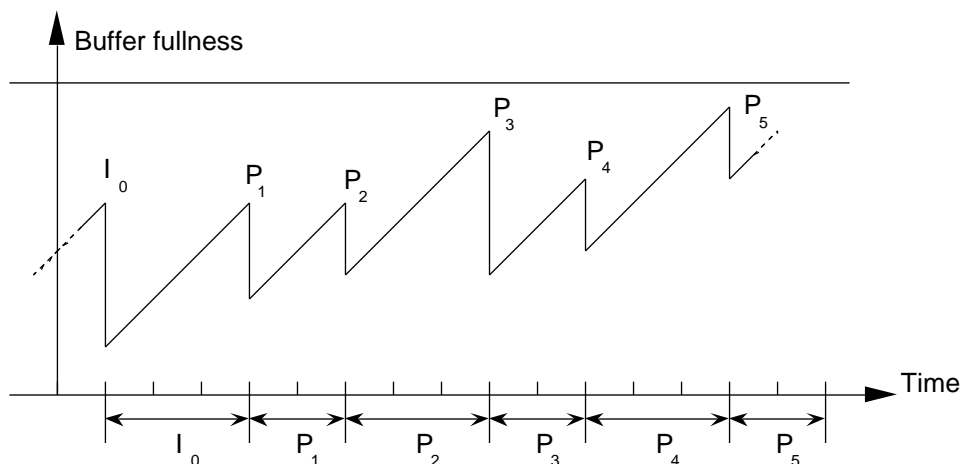
4 C.1.11 This clause defines the time intervals between successive examination of the VBV buffer in
5 the case where **progressive_sequence** equals to "0" and **low_delay** equals to "1". In this case
6 the sequence contains no B frame pictures and there is no frame reordering delay.

7 The time interval $t_{n+1} - t_n$ between two successive examinations of the VBV input buffer is a
8 multiple of T, where T is the inverse of two times the frame rate.

9 If the n'th picture is a *frame-structure* P frame picture or I frame picture with
10 **repeat_first_field** equals to "0", then $t_{n+1} - t_n$ is equal to $2 \cdot T$.

11 If the n'th picture is a *frame-structure* P frame picture or I frame picture with
12 **repeat_first_field** equals to "1", then $t_{n+1} - t_n$ is equal to $3 \cdot T$.

13 If the n'th picture is a *field-structure* P frame picture or I frame picture, then $t_{n+1} - t_n$ is equal
14 to T.



15

16

Figure C-3

17 Figure C-3 shows the VBV in a simple case with only frame-pictures. Frames I₀, P₂ and P₄ have
18 **repeat_first_field** equals to "1".

Annex D

Features supported by the algorithm

(This annex does not form an integral part of this Recommendation | International Standard)

D.1 Overview

The following non-exhaustive list of features is included in this specification:

- 1) Different chrominance sampling formats (i.e., 4:2:0, 4:2:2 and 4:4:4) can be represented.
- 2) Video in both the progressive and interlaced scan formats can be encoded.
- 3) The decoder can use 3:2 pulldown to represent a ~24 fps film as ~30 fps video.
- 4) The displayed video can be selected by a movable pan-scan window within a larger raster.
- 5) A wide range of picture qualities can be used.
- 6) Both constant and variable bit rate channels are supported.
- 7) A low delay mode for face-to-face applications is available.
- 8) Random access (for DSM, channel acquisition, and channel hopping) is available.
- 9) ISO/IEC 11172-2 bitstreams are decodable.
- 10) Bitstreams for high and low (hardware) complexity decoders can be generated.
- 11) Editing of encoded video is supported.
- 12) Fast-forward and fast-reverse playback recorded bitstreams can be implemented.
- 13) The encoded bitstream is resilient to errors.

D.2 Video Formats

D.2.1 Sampling Formats and Color

This specification video coding supports both interlaced and progressive video. The respective indication is provided with a *progressive_sequence* flag transmitted in the Sequence Extension code.

Allowed raster sizes are between 1 and $(2^{14} - 1)$ luminance pels each of the horizontal and vertical directions. The video is represented in a luminance/chrominance color space with selectable color primaries. The chrominance can be sampled in either the 4:2:0 (half as many samples in the horizontal and vertical directions), 4:2:2 (half as many samples in the horizontal direction only). Furthermore, application specific sample aspect ratios and image aspect ratios are flexibly supported. A *chroma_format* parameter is contained in the Sequence Extension code.

Sample aspect ratio information is provided by means of *aspect_ratio_information* and (optional) *display_horizontal_size* and *display_vertical_size* in the *sequence_display_extension()*. Examples of appropriate values for signals sampled in accordance with ITU-R Rec 601 are given in the following table:

Signal Format	display_horizontal_size	display_vertical_size
525-line	711	483
625-line	702	575

This specification implements tools to support 4:4:4 chrominance, for possible future use. However, this is currently not supported in any profile.

D.2.2 Movie Timing

A decoder can implement 3:2 pulldown when a sequence of progressive pictures is encoded. Each encoded movie picture can independently specify whether it is displayed for two or three video field periods, so "irregular" 3:2 pulldown source material can be transmitted as progressive video. Two flags,

1 *top_field_first* and *repeat_first_field*, are transmitted with the Picture Coding Extensions and
2 adequately describe the necessary display timing.

3 **D.2.3 Display Format Control**

4 The display process converts a sequence of digital frames (in the case of progressive video) or fields (in
5 the case of interlaced video) to output video. It is not a normative part of the this standard. The video
6 syntax of this specification does communicate certain display parameters for use in reconstructing the
7 video. Optional information (in the sequence display extension) specifies the chromaticities, the
8 display primaries, the opto-electronic transfer characteristics (e.g., the value of gamma) and the RGB-to-
9 luminance/chrominance conversion matrix.

10 Moreover, a display window within the encoded raster may be defined as, e.g., in the case of pan and
11 scan. Alternatively the encoded raster may be defined as a window on a large area display device. In
12 the case of pan-scan the position of the window representing the displayed region of a larger picture
13 can be specified on a field-by-field basis. It is specified in the Picture Display Extension described in
14 Clause 6.3.11. A typical use for the pan-scan window is to describe the "important" 4:3 aspect ratio
15 rectangle within a 16:9 video sequence. Similarly, in the case of small encoded pictures on a large
16 display the size of the display and the position of the window within that display may be specified.

17 **D.2.5 Transparent coding of composite video**

18 Decoding from PAL/NTSC before transmission and recoding to PAL/NTSC after transmission of
19 composite source signals in non low quality applications, such as contribution and distribution, requires
20 a precise reconstruction of the carrier amplitude and phase reference signal (and v-axis switch for
21 PAL).

22 The input format can be indicated in the sequence header using the *video_format* bits. Possible source
23 formats are: PAL, NTSC, SECAM and MAC. Reconstruction of the carrier signal is possible by using
24 the carrier parameters: *V_axis*, *field_sequence*, *sub_carrier*, *burst_amplitude* and *sub_carrier_phase*
25 that are enabled by setting the *composite_display_flag* in the picture header.

26 **D.3 Picture Quality**

27 High picture quality is provided according to the bit rate used. Provision for very high picture quality is
28 made by sufficiently high bit rate limits relating to a certain level in a particular profile. High chroma
29 band quality can be achieved by using 4:2:2 chroma

30 Quantiser matrices can be downloaded and used with a small a *quantiser_scale_code* to achieve near
31 lossless coding.

32 Moreover, scalable coding with flexible bit rate allows for service or quality hierarchy and graceful
33 degradation. E.g., decoding a sub-set of the bit stream carrying a lower resolution picture allows for
34 decoding this signal in a low-cost receiver with related quality; decoding the complete bit stream
35 allows to obtain the high overall quality.

36 Furthermore, operation at low bit rates can be accommodated by using low frame rates (by either pre-
37 processing before coding or frame skipping indicated by the *temporal_reference* in the picture header)
38 and low spatial resolution.

39 **D.4 Data Rate Control**

40 The number of transmitted bits per unit time, which is selectable in a wide range, may be controlled in
41 two ways, which are both supported by this specification. A *bit_rate* description is transmitted with the
42 Sequence Header Code.

43 For constant bit rate (CBR) coding, the number of transmitted bits per unit time is constant on the
44 channel. Since the encoder output rate generally varies depending on the picture content, it shall
45 regulate the rate constant by buffering etc. In CBR, picture quality may vary depending on its content.

46 The other mode is the variable bit rate (VBR) coding, in which case the number of transmitted bits per
47 unit time may vary on the channel under some constrictions. VBR is meant to provide constant quality
48 coding. A model for VBR application is near-constant-quality coding over B-ISDN channels subject to
49 Usage Parameter Control (UPC).

1 D.5 Low Delay Mode

2 A low encoding and decoding delay mode is accommodated for real-time video communications such
3 as visual telephony, video-conferencing, monitoring. Total encoding and decoding delay of less than
4 150 milliseconds can be achieved for low delay mode operation of THIS SPECIFICATION. Setting the
5 *low_delay_flag* in the Sequence Header code defines a low delay bit stream.

6 The total encoding and decoding delay can be kept low by generating a bit stream which does not
7 contain B-pictures. This prevents frame reordering delay. By using dual-prime prediction for P-frames
8 the picture quality can still be high.

9 A low buffer occupancy for both encoder and decoder is needed for low delay. Large coded pictures
10 should be avoided by the encoder. By using intra update on the basis of one or more slices per frame
11 (intra slices) instead of intra frames this can be accommodated.

12 In case of exceeding, for low delay operation, the desired number of bits per frame the encoder can skip
13 one or more frames. This action is indicated, to the decoder, by the state of the VBV buffer or the STD
14 buffer; i.e. underflow in the decoder buffer indicates that the encoder has skipped pictures.

15 D.6 Random Access/Channel Hopping

16 The syntax of this specification supports random access and channel hopping. Sufficient random
17 access/channel hopping functionality is possible by encoding suitable random access points into the bit
18 stream without significant loss of image quality.

19 Random access is an essential feature for video on a storage medium. It requires that any picture can be
20 accessed and decoded in a limited amount of time. It implies the existence of access points in the bit
21 stream -- that is segments of information that are identifiable and can be decoded without reference to
22 other segments of data. In this specification access points are provided by *sequence_header()* and this is
23 then followed by intra information (picture data that can be decoded without access to previously
24 decoded pictures). A spacing of two random access points per second can be achieved without
25 significant loss of picture quality.

26 Channel hopping is the similar situation in transmission applications such as broadcasting. As soon as a
27 new channel has been selected and the bit stream of the selected channel is available to the decoder, the
28 next data entry, i.e. random access point has to be found to start decoding the new program in the
29 manner outlined in the previous paragraph.

30 D.7 Scalability

31 The syntax of this specification supports bitstream scalability. To accommodate the diverse
32 functionality requirements of the applications envisaged by this specification a number of bit stream
33 scalability tools have been developed:

34 - **SNR scalability** mainly targets for applications which require graceful degradation.

35 - **Chroma Simulcast** targets at applications with high chroma quality requirements.

36 - **Data Partitioning** is primarily targeted for cell loss resilience in ATM networks.

37 - **Temporal Scalability** is a method suitable for interworking of services using high temporal
38 resolution progressive video formats. Also suitable for high quality graceful degradation in the presence
39 of channel errors.

40 - **Spatial Scalability** allows multiresolution coding technique suitable for video service interworking
41 applications. This tool can also provide coding modes to achieve compatibility with existing coding
42 standards, i.e. ISO/IEC 11172-2, at the lower layer.

43 D.7.1 Use of SNR scalability at a single spatial resolution

44 The aim of SNR scalability is primarily to provide a mechanism for transmission of a two-layer
45 service, these two layers providing the same picture resolution but different quality level. For example,
46 the transmission of service with two different quality levels is expected to become useful in the future
47 for some TV broadcast applications, especially when very good picture quality is needed for large size
48 display receivers. The sequence is encoded into two bitstreams called lower and enhancement layer
49 bitstreams. The lower layer bitstream can be decoded independently from the enhancement layer
50 bitstream. The lower layer, at 3 to 4 Mbit/s, would provide a picture quality equivalent to the current
51 NTSC/PAL/SECAM quality. Then, by using both the lower and the enhancement layer bitstreams, an

1 enhanced decoder can deliver a picture quality subjectively close to the studio quality, with a total bit
2 rate of 7 to 12 Mbit/s.

3 **D.7.1.1 Additional features**

4 **D.7.1.1.1 Error resilience**

5 As was noted in section D.3, the SNR scalable scheme can be used as a mechanism for error resilience.
6 If the two layer bitstreams are received with different error rate, the lower layer, better protected, stands
7 as a good substitute to fall back on, if the enhancement layer is damaged.

8 **D.7.1.1.2 Chroma simulcast**

9 The SNR scalable syntax can be used in a chroma simulcast system. The goal of such a scheme would
10 be to provide a mechanism for simultaneous distribution of services with the same luminance
11 resolution but different chrominance sampling format (e.g. 4:2:0 in the lower layer and 4:2:2, when
12 adding the enhancement layer and the simulcast chrominance components) for applications which
13 would require such a feature. The SNR scalable enhancement layer contains some luminance
14 refinement. The 4:2:2 chrominance is sent in simulcast. Only chrominance DC is predicted from the
15 lower layer. The combination of both layer luminance and of the 4:2:2 chrominance constitutes the
16 high quality level.

17 **D.7.1.2 SNR scalable encoding process**

18 **D.7.1.2.1 Description**

19 In the lower layer, the encoding is similar to the non scalable situation in terms of decisions, adaptive
20 quantisation, buffer regulation. The intra or error prediction macroblocks are DCT transformed. The
21 coefficients are then quantised using a first rather coarse quantiser. The quantised coefficients are then
22 VLC coded and sent together with the required side information (mb_type, vectors,
23 coded_block_pattern()).

24 In parallel, the quantised DCT coefficients coming from the lower layer, are dequantised. The residual
25 error between the coefficients and the dequantised coefficients is then re-quantised, using a second
26 finer quantiser. The resulting refinement coefficients are VLC coded and form the additional
27 enhancement layer, together with a marginal amount of side information (quantiser_scale_code,
28 coded_block_pattern()). The non-intra VLC table is used for all the coefficients in the enhancement
29 layer, since the transmitted signal is of differential nature.

30 **D.7.1.2.2 A few important remarks**

31 Since the prediction is the same for both layers, it is recommended to use the refined images in the
32 motion estimation loop (e.g. the images obtained by the conjunction of the lower and the enhancement
33 layer). Thus, there is a drift between the prediction signal used at the encoder side and what the low
34 level decoder can get as a prediction. This drift does accumulate from P-picture to P-picture and is
35 reset to zero at each I-Picture. However the drift has been found to have little visual effect intra pictures
36 every 15 pictures or so.

37 Since the enhancement layer only contains refinement coefficients, the needed overhead is quite
38 reduced: most of the information about the macroblocks (macroblock types, vectors...) are included in
39 the lower layer. Therefore the syntax of this stream is very much simplified:

40 - the macroblock type table only indicates if the quantiser_scale_code in the enhancement
41 layer has changed and if the macroblock is NOT-CODED (for first and last macroblock of the slices),
42 which amounts to three VLC words.

43 - quantiser_scale_code in the enhancement layer is sent if the value has changed.

44 - coded_block_pattern() is transmitted for all coded macroblocks.

45 All NOT-CODED macroblocks that are not at the beginning or end of a slice are skipped, since the
46 overhead information can be deduced from the lower layer.

47 It is recommended to use different weighting matrices for the lower and the enhancement layer. Some
48 better results are obtained when the first quantisation is steeper than the second one. However it is
49 recommended not to quantise too coarsely the DCT coefficient that corresponds to the interlace motion,
50 to avoid juddering effects.

1 **D.7.2 Multiple resolution scalability bitstreams using SNR scalability**

2 The aim of resolution scalability is to decode the base layer video suitable for display at reduced spatial
3 resolution. In addition it is desirable to implement a decoder with reduced complexity for this purpose.
4 This functionality is useful for applications where the receiver display is either not capable or willing to
5 display the full spatial resolution supported by both layers and for applications where software
6 decoding is targeted. The method described in this Section uses the SNR Scalability syntax outlined in
7 Clause 7 to transmit the video in two layers. Note that none of the options suggested in this Section
8 changes the structure of the highest resolution decoder, which remains identical to the one outlined in
9 Figure 7-14. The bitstream generated on both layers is compatible with the HIGH profile. However, the
10 base layer decoder could be implemented differently with reduced implementation complexity suitable
11 to software decoding.

12 **D.7.2.1 Decoder Implementation**

13 In decoding to a smaller spatial resolution, an inverse DCT of reduced size could be used when
14 decoding the base layer. The frame memory requirement in the decoder MC loop would also be
15 reduced accordingly.

16 If the bitstream of the two SNR Scalability layers was generated with only one MC loop at the encoder
17 the base video will be subject to drift. This drift may or may not be acceptable depending on the
18 application. Image quality will, to a large extent, depend on the sub-pel accuracy used for motion
19 compensation in the decoder. It is possible to use the full precision motion vector as transmitted in the
20 base layer for motion compensation with a sub-pel accuracy comparable to that of the higher layer.
21 Drift can be minimized by using advanced sub-pel interpolation filters (see "12", "13" and "16" in
22 Annex G).

23 **D.7.2.2 Encoder Implementation**

24 It is possible to tailor the base layer SNR Scalability bitstream to the particular requirements of the
25 resolution scaled decoder. A smaller DCT size can be more easily supported by only transmitting the
26 appropriate DCT-coefficients belonging to the appropriate sub-set in the base layer bitstream.

27 Finally it is possible to support a drift-free decoding at lower resolution scale by incorporating more
28 than one MC loop in the encoder scheme. An identical reconstruction process is used in the encoder
29 and decoder .

30 **D.7.3 Bitrate allocation in data partitioning**

31 Data partitioning allows splitting a bitstream for increased error resilience when two channels with
32 different error performance are available. It is often required to constrain the bitrate of each partition.
33 This can be achieved at the encoder by adaptively changing priority breakpoint at each slice.

34 The encoder can use two virtual buffers for the two bitstreams, and implement feedback rate control by
35 picking a priority breakpoint that approximately meets the target rate for each channel. Difference
36 between target and actual rates is used to revise the target for the next frame in a feedback loop.

37 It is desirable to vary the bitrate split from frame to frame for higher error resilience. Typically, I-
38 pictures benefit from having more of the data in partition 0 than the P-pictures while B-pictures could
39 be placed entirely in partition 1.

40 **D.7.4 Temporal scalability**

41 A two-layer temporally scalable coding structure consisting of a base and an enhancement layer is
42 shown in Figure 1. Consider video input at full temporal rate to temporal demultiplexer; in our example
43 it is temporally demultiplexed to form two video sequences, one input to the base layer encoder and the
44 other input to the enhancement layer encoder. The base-layer encoder is a non hierarchical encoder
45 operating at half temporal rate, the enhancement-layer encoder is like a MAIN profile encoder and also
46 operates at half temporal rate except that it uses base-layer decoded pictures for motion compensated
47 prediction. The encoded bitstreams of base and enhancement layers are multiplexed as a single stream
48 in the systems multiplexer. The systems demultiplexer extracts two bitstreams and inputs
49 corresponding bitstreams to base- and enhancement-layer decoders. The output of the base-layer
50 decoder can be shown standalone at half temporal rate or after multiplexing with enhancement-layer
51 decoded frames and shown at full temporal rate.

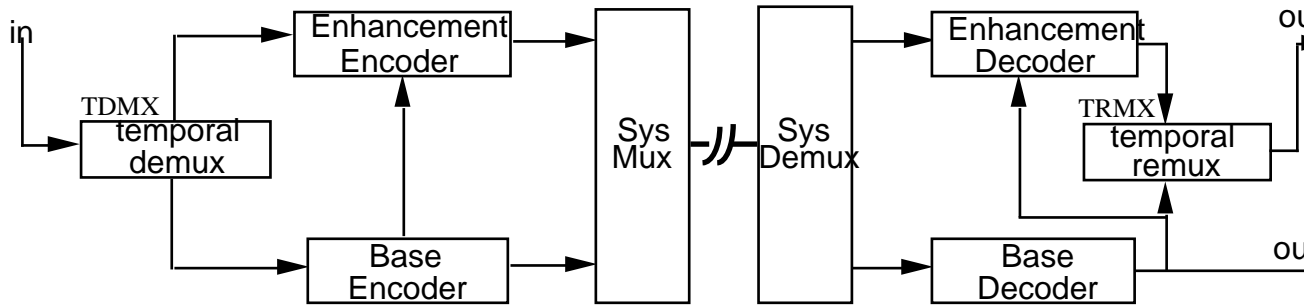


Fig 1 A two- layer codec structure for Temporal Scalability

1
2

3 The following forms of temporal scalability are supported and are expressed as higher layer: base
4 layer-to-enhancement layer picture formats.

- 5 1. Progressive:progressive-to-progressive Temporal Scalability
- 6 2. Progressive:interlace-to-interlace Temporal Scalability
- 7 3. Interlace:interlace-to-interlace Temporal Scalability

8 **D.7.4.1 Progressive:progressive-to-progressive Temporal Scalability**

9 Assuming progressive video input, if it is necessary to code progressive- format video in base and
10 enhancement layers, the operation of *temporal demux* may be relatively simple and involve temporal
11 demultiplexing of input frames into two progressive sequences; The operation of *temporal remux* is
12 inverse, i.e., it performs remultiplexing of two- progressive sequences to generate full temporal rate
13 progressive output.

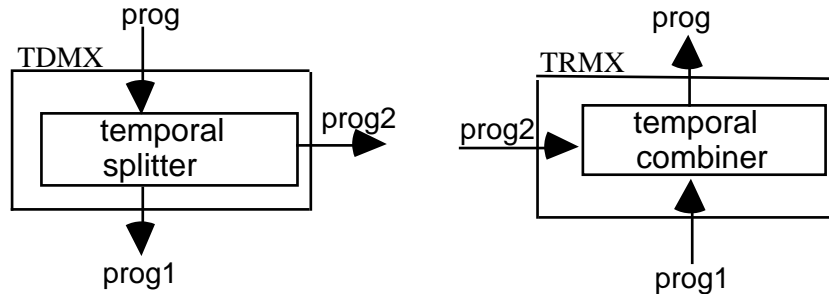


Fig 2 Temporal demultiplexer and remultiplexer for progressive:progressive-to-progressive temporal scalabilit

14

15 **D.7.4.2 Progressive:interlace-to-interlace temporal scalability**

16 Again, assuming full temporal rate progressive video input, if it is necessary to code interlaced-
17 format video in base layer, the operation of *temporal demux* may involve progressive to two-interlace
18 conversion; this process involves extraction of a normal interlaced- and a complementary interlaced-
19 sequence from progressive input video. The operation of *temporal remux* is inverse, i.e., it performs
20 two-interlace to progressive conversion to generate full temporal rate progressive output. Figure 3 and
21 Figure 4 show operations required in progressive to two-interlace and two interlace to progressive
22 conversion.

23

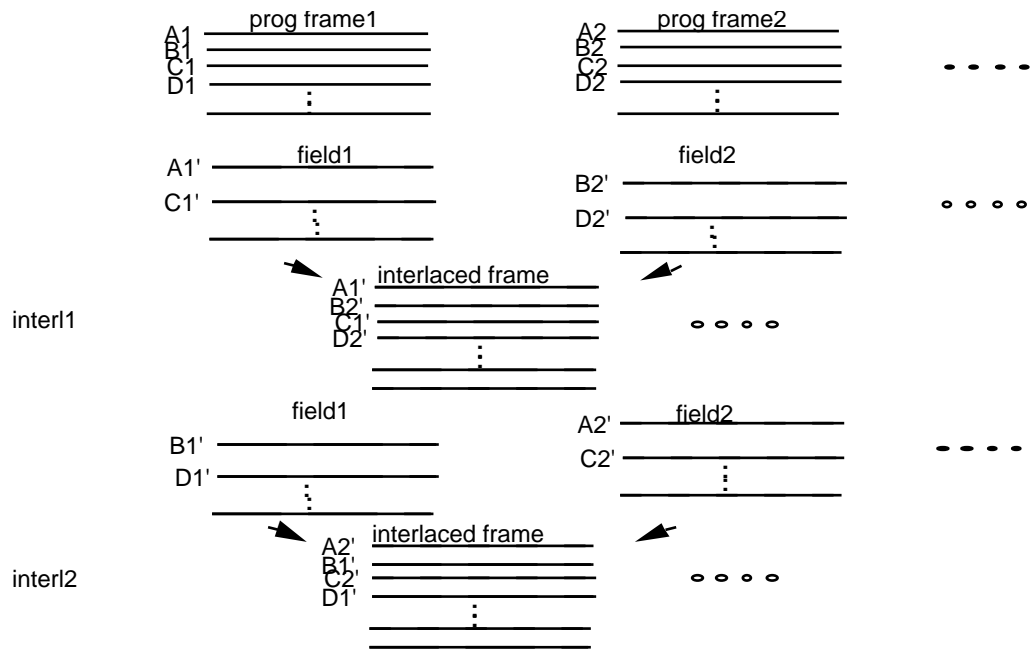


Fig 3 Progressive to two-Interlace conversion

1
2

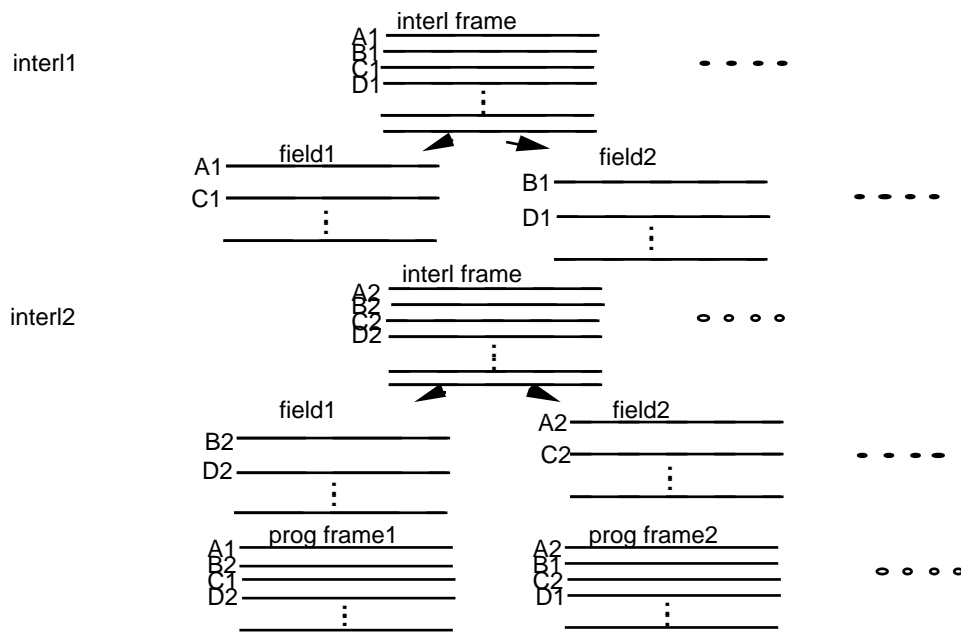


Fig 4 Two-Interlace to Progressive conversion

3
4

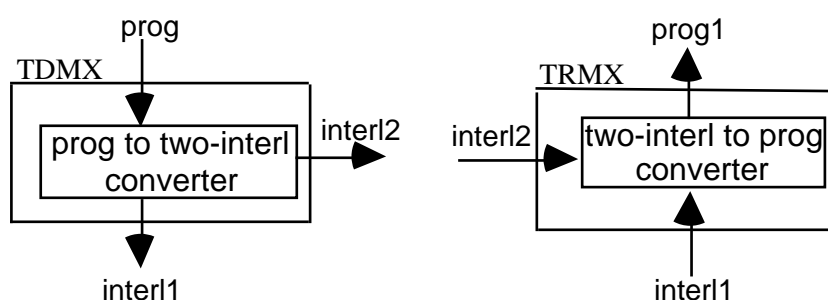


Fig 5 Temporal demultiplexer and remultiplexer for progressive:interlace-to-interlace temporal scalability

1

2

3 D.7.4.3 Interlace:interlace-to-interlace Temporal Scalability

4 Assuming interlaced video input, if it is necessary to code interlaced- format video in base- and
 5 enhancement- layers, the operation of *temporal demux* may be relatively simple and involve temporal
 6 demultiplexing of input frames into two interlaced sequences; The operation of *temporal remux* is
 7 inverse, i.e., it performs remultiplexing of two- interlaced sequences to generate full temporal rate
 8 interlaced output. The demultiplexing and remultiplexing is similar to that in Figure 2.

9 D.7.5 Hybrids of the spatial, the SNR and the temporal scalable extensions

10 This standard also allows combinations of scalability tools to produce more than 2 video layers as may
 11 be useful and practical to support more demanding applications. Taken two at a time, 3 explicit
 12 combinations result. Moreover, within each combination, the order in which each scalability is applied,
 13 when interchanged, results in distinct applications. In the hybrid scalabilities involving three layers,
 14 the layers are referred to as base layer, enhancement layer 1 and enhancement layer 2.

15 D.7.5.1 Spatial and SNR hybrid scalability applications

16 A) HDTV with standard TV at two qualities:

17 Base layer provides standard TV resolution at basic quality, enhancement layer 1 helps generate
 18 standard TV resolution but at higher quality by SNR scalability and the enhancement layer 2 employs
 19 HDTV resolution and format which is coded with spatial scalability with respect to high quality
 20 standard TV resolution generated by using enhancement layer 1.

21 B) Standard TV at two qualities and low definition TV/videophone:

22 Base layer provides videophone/low definition quality, using spatial scalability enhancement layer 1
 23 provides standard TV resolution at a basic quality and enhancement layer 2 uses SNR scalability to
 24 help generate high quality standard TV.

25 C) HDTV at two qualities and standard TV:

26 Base layer provides standard TV resolution. Using spatial spatial scalability enhancement layer1
 27 provides basic quality HDTV and enhancement layer 2 uses SNR scalability to help generate high
 28 quality HDTV.

29 D.7.5.2 Spatial and temporal hybrid scalability applications

30 A) High temporal resolution progressive HDTV with basic interlaced HDTV and standard TV:

31 Base layer provides standard TV resolution, using spatial scalability enhancement layer 1 provides
 32 basic HDTV of interlaced format and enhancement layer 2 uses temporal scalability to help generate
 33 full temporal resolution progressive HDTV.

34 B) High resolution progressive HDTV with enhanced progressive HDTV and basic progressive HDTV:

35 Base layer provides basic progressive HDTV format at temporal resolution, using temporal scalability
 36 enhancement layer 1 helps generate progressive HDTV at full temporal resolution and enhancement
 37 layer 2 uses spatial scalability to provide high spatial resolution progressive HDTV (at full temporal
 38 resolution).

1 C) High resolution progressive HDTV with enhanced progressive HDTV and basic interlaced HDTV:
2 Base layer provides basic interlaced HDTV format, using temporal scalability enhancement layer 1
3 helps generate progressive HDTV at full temporal resolution and enhancement layer 2 uses spatial
4 scalability to provide high spatial resolution progressive HDTV (at full temporal resolution).

5 **D.7.5.3 Temporal and SNR hybrid scalability applications**

6 A) Enhanced progressive HDTV with basic progressive HDTV at two qualities:

7 Base layer provides basic progressive HDTV at lower temporal rate, using temporal scalability
8 enhancement layer 1 helps generate progressive HDTV at full temporal rate but with basic quality and
9 enhancement layer 2 uses SNR scalability to help generate progressive HDTV with high quality (at full
10 temporal resolution).

11 B) Enhanced progressive HDTV with basic interlaced HDTV at two qualities:

12 Base layer provides provides interlaced HDTV of basic quality, using SNR scalability enhancement
13 layer 1 helps generate interlaced HDTV at high quality and enhancement layer 2 uses temporal
14 scalability to help generate progressive HDTV at full temporal resolution (at high quality).

15 **D.8 Compatibility**

16 The standard supports compatibility between different resolution formats as well as compatibility with
17 ISO/IEC 11172-2 (and ITU-T Rec. H.261).

18 **D.8.1 Compatibility with higher and lower resolution formats**

19 This specification supports compatibility between different resolution video formats. Compatibility is
20 provided for spatial and temporal resolutions with the Spatial Scalability and Temporal Scalability
21 tools. The video is encoded into two resolution layers. A decoder only capable or willing to display a
22 lower resolution video accepts and decodes the lower layer bitstream. The full resolution video can be
23 reconstructed by accepting and decoding both resolution layers provided.

24 **D.8.2 Compatibility with ISO/IEC IS 11172-2 (and ITU-T Rec. H.261)**

25 The syntax of this specification supports both backward and forward compatibility with
26 ISO/IEC 11172-2. Forward compatibility with ISO/IEC 11172-2 is provided since the syntax of this
27 specification is a superset of the ISO/IEC 11172-2 syntax. An MPEG-2 bit stream that does not contain
28 a *sequence_extension* is forward compatible. The Spatial Scalability tool provided by this specification
29 allows using ISO/IEC 11172-2 coding in the lower resolution, i.e. base layer, thus achieving backward
30 compatibility.

31 The video syntax contains tools that are needed to implement H.261 compatibility that may be needed
32 for possible future use, however, this is currently not supported by any profile.

33 Simulcast serves as a simple alternative method to provide backward compatibility with both H.261
34 and 11172-2.

35 **D.9 Complexity**

36 The MPEG-2 standard supports combinations of high performance/high complexity and low
37 performance/low complexity decoders. This is accommodated by MPEG-2 with the Profiles and Levels
38 definitions which introduce new sets of tool and functionality with every new profile. It is thus possible
39 to trade-off performance of the MPEG-2 coding schemes by decreasing implementation complexity.

40 Moreover, certain restrictions could allow reducing decoder implementation cost.

41 **D.9.1 Restrictions to reduce decoder implementation cost**

42 As the bitstream is currently specified, it is possible for a large number of consecutive macroblocks to
43 be coded such that each coefficient is represented by a 24 bit code. It is recognized that it is difficult
44 and unnecessarily expensive to implement a real-time device which is capable of decoding such a
45 bitstream allowed by this specification. Since such bitstreams are of no practical use, it would be
46 unreasonable that decoders should be required to decode them.

47 It appears that reasonable bitstreams will not represent any macroblock with data expansion - that is
48 they use fewer than $8 * 64 * B$ bits to represent a macroblock, where B is the number of blocks in the

1 source macroblock. A normative limit will be included in the Conformance Part (Part 4) of this
2 specification.

3 **D.10 Editing Encoded Bit Streams**

4 Many operations on the encoded bitstream are supported to avoid the expense and quality costs of re-
5 coding. Editing, and concatenation of encoded bitstreams with no recoding and no disruption of the
6 decoded image sequence is possible.

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

10 Editing of encoded MPEG-2 bitstreams is supported due to the syntactic hierarchy of the encoded
11 video bitstream. Unique start codes are encoded with different level in the hierarchy (i.e. video
12 sequence, group of pictures etc.). Video can be encoded with Intra-picture/intra-slices access points in
13 the bitstream. This enables the identification, access and editing of parts of the bitstream without the
14 necessity to decode the entire video.

15 **D.11 Trick modes**

16 Certain DSM (Digital Storage Media) provide the capability of trick modes, such as FF/FR (Fast
17 Forward/Fast Reverse). The MPEG-2 syntax supports all special access, search and scan modes of
18 11172-2. This functionality is supported with the syntactic hierarchy of the video bitstream which
19 enables the identification of relevant parts within a video sequence. It can be assisted by MPEG-2 tools
20 which provide bit stream scalability to limit the access bit rate (i.e. Data Partitioning and the general
21 slice structure). This section provides some guideline for decoding a bitstream provided by a DSM.

22 The decoder is informed by means of a 1bit flag (`DSM_trick_mode_flag`) in the PES packet header.
23 This flag indicates that the bitstream is reconstructed by DSM in trick mode, and the bitstream is valid
24 from syntax point of view, but invalid from semantics point of view. When this bit is set, an 8bit field
25 (`DSM_trick_modes`) follows. The semantics of `DSM_trick_modes` are in the ISO/IEC 13818-1.

26

27 **[Decoder]**

28 While the decoder is decoding PES Packet whose `DSM_trick_mode_flag` is set to 1, the decoder is
29 recommended to:

30 Decode bitstream and display according to `DSM_trick_modes`

31

32 **[Pre_processing]**

33 When the decoder encounters PES Packet whose `DSM_trick_mode_flag` is set to 1, the decoder is
34 recommended to:

35 Clear non `trick_mode` bitstream in buffer

36 **[Post_processing]**

37 When the decoder encounters PES Packet whose `DSM_trick_mode_flag` is set to 0, the decoder is
38 recommended to:

39 Clear `trick_mode` bitstream in buffer

40

41 **[Video Part]**

42 While the decoder is decoding PES Packet whose `DSM_trick_mode_flag` is set to 1, the decoder is
43 recommended to:

44 Neglect VBV and Temporal Reference value

45 Decode one picture and display it until next picture is decoded.

46 The bitstream in trick mode may have a gap between slices. When the decoder encounters a gap
47 between slices, the decoder is recommended to:

1 Decode the slice and display it according to VP(vertical position) in slice header

2 Fill up the gap with co-sited part of the last displayed picture

3 **[Encoder]**

4 The encoder is recommended to:

5 Encode with short size of slice with Intra_macroblocks.

6 Encode with short periodic refreshment by Intra_picture or Intra_slice.

7 **[DSM]**

8 DSM is recommended to provide the bitstream in trick mode with perfect syntax.

9 [Preprocessing]

10 DSM is recommended to:

11 Complete "normal" bitstream at Picture_layer or higher layer.

12 [System Part]

13 DSM is recommended to:

14 Set DSM_trick_mode_flag to 1 in a PES Packet header.

15 Set DSM_trick_modes(8bit) according to the trick mode.

16 [Video Part]

17 DSM is recommended to:

18 Insert Sequence header with the same parameter as normal bitstream except bitrate. Set bitrate
19 = 3FFFFFFF, this is VBR(Variable Bit)

20 Insert Sequence extension header with the same parameter as normal bitstream

21 Insert Picture header with the same parameter as normal bitstream

22 Remark: Temporal reference and VBV are neglected in the decoder, therefore DSM needs not to set
23 temporal reference and VBV in correct value.

24 Concatenate Slices which consists of Intra_coded_Macroblocks. The concatenated slices
25 should have VP's in increasing order.

26 **D.12 Error Resilience**

27 Most digital storage media and communication channels are not error-free. Appropriate channel coding
28 schemes should be used and are beyond the scope of this specification. Nevertheless the MPEG-2
29 syntax supports error resilient modes relevant to cell loss in ATM networks and bit errors (isolated and
30 in bursts) in transmissions. The slice structure of the compression scheme defined in this specification
31 allows a decoder to recover after a residual data error and to resynchronise its decoding. Therefore, bit
32 errors in the compressed data will cause errors in the decoded pictures to be limited in area. Decoders
33 may be able to use concealment strategies to disguise these errors. Error resilience includes graceful
34 degradation in proportion to bit error rate (BER) and graceful recovery in the face of missing video bits
35 or data packets. It has to be noted that all items may require additional support at the system level.

36 Being an example of a packet-based system, B-ISDN with its Asynchronous Transfer Mode (ATM) is
37 addressed in some detail in the following. Similar statements can be made for other systems where
38 certain packets of data are protected individually by means of forward error-correcting coding.

39 ATM uses short, fixed length packets, called cells, consisting of a 5 byte header containing routing
40 information, and a user payload of 48 bytes. The nature of errors on ATM is such that some cells may
41 be lost, and the user payload of some cells may contain bit errors. Depending on AAL (ATM
42 Adaptation layer) functionality, indications of lost cells and cells containing bit errors may be available.

43 As an indication of the impact of cell loss in an ATM environment Table D-1 summarises the average
44 interval between cell losses for a range of CLR and service bit rates based on simple statistical
45 modelling. (A cell payload must be assumed for this. Allowing 1 byte/cell for AAL functions leaves

1 376 bits = 47 bytes). Note, however, that this summary ignores cell loss bursts and other shorter term
 2 temporal statistics.

3 **Table D-1. Average interval between cell losses for a range of CLR and service bit rates.**

	Average interval time of error			
	5 Mb/s	10 Mb/s	50 Mb/s	100 Mb/s
10^{-2}	7.52 ms	3.76 ms	0.752 ms	0.376 ms
10^{-3}	75.2 ms	37.6 ms	7.52 ms	3.76 ms
10^{-4}	752 ms	376 ms	75.2 ms	37.6 ms
10^{-5}	7.52 s	3.76 s	752 ms	376 ms
10^{-6}	1.25 m	37.6 s	7.52 s	3.76 s
10^{-7}	12.5 m	6.27 m	1.25 m	37.6 s
10^{-8}	2.09 h	1.04 h	12.5 m	6.27 m

4

5 Bit Error Ratios (BERs) corresponding to the above mean times between errors can be calculated easily
 6 for the case of isolated bit errors. The BER that would cause the same incidence rate of errors is found
 7 by dividing by the cell payload size. i.e. $BER = CLR/376$.

8 The following techniques of minimising the impact of lost cells and other error/loss effects are
 9 provided for reference, and indicate example methods of using the various tools available in this
 10 specification to provide good performance in the presence of those errors. Note that the techniques
 11 described may be applicable in the cases of packets of other sizes (e.g. LANs or certain storage media)
 12 or video data with uncorrected errors of different characteristics, in addition to cell loss. It may be
 13 appropriate to treat a known erasure (uncorrected bit error(s) known to exist somewhere in a data
 14 block) as a lost data block, since the impact of bit errors cannot be predicted. However, this should be a
 15 decoder option. The discussion that follows refers generally to "transport packets" where appropriate,
 16 to emphasise the applicability to a variety of transport and storage systems. However, specific
 17 examples will refer to Cell Loss Ratios (CLRs) because cell transport is the most completely defined at
 18 the time of preparing this specification.

19 The error resilience techniques are summarised in three categories, covering methods of concealing the
 20 error once it has occurred, and the restriction of the influence of a loss or error in both space (within a
 21 picture) and time (from picture to picture).

22 **D.12.1. Concealment possibilities**

23 Concealment techniques hide the effect of losses/errors once they have occurred. Some concealment
 24 methods can be implemented using any encoded bitstream, while others are reliant on the encoder to
 25 structure the data or provide additional information to enable enhanced performance.

26 **D.12.1.1 Temporal predictive concealment**

27 A decoder can provide concealment of the errors by estimating the lost data from spatio-temporally
 28 adjacent data. The decoder uses information which has been successfully received to make an informed
 29 estimate of what should be displayed in place of the lost/errored data, under the assumption that the
 30 picture characteristics are fairly similar across adjacent blocks (in both the spatial and temporal
 31 dimensions). In the temporal case, this means estimation of errored or lost data from nearby fields or
 32 frames.

33 **D.12.1.1.1 Substitution from previous frame**

34 The simplest possible approach is to replace a lost macroblock with the macroblock in the same
 35 location in the previous picture. This approach is suitable for relatively static picture areas but block
 36 displacement is noticeable for moving areas.

37 Note that "previous picture" must be interpreted with care due to the use of bi-directional prediction
 38 and a difference between picture decoding order and picture display order. When a macroblock is lost
 39 in a P- or I-picture, it can be concealed by copying the data corresponding to the same macroblock in
 40 the previous P-picture or I-picture. This ensures that the picture is complete before it is used for further

1 prediction. Lost macroblocks in B-pictures can be substituted from the last displayed picture, of any
2 type, or from a future I- or P-picture held in memory but not yet displayed.

3 **D.12.1.1.2 Motion compensated concealment**

4 The concealment from neighbouring pictures can be improved by estimating the motion vectors for the
5 lost macroblock, based on the motion vectors of neighbouring macroblocks in the affected picture
6 (provided these are not also lost). This improves the concealment in moving picture areas, but there is
7 an obvious problem with errors in macroblocks whose neighbouring macroblocks are coded intra,
8 because there are ordinarily no motion vectors. Encoder assistance to get around this problem is
9 discussed in clause D.3.1.1.3.

10 Sophisticated motion vector estimation might involve storage of adjacent macroblock motion vectors
11 from above and below the lost macroblock, for predictions both forward and backward (for B-pictures)
12 in time. The motion vectors from above and below (if available) could then be averaged.

13 Less complex decoders could use, for example, only forward prediction and/or only the motion vector
14 from the macroblock above the lost macroblock. This would save on storage and interpolation.

15 **D.12.1.1.3 Use of Intra MVs**

16 The motion compensated concealment technique outlined in D.3.1.1.2 could not ordinarily be applied
17 when the macroblocks above and below the lost/errored macroblock are Intra-coded, since there is no
18 motion vector associated with Intra-coded macroblocks. In particular, in I-pictures, this type of
19 concealment would not be possible with the normal calculation and use of motion vectors.

20 The encoding process can be extended to include motion vectors for intra macroblocks. Of course, the
21 motion vector and coded information for a particular macroblock must be transmitted separately (e.g. in
22 different packets) so that the motion vector is still available in the event that the image data is lost.

23 When "concealment_motion_vectors" = 1, motion vectors are transmitted with Intra macroblocks,
24 allowing improved concealment performance of the decoders. The concealment motion vector
25 associated with an Intra-coded macroblock is intended to be used only for concealment (if necessary)
26 of the macroblock located immediately below the Intra-coded macroblock.

27 For simplicity, concealment motion vectors associated with Intra-coded macroblocks are always
28 forward, and are considered as frame motion-vectors in Frame-pictures and field motion-vectors in
29 field-pictures.

30 Therefore, encoders that choose to generate concealment motion vectors should transmit, for a given
31 Intra-coded macroblock, the frame- or field-motion vector that should be used to conceal (i.e. to
32 predict, with forward frame- or field-based prediction respectively) the macroblock located
33 immediately below the Intra-coded macroblock.

34 Concealment motion vectors are intended primarily for I- and P-pictures, but the syntax allows their
35 use in B-pictures. Concealment in B-pictures is not critical, since B-pictures are not used as predictors
36 and so errors do not propagate to other pictures. Therefore, it may be wasteful to transmit concealment
37 motion vectors in B-pictures.

38 Concealment motion vectors transmitted with Intra macroblocks located in the bottom row of a picture
39 cannot be used for concealment. However, if "concealment_motion_vectors" = 1, those concealment
40 motion vectors must be transmitted. Encoders can use the (0, 0) motion vector to minimise the coding
41 overhead.

42 When concealment motion vectors are used, it is a good idea to have one slice contain one row of
43 macroblocks (or smaller), so that concealment can be limited to less than one row of macroblocks when
44 a slice, or part of a slice, is lost. This means that the loss of macroblocks in two successive rows is
45 much less likely, and therefore the chances of achieving effective concealment using concealment
46 motion vectors is improved.

47 Note that, when "concealment_motion_vectors" = 1, Prediction Motion Vectors (PMVs) are NOT reset
48 when an Intra macroblock is transmitted. Ordinarily, an Intra macroblock would reset the PMVs.

49 **D.12.1.2 Spatial predictive concealment**

50 The generation of predicted, concealment macroblocks is also possible by interpolation from
51 neighbouring macroblocks within the one picture [Annex E-17]. This is best suited to areas of high
52 motion, where temporal prediction is not successful, or as an alternative means of concealment for Intra

1 macroblocks when concealment motion vectors (clause D.3.1.1.3) are not available. It also could be
2 particularly useful for cell loss after scene changes.

3 There are several possible approaches to spatial interpolation, and it could be carried out in the spatial
4 or DCT domain, but normally it is only feasible and useful to predict the broad features of a lost
5 macroblock, such as the DC coefficient and perhaps the lowest AC coefficients. Spatial prediction of
6 fine detail (high frequencies) is likely to be unsuccessful and is of little value in fast-moving pictures
7 anyway.

8 Spatial predictive macroblock concealment may also be useful in combination with layered coding
9 methods (i.e. Data Partitioning or SNR scalable pyramid, see D.3.1.3). If in the event of cell loss some
10 DCT coefficients in a macroblock are recovered from the lower layer, it is possible to use all
11 information available (DCT coefficients recovered in the same macroblock from the lower layer and all
12 DCT coefficients received in the adjacent macroblocks) for error concealment. This is especially useful
13 if the lower layer only contains DC coefficients due to bandwidth constraints.

14 **D.12.1.3 Layered coding to facilitate concealment**

15 It is possible to assist the concealment process further by arranging the coded video information such
16 that the most important information is most likely to be received. The loss of the less important
17 information can then be more effectively concealed. This approach can gain from use of a transmission
18 medium or storage device with different priority levels (such as priority-controlled cell-based
19 transmission in the B-ISDN, or where different error protection or correction is provided on different
20 channels). The components produced by the coding process can be placed in a hierarchy of importance
21 according to the effect of loss on the reconstructed image. By indicating the priority of bitstream
22 components and treating the individual components with due importance, superior error concealment
23 performance may be possible.

24 Strategies available for producing hierarchically ordered bitstreams, or layers, include

25 **data partitioning** - the coded macroblock data is partitioned into multiple layers such that partition
26 zero contains address and control information and lower order DCT coefficients, while partition one
27 contains high frequency DCT coefficients.

28 **SNR scalable pyramid** - two sets of coefficients are dequantised and then added together at the
29 receiver before decoding. One set of coefficients could be a refinement of the quantisation error of the
30 other, but other combinations (including an emulation of data partitioning) are possible.

31 **spatial scalable pyramid** - the lower layer may be coded without regard for the enhancement layer,
32 and could use other standard coding methods (ISO/IEC 11172-2 etc.). The enhancement layer contains
33 the coded difference signal from a prediction based on the lower layer.

34 **temporal scalable pyramid** - the enhancement layer defines additional pictures which, when
35 remultiplexed with the base layer, provides a combined picture sequence of greater picture rate.

36 These strategies produce layers which, when added progressively, produce increasing quality of the
37 reconstructed signal. While some of these source coding techniques may result in a bit rate increase
38 compared to the system without layering, the performance of the layered systems, when subjected to
39 channel errors, may be greater.

40 Considering error resilience alone, the hierarchically ordered layers should be handled with due quality,
41 such that some function (such as picture quality for a given total bit rate) is optimised. The bitstream
42 components may be treated differently at one or more of the following locations:

- 43 • encoder - different channel coding might be used
- 44 • channel - the channel may be able to provide different cell/packet loss probabilities or error
45 characteristics to the different bitstream components.
- 46 • decoder - error concealment could be performed differently within each bitstream

47 **D.12.1.3.1 Use of data partitioning**

48 Data partitioning allows a straightforward division of macroblock data into two layers. The PBP
49 pointer determines the contents of each layer. Ordinarily, data partition 0 contains the address and
50 control information and the low frequency DCT coefficients, while data partition 1 contains the high
51 frequency DCT coefficients.

1 At the encoder the value of the PBP pointer may be different for each slice such that the distribution of
2 bits between the two layers may be controlled (e.g. maintained constant). The distribution may be
3 different for I, P, and B frames. The management of rate between the layers could mean that, for some
4 macroblocks, data partition 0 contains no DCT coefficients or motion vectors.

5 Good tolerance to errors can be achieved if channel errors are distributed so that data partition 1
6 receives most errors.

7 It is assumed that errors can be detected at the decoder, so that actions can be taken to prevent errored
8 data from being displayed. For data partition 1, errored data is simply not displayed (i.e. only data
9 partition 0 is used). Losses or errors in data partition 0 should be minimised through use of high
10 reliability transport. Decoder concealment actions may also be necessary.

11 **D.12.1.3.2 Use of SNR scalable coding**

12 SNR scalable coding provides two layers with the same spatial resolution but different image quality,
13 depending on whether one or both layers are decoded. This technique is mainly intended to provide a
14 lower-quality layer that is usable even when the enhancement layer is absent. However, it also provides
15 good error resilience if the errors can be mainly confined to the enhancement layer.

16 In case of errors in the enhancement layer the lower layer signal can be used alone for the affected
17 image area. Especially in the case of frequent errors, temporary loss or permanent unavailability of the
18 enhancement layer this concealment is very effective, since the displayed signal can be made relatively
19 free of non-linear distortions like blocking or motion jerkiness.

20 If the enhancement layer is permanently unavailable and so only the lower layer is decoded, a small
21 drift may occur in the case where only one MC prediction loop is implemented in the encoder.
22 However, this drift is likely to be invisible in most configurations (e.g. $M=3$, $N=12$ would normally
23 provide correction often enough).

24 The lower-layer signal of an SNR Scalable system is well suited to concealment in the case of a very
25 high error rate, temporary or permanent loss of the upper-layer signal. However, the upper-layer quality
26 in the error-free case does not achieve that of a sub-band like layered scheme (e.g. data partitioning).

27 **D.12.1.3.3 Use of spatial scalable coding**

28 Spatial scalable coding allows the lower layer to be coded without regard for the enhancement layer,
29 and other standard coding methods (ISO/IEC 11172-2 etc.) could be used. The enhancement layer
30 contains the coded difference signal from a prediction based on the lower layer. In case of errors in the
31 enhancement layer the upconverted lower layer signal can be used directly as concealment information
32 for the affected image area. Especially in case of frequent errors or temporary loss of the enhancement
33 layer signal this concealment signal is relatively free of non-linear distortions like blocking (which
34 could arise if high frequency DCT coefficients are completely absent from the lower layer) or motion
35 jerkiness (if the motion information is omitted from the high priority layer).

36 In the error-free case the upconverted lower layer signal is used as an additional prediction signal in a
37 macroblock-adaptive way to improve the upper-layer coding performance. The enhancement layer bit
38 stream therefore consists of the quantised residual temporal or lower layer prediction errors.

39 Spatial scalable coding provides a lower layer signal that is very suitable for concealment in case of a
40 high error rate or temporary loss of the enhancement layer signal. However, the quality of the enhanced
41 picture when both layers are available will not, in general, be as good as other layered coding
42 approaches.

43 **D.12.1.3.4 Use of temporal scalable coding**

44 Temporal scalability is a coding technique that allows layering of video frames. The spatial resolution
45 of frames in each layer is the same but the temporal rates of each layer are lower than that of the
46 source; however the combined temporal rate of the two layers results in full temporal rate of the source.
47 In case of errors in the enhancement layer, the base layer of full spatial resolution can be easily used for
48 concealment. Especially in case of frequent errors or temporary loss of the enhancement layer signal,
49 the base layer signal offers good concealment properties.

50 In telecommunications applications such as those employing the SCIF format high degree of error
51 resilience can be achieved with temporal scalability by encoding the base- layer using the SCIF spatial
52 resolution but only half the temporal resolution; the remaining frames corresponding to the other half
53 of the temporal resolution are coded in the enhancement- layer. Typically, the enhancement- layer data
54 may be assigned lower priority and when lost, the base- layer decoded frames can be used for

1 concealment by frame repetition. This type of concealment leads to only a temporary loss of full
2 temporal resolution while maintaining full spatial quality and full spatial resolution.

3 In HDTV applications such as those using high temporal resolution progressive video format as source,
4 high degree of error resilience can be achieved with temporal scalability. Such an application is
5 envisaged to require 2 layers, a base- layer and an enhancement- layer, each of which process same
6 picture formats (either both progressive or both interlaced) but at half the temporal rates. Temporal
7 remultiplexing of the base- and enhancement- layer signals irrespective of their chosen formats always
8 results in full progressive temporal resolution of the source. In HDTV transmission, if the lower
9 priority enhancement layer signal is corrupted, the base- layer signal can be used for concealment,
10 either directly, as in case of progressive format base- layer or after reversal of parity of fields for
11 interlaced format base- layer.

12 Typically, the enhancement layer data may be assigned lower priority and when lost, the base layer
13 decoded frames can be used for concealment by either frame repetition or frame averaging. This type
14 of concealment leads to only a temporary indistinguishable loss in temporal resolution while
15 maintaining full spatial quality and full spatial resolution.

16 **D.12.2 Spatial localisation**

17 Spatial localisation encompasses those methods aimed at minimising the extent to which errors
18 propagate within a picture, by providing early resynchronisation of the elements in the bitstream that
19 are coded differentially between macroblocks.

20 Isolated bit errors may be detected through invalid codewords and so a decoder designer may choose to
21 allow an errored sequence to be decoded. However, the effect on the picture is difficult to predict
22 (legal, but incorrect, codewords could be generated) and it may be preferable to control the error
23 through concealment of the entire affected slice(s) even when only one bit is known to be in error
24 somewhere in a block of data.

25 When long consecutive errors occur (e.g. packet or cell loss), virtually the only option is to discard data
26 until the next resynchronisation point is located (a start code at the next slice or picture header). By
27 providing more resynchronisation points, the area of the screen affected by a loss or error can be
28 reduced, in turn reducing the demands on the concealment techniques and making the errors less visible
29 at the expense of coding efficiency. Spatial localisation of errors is therefore dependent on controlling
30 the slice size since this is the smallest coded unit with resynchronisation points (start codes).

31 **D.12.2.1 Small slices**

32 The most basic method for achieving spatial localisation of errors is to reduce the (fixed) number of
33 macroblocks in a slice. The increased frequency of resynchronisation points will reduce the affected
34 picture area in the event of a loss. It is effective in any transport or storage media, and in any profile
35 since the slice structure is always present in MPEG coded video.

36 The method results in a small loss of coding efficiency due to the increase of overhead information.
37 The loss is about 3% for 11 Macroblocks per slice and 12% for 4 Macroblocks per slice based on Rec.
38 601 picture format at 4 Mb/s, (percentages calculated relative to a system using 44 Macroblocks, or one
39 picture width, per slice). The efficiency loss results in degradation of picture quality up to about 1 dB
40 with 4 Macroblocks per slice and 0.2 dB with 11 Macroblocks per slice without errors at 4 Mb/s.
41 However, the method performs approximately 1 to 5 dB better at $CLR = 10^{-2}$, depending on the
42 concealment method used (simple macroblock replacement or motion compensated concealment).

43 From the view point of perceived picture quality, the performance of this method is generally
44 dependent on the relative size of slice size and picture. Therefore, the slice size should be decided by
45 considering the picture size (in macroblocks) and the trade-off between coding efficiency and visual
46 degradation due to errors.

47 **D.12.2.2 Adaptive slice size**

48 There is a significant variation in the number of bits required to code a picture slice, depending on the
49 coding mode, picture activity, etc. If slices contain only a few macroblocks, it will be possible that one
50 transport packet, even a short packet or cell, could contain several slices. Offering multiple
51 resynchronisation points in the same transport packet serves no purpose. Another problem with the
52 simplistic short slice approach is that, because no account is taken of the transport packet structure, the
53 first valid transport packet after a loss could contain most of the information for a slice, but it is
54 unusable because the start code was lost.

1 An improvement over the small slice method may be to use adaptive slice sizes. As the encoder is
2 producing the bitstream, it keeps track of the data contents within transport packets. The start of a slice
3 is placed at the first opportunity in every transport packet (or in every second, third, ...). This approach
4 can achieve about the same spatial localisation of errors as small, fixed size slices, but with a greater
5 efficiency.

6 Note, however, that this method ONLY gives an advantage for cell or packet based transmission, or
7 where error detection occurs over a large block of data. The frequent resynchronisation points of small
8 slice localisation are only wasteful if more than one is lost in the event of an error. If isolated bit errors
9 affect just one slice anyway, then there is no advantage in adapting the slice size.

10 Furthermore, the adaptive slice size technique requires an intimate connection between encoder and
11 packetiser, to allow a new slice for a new packet or cell. As such, it may not be appropriate for some
12 applications (e.g. stored video intended to be distributed by multiple means) because only one transport
13 packet structure would be assumed during encoding.

14 **D.12.3 Temporal localisation**

15 Temporal localisation encompasses those methods aimed at minimising the extent to which errors
16 propagate from picture to picture in the temporal sequence, by providing early resynchronisation of
17 pictures that are coded differentially. An obvious way to do this is to make use of intra mode coding.

18 **D.12.3.1 Intra pictures**

19 By use of intra pictures a single error will not stay in the decoded picture longer than $(N + M - 1)$
20 pictures if every N th picture is coded intra and $(M-1)$ B pictures are displayed before each I picture.

21 While the intra pictures, normally used as "anchors" for synchronising the video decoding part way
22 through a sequence, are useful for temporal localisation, care should be taken in adding extra intra
23 pictures (i.e. reducing N) for error resilience. Intra pictures require a large number of bits to code, take
24 up a relatively large proportion of the encoded bitstream and, as a result, are more likely to be affected
25 by losses or errors themselves.

26 **D.12.3.2 Intra slices**

27 To avoid the additional delay caused by intra pictures, some applications requiring low delay may want
28 to update the picture by coding only parts of the picture intra. This may provide the same kind of error
29 resilience as intra pictures. As an example assume that a constant number of slices per picture from top
30 to bottom are intra coded so that the whole picture is updated every P pictures. Three aspects of this
31 kind of updating should be kept in mind:

32 • While an errored portion of the scene will ordinarily be erased within P pictures (with an
33 average duration of about $P/2$), it is possible that motion compensation will allow the disturbance to
34 bypass the intra refresh and it may persist as long as $2P$ pictures.

35 • To ensure that errors are not propagating into the updated region of the picture, restrictions
36 could be put on motion vectors, limiting the vertical vector components to ensure that predictions are
37 not made from the "oldest" parts of the picture.

38 • The visual effect of clearing errors can be similar to a windscreen wiper clearing water. This
39 *windscreen wiper* effect can become noticeable in some cases in the error free sequence, unless the rate
40 control mechanism ensures that the quality of the intra slice is close to that of the surrounding non-intra
41 macroblocks.

42 **D.12.4 Summary**

43 Table D-2 summarises the above error resilience techniques, with a guide to their applicability.

1

Table D-2. Summary of error concealment techniques.

Category	Technique	Profile/Applicability
Concealment	Temporal predictive - substitution from previous picture	Any profile. Most suited to static pictures.
	Temporal predictive - Motion compensated	Any profile. Choice of sophistication in motion vector estimation.
	Temporal predictive - using concealment MVs	Any profile, but calculation of Intra MVs is an encoder option.
	Spatial predictive	Any profile. Not suitable for static, complex pictures.
	Data Partitioning	Not currently used in a profile, but may be added as post/pre-processing. Minimal overhead and complexity. Depending on bit-rate allocation, lower layer may not provide usable pictures by itself.
	SNR Scalability	SNR SCALABLE, SPATIALLY SCALABLE, HIGH profiles. Suitable for very high error rates or temporary unavailability of the enhancement layer. Relatively simple to implement.
	Spatial Scalability	SPATIALLY SCALABLE and HIGH profiles. Suitable for very high error rates or temporary unavailability of the enhancement layer.
	Temporal Scalability	Not currently used in a profile. Suitable for very high error rates or temporary unavailability of the enhancement layer.
Spatial Localisation	Small Slices	Any profile
	Adaptive slice sizes	Any profile, but requires knowledge of transmission characteristics when packet size is decided.
Temporal Localisation	Intra pictures	Any profile, but has delay implications.
	Intra slices	Any profile, but errors may persist longer than for Intra picture method.

2

3 It is not possible to provide a concise indication of error resilience performance, because assessments
4 must necessarily be subjective and application dependent, and so should be taken as nothing more than
5 a guide. It is also true that several different approaches to error resilience are likely to be used in
6 combination. However, the following descriptions are provided as some guidance to performance.
7 They are the results of cell loss experiments, looking only at cell-based transmission of video
8 information.

9 A simple macroblock substitution from a previous frame combined with the small-slice method (4
10 macroblocks per slice) will provide adequate picture quality for most sequences in the presence of
11 rather low error rates of around $CLR = 10^{-5}$ (in a reference 4 Mbit/s, Main Profile, Main Level
12 system).

13 Including sophisticated motion compensated concealment (with full spatial and temporal interpolation
14 of motion vectors for lost macroblocks, and concealing losses in P pictures that use intra slice updating,
15 i.e. $N = \text{infinity}$, $M = 1$) provides adequate picture quality at $CLR = 10^{-3}$ (again, in a reference 4 Mbit/s,
16 Main Profile, Main Level system).

17 Operation in environments with greater loss may require use of one of the layered coding methods.
18 With adequate protection of the high priority information, these schemes can provide adequate
19 performance in the face of CLR's as high as 10^{-2} or even 10^{-1} . Data partitioning, implemented as a
20 post-processing function to a 4 Mbit/s Main Profile, Main Level system, with 50% of the rate allocated
21 to each partition and no loss in the base layer, has been shown in one example to give approximately
22 0.5 dB loss in SNR at a CLR of 10^{-3} , about 1.5 dB loss at 10^{-2} , and with almost no visible degradation
23 in either case.

1 Given the range of different layered coding approaches that are possible, some general comments may
2 be useful. In general, it is not expected that inclusion of the most complex layered coding methods
3 could be justified purely on the basis of error resilience. Instead, they could be utilised for error
4 resilience if they were required to satisfy other system requirements. Data partitioning is very simple to
5 implement and is likely to provide error resilience very nearly the same as any of the other methods
6 except in the case of extremely high error rates (>10% loss) or where the enhancement signal could be
7 lost completely. SNR scalability is slightly more complex, and has slightly lower efficiency than data
8 partitioning, but it is easier to produce lower layer signals of a usable quality when the enhancement
9 layer is absent. Spatial scalability is more complex again, but provides a good lower layer picture
10 quality at the expense of overall (two layer) efficiency.
11

1

Annex E

2

Profile and level restrictions

3

(This annex does not form an integral part of this Recommendation | International Standard)

4

This clause tabulates all of the syntactic elements defined in this specification. Each is classified to indicate whether it is restricted by either the profile or level. Where restrictions do apply these restrictions are documented in tables.

5

6

7

Note. This clause is informative and is simply intended as a summary of the normative restrictions set out in clause 8. If, because of an error in the preparation of this text, a discrepancy exists between clause 8 and clause E the normative text in clause 8 shall always take precedence.

8

9

10

In the tables that follow a number of abbreviations are used as shown in table E-1.

11

Table E-1. Abbreviations used in the tables of clause E

Abbreviation	Used in	Meaning
x	Status	included in the Profile
o	Status	not included in the Profile
(i)	Status	implicit
(c)	Status	conditional syntax element
(o)	Status	optional syntax element
D	Type	item with Level-dependent parameters
I	Type	item independent of the Level in the Profile
P	Type	item for post-processing after decoding

12

1

Table E-2. Sequence headers.

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL			TEMPORAL			
		MAIN		SNR				
		SIMPLE		SNR				
01	horizontal_size_value	x	x	x	x	x	D	See table E-3
02	vertical_size_value	x	x	x	x	x	D	See table E-3
03	aspect_ratio_information	x	x	x	x	x	P	
04	frame_rate_code	x	x	x	x	x	D	See tables E-3 and E-4
05	(pel rate) Note: this is not a syntactic element	(i)	(i)	(i)	(i)	(i)	D	See table E-5
06	bit_rate_value	x	x	x	x	x	D	See table E-6
07	vbv_buffer_size_value	x	x	x	x	x	D	See table E-7
08	constrained_parameters_flag	x	x	x	x	x	I	set to '1' if MPEG-1 constrained set to '0' if MPEG-2
09	load_intra_quantiser_matrix	x	x	x	x	x	I	
10	intra_quantiser_matrix[64]	x (c)	x (c)	x (c)	x (c)	x (c)	I	
11	load_non_intra_quantiser_matrix	x	x	x	x	x	I	
12	non_intra_quantiser_matrix[64]	x (c)	x (c)	x (c)	x (c)	x (c)	I	
13	Sequence Extension	x	x	x	x	x	I	(always present if MPEG-2)
14	Sequence Display Extension	(o)	(o)	(o)	(o)	(o)	I	
15	Sequence Scalable Extension	o	o	x	x	x	I	See table E-8 for maximum number of scalable layers.
16	User data	(o)	(o)	(o)	(o)	(o)	I	

2

1

Table E-3. Upper bounds for sampling density

Level	Spatial resolution layer		Profile				
			Simple	Main	SNR	Spatial	High
High	Enhancement	pels/line lines/frame frames/sec		-			1920 1152 60
	Base	pels/line lines/frame frames/sec		1920 1152 60			960 576 30
High-1440	Enhancement	pels/line lines/frame frames/sec		-		1440 1152 60	1440 1152 60
	Base	pels/line lines/frame frames/sec		1440 1152 60		720 576 30	720 576 30
Main	Enhancement	pels/line lines/frame frames/sec	-	-	-		720 576 30
	Base	pels/line lines/frame frames/sec	720 576 30	720 576 30	720 576 30		352 288 30
Low	Enhancement	pels/line lines/frame frames/sec		-	-		
	Base	pels/line lines/frame frames/sec		352 288 30	352 288 30		

2

3

Table E-4. Frame rates allowed (independent of profile)

Level	Frame rate (Hz)
High	23.976/24/25/29.97/30/50/59.94/60
High-1440	23.976/24/25/29.97/30/50/59.94/60
Main	23.976/24/25/29.97/30
Low	23.976/24/25/29.97/30

4

1

Table E-5. Upper bounds for luminance pel rate (Msamples/sec)

Level	Spatial resolution layer	Profile				
		Simple	Main	SNR	Spatial	High
High	Enhancement		-			62.6688 (4:2:2) 83.5584 (4:2:0)
	Base		62.6688			14.7456 (4:2:2) 19.6608 (4:2:0)
High-1440	Enhancement		-		47.0016	47.0016 (4:2:2) 62.6688 (4:2:0)
	Base		47.0016		10.3680	11.0592 (4:2:2) 14.7456 (4:2:0)
Main	Enhancement	-	-	-		11.0592 (4:2:2) 14.7456 (4:2:0)
	Base	10.3680	10.3680	10.3680		- 3.04128 (4:2:0)
Low	Enhancement		-	-		
	Base		3.04128	3.04128		

2

3

Table E-6. Upper bounds for bit rates (Mbit/s)

Level	Profile				
	Simple	Main	SNR	Spatial	High
High		80			100 all layers 80 middle + base layer 25 base layer
High-1440		60		60 all layers 40 middle + base layers 15 base layer	80 all layers 60 middle + base layers 20 base layer
Main	15	15	- 15 both layers 10 base layer		20 all layers 15 middle + base layer 4 base layer
Low		4	- 4 both layers 3 base layer		

4

1

Table E-7. Buffer size requirements (bits)

Level	Profile				
	Simple	Main	SNR	Spatial	High
High		9,787,392			12,233,728 9,787,392 3,036,160
High-1440		7,340,032		7,340,032 4,893,696 1,835,008	9,786,392 7,340,032 2,447,360
Main	1,835,008	1,835,008	- 1,835,008 1,223,680		2,447,360 1,835,008 489,472
Low		489,472	- 489,472 367,616		

2

3

Table E-8. Upper bounds for scalable layers in SNR, Spatial and High profiles

Level	Maximum Number of	Profile		
		SNR	Spatial	High
High	All layers (base + enh.)			3
	Spatial enhancement layers			1
	SNR enhancement layers			1
High-1440	All layers (base + enh.)		3	3
	Spatial enhancement layers		1	1
	SNR enhancement layers		1	1
Main	All layers (base + enh.)	2		3
	Spatial enhancement layers	0		1
	SNR enhancement layers	1		1
Low	All layers (base + enh.)	2		
	Spatial enhancement layers	0		
	SNR enhancement layers	1		

4

1

Table E-9. Sequence extension

#	Syntactic elements	Status					Type	Parameter range in Level
		SIMPLE	MAIN	SNR	SPATIAL	HIGH		
01	profile_and_level_indication	x	x	x	x	x	D	Profile: one of 8 values Level: one of 16 values Escape bit: one of 2 values
02	progressive_sequence	x	x	x	x	x	I	
03	chroma_format	x	x	x	x	x	I	See table E-10
04	horizontal_size_extension	x	x	x	x	x	D	(input picture size related)
05	vertical_size_extension	x	x	x	x	x	D	(input picture size related)
06	bit_rate_extension	x	x	x	x	x	D	(input picture size related)
07	vbv_buffer_size_extension	x	x	x	x	x	D	(input picture size related)
08	low_delay	x	x	x	x	x	I	
09	frame_rate_extension_n	x	x	x	x	x	I	Set to 0 for all Profiles
10	frame_rate_extension_d	x	x	x	x	x	I	Set to 0 for all Profiles

2

3

Table E-10. permissible chroma_format (independent of level)

Profile	Chroma Format
SIMPLE	4:2:0
MAIN	4:2:0
SNR	4:2:0
SPATIAL	4:2:0
HIGH	4:2:0, 4:2:2

4

1

Table E-11. Sequence display extension elements (See Table E-2 for Status)

	Status					Type
	HIGH					
#	SPATIAL					Parameter range in Level
	SNR					
#	MAIN					
	SIMPLE					
#	Syntactic elements					
01	video_format					P
02	colour_description					P (input format related)
03	colour primaries					
04	transfer_characteristics					
05	matrix_coefficients					
06	display_horizontal_size					P (input format related)
07	display_vertical_size					P (input format related)

2

1

Table E-12. Sequence scalable extension

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	scalable_mode	o	o	x	x	x	I	(spatial/snr/temporal scalability)
02	layer_id	o	o	x	x	x	I	
	if(spatial scalable)							
03	lower_layer_prediction_ horizontal_size	o	o	o	x	x	D	See tabel E-5 for luminance sampling density.
04	lower_layer_prediction_ vertical_size	o	o	o	x	x	D	See tabel E-5 for luminance sampling density.
05	horizontal_subsampling_ factor_m	o	o	o	x	x	I	
06	horizontal_subsampling_ factor_n	o	o	o	x	x	I	
07	vertical_subsampling_ factor_m	o	o	o	x	x	I	
08	vertical_subsampling_ factor_n	o	o	o	x	x	I	
	If(temporal scalable)							
09	picture_mux_enable	o	o	o	o	o	I	
10	mux_to_progressive_sequence	o	o	o	o	o	I	
11	picture_mux_order	o	o	o	o	o	I	
12	picture_mux_factor	o	o	o	o	o	I	

2

3

Table E-13. Group of pictures header

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	time_code	x	x	x	x	x	I	
02	closed_gop	x	x	x	x	x	I	
03	broken_link	x	x	x	x	x	I	

4

1

Table E-14. Picture header

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	temporal_reference	x	x	x	x	x	I	
02	picture_coding_type	x	x	x	x	x	I	See Table E-15
03	vbv_delay	x	x	x	x	x	I	
04	full_pel_forward_vector	x	x	x	x	x	I	"0" for MPEG-2
05	forward_f_code	x	x	x	x	x	I	"111" for MPEG-2
06	full_pel_backward_vector	x	x	x	x	x	I	"0" for MPEG-2
07	backward_f_code	x	x	x	x	x	I	"111" for MPEG-2
08	extra_information_picture	x	x	x	x	x	I	
09	Picture coding extension	x	x	x	x	x		
10	Quant matrix extension	(o)	(o)	(o)	(o)	(o)	I	
11	Picture pan-scan extension	(o)	(o)	(o)	(o)	(o)	P	
12	Picture spatial scalable extension	o	o	o	x	x	I	
13	Picture temporal scalable extension	o	o	o	o	o	I	

2

3

Table E-15. permissible picture_coding_type (independent of level)

Profile	picture_coding_type
SIMPLE	I,P
MAIN	I,P,B
SNR	I,P,B
SPATIAL	I,P,B
HIGH	I,P,B

4

1

Table E-16. Picture coding extension

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	forward_horizontal_f_code	x	x	x	x	x	I	[1:8]
02	forward_vertical_f_code	x	x	x	x	x	I	[1:5]
03	backward_horizontal_f_code	x	x	x	x	x	I	[1:8]
04	backward_vertical_f_code	x	x	x	x	x	I	[1:5]
05	intra_dc_precision	x	x	x	x	x	I	Simple, Main, SNR & Spatial profiles: [8:10] Next profile: [8:11]
06	picture_structure	x	x	x	x	x	I	
07	top_field_first	x	x	x	x	x	I	
08	frame_pred_frame_dct	x	x	x	x	x	I	
09	concealment_motion_vectors	x	x	x	x	x	I	
10	q_scale_type	x	x	x	x	x	I	
11	intra_vlc_format	x	x	x	x	x	I	
12	alternate_scan	x	x	x	x	x	I	
13	repeat_first_field	x	x	x	x	x	I	
14	chroma_420_type	x	x	x	x	x	P	
15	progressive_frame	x	x	x	x	x	P	
16	composite_display_flag	x	x	x	x	x	P	
17	v_axis							
18	field_sequence							
19	sub_carrier							
20	burst_amplitude							
21	sub_carrier_phase							

1

Table E-17. Quant matrix extension

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	load_intra_quantiser_matrix	x	x	x	x	x	I	
02	intra_quantiser_matrix[64]	x (c)	x (c)	x (c)	x (c)	x (c)	I	
03	load_non_intra_quantiser_matrix	x	x	x	x	x	I	
04	non_intra_quantiser_matrix[64]	x (c)	x (c)	x (c)	x (c)	x (c)	I	
05	load_chroma_intra_quantiser_matrix	o	o	o	o	x	I	
06	chroma_intra_quantiser_matrix[64]	o	o	o	o	x (c)	I	
07	load_chroma_non_intra_quantiser_matrix	o	o	o	o	x	I	
08	chroma_non_intra_quantiser_matrix[64]	o	o	o	o	x (c)	I	

2

3

Table E-18. Picture display extension. (See Table E-14 for Status)

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	frame_centre_horizontal_offset						P	(input format related)
02	frame_centre_vertical_offset						P	(input format related)

4

1

Table E-19. Picture temporal scalable extension

	Status						Type
	HIGH						
	SPATIAL						
	SNR						
	MAIN						
	SIMPLE						
#	Syntactic elements						Parameter range in Level
01	reference_select_code	o	o	o	o	o	I
02	forward_temporal_reference	o	o	o	o	o	I
03	backward_temporal_reference	o	o	o	o	o	I

2

3

Table E-20. Picture spatial scalable extension

	Status						Type
	HIGH						
	SPATIAL						
	SNR						
	MAIN						
	SIMPLE						
#	Syntactic elements						Parameter range in Level
01	lower_layer_temporal_reference	o	o	o	x	x	I
02	lower_layer_horizontal_offset	o	o	o	x	x	D (input format related)
03	lower_layer_vertical_offset	o	o	o	x	x	D (input format related)
04	spatial_temporal_weight_code_ table_index	o	o	o	x	x	I
05	lower_layer_progressive_frame	o	o	o	x	x	I
06	lower_layer_deinterlaced_field_ select	o	o	o	x	x	I

4

1

Table E-21. Slice layer

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	slice_vertical_position_extension	x	x	x	x	x	D	(input format related)
02	priority_breakpoint	o	o	o	o	o	I	Only required for data partitioning
03	quantiser_scale_code	x	x	x	x	x	I	
04	intra_slice	x	x	x	x	x	I	
05	extra_information_slice	x	x	x	x	x	I	decoder may skip this data
06	macroblock							

2

3

Table E-22. Macroblock layer

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	macroblock_escape	x	x	x	x	x	I	
02	macroblock_address_increment	x	x	x	x	x	I	
03	macroblock_modes()	x	x	x	x	x	I	
04	quantiser_scale_code	x	x	x	x	x	I	
05	motion_vectors() for forward	x	x	x	x	x	I	
06	motion_vectors() for backward	o	x	x	x	x	I	
07	coded_block_pattern()	x	x	x	x	x	I	
08	block(i)	x	x	x	x	x	I	

4

1

Table E-23. Macroblock modes

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	macroblock_type	x	x	x	x	x	I	
02	spatial_temporal_weight_code	o	o	o	x	x	I	
03	frame_motion_type	x	x	x	x	x	I	(01: Field-based prediction) (10: Frame-based prediction) (11: Dual-prime)
		01	01	01	01	01		
		10	10	10	10	10		
04	field_motion_type	x	x	x	x	x	I	(01: Field-based prediction) (10: 16x8 MC) (11: Dual-prime)
		01	01	01	01	01		
		10	10	10	10	10		
05	dct_type	x	x	x	x	x	I	

2

3

Table E-24. Motion vectors

#	Syntactic elements	Status					Type	Parameter range in Level
		HIGH						
		SPATIAL						
		SNR						
		MAIN						
		SIMPLE						
01	motion_vertical_field_select	x	x	x	x	x	I	
02	motion_vector()	x	x	x	x	x	I	

4

1

Table E-25. Motion vector

	Status						Type
	HIGH						
	SPATIAL						
	SNR						
	MAIN						
	SIMPLE						
#	Syntactic elements						Parameter range in Level
01	motion_horizontal_code	x	x	x	x	x	I
02	motion_horizontal_r	x	x	x	x	x	I
03	dmv_horizontal	x	x	x	x	x	I
04	motion_vertical_code	x	x	x	x	x	I
05	motion_vertical_r	x	x	x	x	x	I
06	dmv_vertical	x	x	x	x	x	I

2

3

Table E-26. Coded Block Pattern

	Status						Type
	HIGH						
	SPATIAL						
	SNR						
	MAIN						
	SIMPLE						
#	Syntactic elements						Parameter range in Level
01	coded_block_pattern_420	x	x	x	x	x	I
02	coded_block_pattern_1	o	o	o	o	x	I (for 4:2:2)
03	coded_block_pattern_2	o	o	o	o	o	I (for 4:4:4)

4

5

Table E-27. Block layer

	Status						Type
	HIGH						
	SPATIAL						
	SNR						
	MAIN						
	SIMPLE						
#	Syntactic elements						Parameter range in Level
01	DCT coefficients	x	x	x	x	x	I
02	End of block	x	x	x	x	x	I

6

1

Table E-28. Forward compatibility between different profiles and levels

Bitstream	Decoder										
	HP @ HL	HP @ H-14	HP @ ML	Spatial @ H-14	SNR @ ML	SNR @ LL	MP @ HL	MP @ H-14	MP @ ML	MP @ LL	SP @ ML
HP@HL	X										
HP@H-14	X	X									
HP@ML	X	X	X								
Spat.@H-14	X	X		X							
SNR @ML	X	X	X	X	X						
SNR @LL	X	X	X	X	X	X					
MP@HL	X						X				
MP@H-14	X	X		X			X	X			
MP@ML	X	X	X	X	X		X	X	X		
MP@LL	X	X	X	X	X	X	X	X	X	X	X*
SP@ML	X	X	X	X	X		X	X	X		X

X indicates the decoder shall be able to decode the bitstream.
 * Note that SP@ML decoders are required to decode MP@LL bitstreams.

2

3 Note- For Profiles and Levels which obey a hierarchical structure, it is recommended that the bit
 4 stream should contain the **profile_and_level_id** of the "simplest" decoder which is
 5 capable of successfully decoding the bit stream. In the case where the
 6 **profile_and_level_id** Escape bit = 0, this will be the numerically largest of the possible
 7 valid values of **profile_and_level_id**.

8

1
2
3
4
5
6
7
8

Annex F

Patent statements

(This annex does not form an integral part of this Recommendation | International Standard)

The following table summarises the formal patent statements received and indicates the parts of the MPEG-2 standard to which the statement applies.

The list includes all the companies that previously submitted the informal statement, but if no "X" is present it means that no formal statement was received from that company.

Company	V	A	S
AT&T	X	X	X
BBC Research Department			
Bellcore	X		
Belgian Science Policy Office	X	X	X
BOSCH	X	X	X
CCETT			
CSELT	X		
David Sarnoff Research Center	X	X	X
Deutsche Thomson-Brandt GmbH	X	X	X
France Telecom CNET			
Fraunhofer Gesellschaft		X	X
GC Technology Corporation	X	X	X
General Instruments			
Goldstar			
Hitachi, Ltd.			
International Business Machines Corporation	X	X	X
IRT		X	
KDD	X		
Massachusetts Institute of Technology	X	X	X
Matsushita Electric Industrial Co., Ltd.	X	X	X
Mitsubishi Electric Corporation			
National Transcommunications Limited			
NEC Corporation		X	
Nippon Hoso Kyokai	X		
Nippon Telegraph and Telephone	X		
Nokia Research Center	X		
Norwegian Telecom Research	X		
Philips Consumer Electronics	X	X	X
OKI			
Qualcomm Incorporated	X		
continued...			

9

Company	V	A	S
Royal PTT Nederland N.V., PTT Research (NL)	X	X	X
Samsung Electronics			
Scientific Atlanta	X	X	X
Siemens AG	X		
Sharp Corporation			
Sony Corporation			
Texas Instruments			
Thomson Consumer Electronics			
Toshiba Corporation	X		
TV/Com	X	X	X
Victor Company of Japan Limited			

1

2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

Annex G

Bibliography

(This annex does not form an integral part of this Recommendation | International Standard)

- 1 Arun N. Netravali & Barry G. Haskell "Digital Pictures, representation and compression"
Plenum Press, 1988
- 2 Didier Le Gall "MPEG: A Video Compression Standard for Multimedia Applications" Trans.
ACM, April 1991
- 3 C Loeffler, A Ligtenberg, G S Moschytz "Practical fast 1-D DCT algorithms with 11
multiplications" Proceedings IEEE ICASSP-89, Vol. 2, pp 988-991, Feb. 1989
- 4 See the Normative Reference for ITU-R Rec 601 (formerly CCIR Rec 601)
- 5 See the Normative Reference for IEC Standard Publication 461
- 6 See the Normative Reference for ITU-T Rec. H.261
- 7 See the Normative reference for IEEE Standard Specification P1180-1990
- 8 ISO/IEC 10918-1 | ITU-T T.81 (JPEG)
- 9 E Viscito and C Gonzales "A Video Compression Algorithm with Adaptive Bit Allocation and
Quantization", Proc SPIE Visual Communications and Image Proc '91 Boston MA November
10-15 Vol. 1605 205, 1991
- 10 A Puri and R Aravind "Motion Compensated Video Coding with Adaptive Perceptual
Quantization", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 1 pp 351
Dec. 1991.
- 11 C. Gonzales and E. Viscito, "Flexibly scalable digital video coding". Image Communications,
Vol. 5, Nos. 1-2, February 1993
- 12 A.W.Johnson, T.Sikora and T.K. Tan, "Filters for Drift Reduction in Frequency Scalable
Video Coding Schemes" <Transmitted for publication to Electronic Letters.>
- 13 R.Mokry and D.Anastassiou, "Minimal Error Drift in Frequency Scalability for Motion-
Compensated DCT Coding". IEEE Transactions on Circuits and Systems for Video
Technology, <accepted for publication>
- 14 K.N. Ngan, J. Arnold, T. Sikora, T.K. Tan and A.W. Johnson. "Frequency Scalability
Experiments for MPEG-2 Standard". Asia-Pacific Conference on Communications, Korea,
August 1993.
- 15 T. Sikora, T.K. Tan and K.N. Ngan, "A Performance Comparison of Frequency Domain
Pyramid Scalable Coding Schemes Within the MPEG Framework". Proc. PCS, Picture
Coding Symposium, Lausanne, pp. 16.1 - 16.2, Switzerland March 1993.
- 16 Masahiro Iwahashi, "Motion Compensation Technique for 2:1 Scaled-down Moving
Pictures". 8-14, Picture Coding Symposium '93.
- 17 Sikora, T. and Pang, K., "Experiments with Optimal Block-Overlapping Filters for Cell Loss
Concealment in Packet Video", Proc. IEEE Visual Signal Processing and Communications
Workshop, Melbourne, 21-22 Sept. 1993, pp. 247-250.
- 18 A. Puri "Video Coding Using the MPEG-2 Compression Standard", <to appear> Proc SPIE
Visual Communications and Image Proc '93 Boston MA November,1993.
- 19 A. Puri and A. Wong "Spatial Domain Resolution Scalable Video Coding", <to appear> Proc
SPIE Visual Communications and Image Proc '93 Boston MA November,1993.