
**Information technology — Coding of
audio-visual objects —**

**Part 10:
Advanced Video Coding**

*Technologies de l'information — Codage des objets audiovisuels —
Partie 10: Codage visuel avancé*



Reference number
ISO/IEC 14496-10:2014(E)

Documents
Document Sharing Hub
© ISO/IEC 2014



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

0	Introduction.....	xvii
0.1	Prologue	xvii
0.2	Purpose.....	xvii
0.3	Applications	xvii
0.4	Publication and versions of this Specification.....	xvii
0.5	Profiles and levels	xviii
0.6	Overview of the design characteristics.....	xix
0.6.1	Predictive coding	xix
0.6.2	Coding of progressive and interlaced video	xx
0.6.3	Picture partitioning into macroblocks and smaller partitions	xx
0.6.4	Spatial redundancy reduction	xx
0.7	How to read this Specification	xx
1	Scope.....	1
2	Normative references	1
3	Definitions	1
4	Abbreviations	9
5	Conventions	10
5.1	Arithmetic operators.....	10
5.2	Logical operators.....	11
5.3	Relational operators.....	11
5.4	Bit-wise operators	11
5.5	Assignment operators	11
5.6	Range notation	12
5.7	Mathematical functions.....	12
5.8	Order of operation precedence	13
5.9	Variables, syntax elements, and tables	14
5.10	Text description of logical operations	14
5.11	Processes	15
6	Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships.....	16
6.1	Bitstream formats	16
6.2	Source, decoded, and output picture formats	16
6.3	Spatial subdivision of pictures and slices	21
6.4	Inverse scanning processes and derivation processes for neighbours	22
6.4.1	Inverse macroblock scanning process	22
6.4.2	Inverse macroblock partition and sub-macroblock partition scanning process	23
6.4.3	Inverse 4x4 luma block scanning process	25
6.4.4	Inverse 4x4 Cb or Cr block scanning process for ChromaArrayType equal to 3	25
6.4.5	Inverse 8x8 luma block scanning process	25
6.4.6	Inverse 8x8 Cb or Cr block scanning process for ChromaArrayType equal to 3	26
6.4.7	Inverse 4x4 chroma block scanning process	26
6.4.8	Derivation process of the availability for macroblock addresses	26
6.4.9	Derivation process for neighbouring macroblock addresses and their availability	26
6.4.10	Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames	27
6.4.11	Derivation processes for neighbouring macroblocks, blocks, and partitions	28
6.4.12	Derivation process for neighbouring locations.....	33
6.4.13	Derivation processes for block and partition indices.....	35
7	Syntax and semantics.....	37
7.1	Method of specifying syntax in tabular form	37
7.2	Specification of syntax functions, categories, and descriptors	38
7.3	Syntax in tabular form.....	40
7.3.1	NAL unit syntax	40

7.3.2	Raw byte sequence payloads and RBSP trailing bits syntax.....	40
7.3.3	Slice header syntax	49
7.3.4	Slice data syntax	54
7.3.5	Macroblock layer syntax.....	55
7.4	Semantics.....	62
7.4.1	NAL unit semantics	62
7.4.2	Raw byte sequence payloads and RBSP trailing bits semantics	72
7.4.3	Slice header semantics	86
7.4.4	Slice data semantics	98
7.4.5	Macroblock layer semantics.....	98
8	Decoding process.....	111
8.1	NAL unit decoding process	112
8.2	Slice decoding process.....	113
8.2.1	Decoding process for picture order count	113
8.2.2	Decoding process for macroblock to slice group map	117
8.2.3	Decoding process for slice data partitions	121
8.2.4	Decoding process for reference picture lists construction.....	121
8.2.5	Decoded reference picture marking process	128
8.3	Intra prediction process.....	133
8.3.1	Intra_4x4 prediction process for luma samples.....	133
8.3.2	Intra_8x8 prediction process for luma samples.....	140
8.3.3	Intra_16x16 prediction process for luma samples.....	148
8.3.4	Intra prediction process for chroma samples.....	150
8.3.5	Sample construction process for I_PCM macroblocks	154
8.4	Inter prediction process.....	155
8.4.1	Derivation process for motion vector components and reference indices.....	158
8.4.2	Decoding process for Inter prediction samples	170
8.4.3	Derivation process for prediction weights	179
8.5	Transform coefficient decoding process and picture construction process prior to deblocking filter process..	182
8.5.1	Specification of transform decoding process for 4x4 luma residual blocks.....	182
8.5.2	Specification of transform decoding process for luma samples of Intra_16x16 macroblock prediction mode	183
8.5.3	Specification of transform decoding process for 8x8 luma residual blocks.....	184
8.5.4	Specification of transform decoding process for chroma samples.....	184
8.5.5	Specification of transform decoding process for chroma samples with ChromaArrayType equal to 3	186
8.5.6	Inverse scanning process for 4x4 transform coefficients and scaling lists.....	186
8.5.7	Inverse scanning process for 8x8 transform coefficients and scaling lists.....	187
8.5.8	Derivation process for chroma quantisation parameters	189
8.5.9	Derivation process for scaling functions.....	189
8.5.10	Scaling and transformation process for DC transform coefficients for Intra_16x16 macroblock type	191
8.5.11	Scaling and transformation process for chroma DC transform coefficients	191
8.5.12	Scaling and transformation process for residual 4x4 blocks	193
8.5.13	Scaling and transformation process for residual 8x8 blocks	196
8.5.14	Picture construction process prior to deblocking filter process.....	200
8.5.15	Intra residual transform-bypass decoding process	201
8.6	Decoding process for P macroblocks in SP slices or SI macroblocks	202
8.6.1	SP decoding process for non-switching pictures.....	202
8.6.2	SP and SI slice decoding process for switching pictures	205
8.7	Deblocking filter process	207
8.7.1	Filtering process for block edges	211
8.7.2	Filtering process for a set of samples across a horizontal or vertical block edge.....	213
9	Parsing process	219
9.1	Parsing process for Exp-Golomb codes	219
9.1.1	Mapping process for signed Exp-Golomb codes	221
9.1.2	Mapping process for coded block pattern	221
9.2	CAVLC parsing process for transform coefficient levels	224
9.2.1	Parsing process for total number of non-zero transform coefficient levels and number of trailing ones..	225
9.2.2	Parsing process for level information	229
9.2.3	Parsing process for run information.....	230

9.2.4	Combining level and run information	233
9.3	CABAC parsing process for slice data	233
9.3.1	Initialisation process	235
9.3.2	Binarization process	259
9.3.3	Decoding process flow	268
9.3.4	Arithmetic encoding process (informative)	290
Annex A (normative)	Profiles and levels	298
A.1	Requirements on video decoder capability	298
A.2	Profiles	298
A.2.1	Baseline profile	298
A.2.2	Main profile	299
A.2.3	Extended profile	299
A.2.4	High profile	300
A.2.5	High 10 profile	301
A.2.6	High 4:2:2 profile	301
A.2.7	High 4:4:4 Predictive profile	302
A.2.8	High 10 Intra profile	302
A.2.9	High 4:2:2 Intra profile	303
A.2.10	High 4:4:4 Intra profile	303
A.2.11	CAVLC 4:4:4 Intra profile	304
A.3	Levels	304
A.3.1	Level limits common to the Baseline, Constrained Baseline, Main, and Extended profiles	304
A.3.2	Level limits common to the High, Progressive High, High 10, High 4:2:2, High 4:4:4 Predictive, High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, and CAVLC 4:4:4 Intra profiles	307
A.3.3	Profile-specific level limits	309
A.3.4	Effect of level limits on frame rate (informative)	314
A.3.5	Effect of level limits on maximum DPB size in units of frames (informative)	317
Annex B (normative)	Byte stream format	319
B.1	Byte stream NAL unit syntax and semantics	319
B.1.1	Byte stream NAL unit syntax	319
B.1.2	Byte stream NAL unit semantics	319
B.2	Byte stream NAL unit decoding process	320
B.3	Decoder byte-alignment recovery (informative)	320
Annex C (normative)	Hypothetical reference decoder	322
C.1	Operation of coded picture buffer (CPB)	326
C.1.1	Timing of bitstream arrival	326
C.1.2	Timing of coded picture removal	327
C.2	Operation of the decoded picture buffer (DPB)	328
C.2.1	Decoding of gaps in frame_num and storage of "non-existing" frames	329
C.2.2	Picture decoding and output	329
C.2.3	Removal of pictures from the DPB before possible insertion of the current picture	330
C.2.4	Current decoded picture marking and storage	331
C.3	Bitstream conformance	332
C.4	Decoder conformance	334
C.4.1	Operation of the output order DPB	335
C.4.2	Decoding of gaps in frame_num and storage of "non-existing" pictures	335
C.4.3	Picture decoding	335
C.4.4	Removal of pictures from the DPB before possible insertion of the current picture	336
C.4.5	Current decoded picture marking and storage	337
Annex D (normative)	Supplemental enhancement information	341
D.1	SEI payload syntax	342
D.1.1	Buffering period SEI message syntax	344
D.1.2	Picture timing SEI message syntax	344
D.1.3	Pan-scan rectangle SEI message syntax	345
D.1.4	Filler payload SEI message syntax	346
D.1.5	User data registered by ITU-T Rec. T.35 SEI message syntax	346
D.1.6	User data unregistered SEI message syntax	346
D.1.7	Recovery point SEI message syntax	346
D.1.8	Decoded reference picture marking repetition SEI message syntax	347

D.1.9	Spare picture SEI message syntax	347
D.1.10	Scene information SEI message syntax	348
D.1.11	Sub-sequence information SEI message syntax	348
D.1.12	Sub-sequence layer characteristics SEI message syntax	348
D.1.13	Sub-sequence characteristics SEI message syntax	349
D.1.14	Full-frame freeze SEI message syntax	349
D.1.15	Full-frame freeze release SEI message syntax	349
D.1.16	Full-frame snapshot SEI message syntax	349
D.1.17	Progressive refinement segment start SEI message syntax	350
D.1.18	Progressive refinement segment end SEI message syntax	350
D.1.19	Motion-constrained slice group set SEI message syntax	350
D.1.20	Film grain characteristics SEI message syntax	351
D.1.21	Deblocking filter display preference SEI message syntax	351
D.1.22	Stereo video information SEI message syntax	352
D.1.23	Post-filter hint SEI message syntax	352
D.1.24	Tone mapping information SEI message syntax	353
D.1.25	Frame packing arrangement SEI message syntax	354
D.1.26	Reserved SEI message syntax	354
D.2	SEI payload semantics	354
D.2.1	Buffering period SEI message semantics	354
D.2.2	Picture timing SEI message semantics	355
D.2.3	Pan-scan rectangle SEI message semantics	360
D.2.4	Filler payload SEI message semantics	361
D.2.5	User data registered by ITU-T Rec. T.35 SEI message semantics	361
D.2.6	User data unregistered SEI message semantics	362
D.2.7	Recovery point SEI message semantics	362
D.2.8	Decoded reference picture marking repetition SEI message semantics	364
D.2.9	Spare picture SEI message semantics	364
D.2.10	Scene information SEI message semantics	366
D.2.11	Sub-sequence information SEI message semantics	368
D.2.12	Sub-sequence layer characteristics SEI message semantics	369
D.2.13	Sub-sequence characteristics SEI message semantics	370
D.2.14	Full-frame freeze SEI message semantics	372
D.2.15	Full-frame freeze release SEI message semantics	372
D.2.16	Full-frame snapshot SEI message semantics	372
D.2.17	Progressive refinement segment start SEI message semantics	372
D.2.18	Progressive refinement segment end SEI message semantics	373
D.2.19	Motion-constrained slice group set SEI message semantics	373
D.2.20	Film grain characteristics SEI message semantics	374
D.2.21	Deblocking filter display preference SEI message semantics	380
D.2.22	Stereo video information SEI message semantics	382
D.2.23	Post-filter hint SEI message semantics	382
D.2.24	Tone mapping information SEI message semantics	383
D.2.25	Frame packing arrangement SEI message semantics	385
D.2.26	Reserved SEI message semantics	395
Annex E (normative) Video usability information		396
E.1	VUI syntax	397
E.1.1	VUI parameters syntax	397
E.1.2	HRD parameters syntax	398
E.2	VUI semantics	398
E.2.1	VUI parameters semantics	398
E.2.2	HRD parameters semantics	411
Annex F (informative) Patent Rights		413
Annex G (normative) Scalable video coding		415
G.1	Scope	415
G.2	Normative references	415
G.3	Definitions	415
G.4	Abbreviations	419
G.5	Conventions	419

G.6	Source, coded, decoded and output data formats, scanning processes, neighbouring and reference layer relationships	419
G.6.1	Derivation process for reference layer macroblocks	419
G.6.2	Derivation process for reference layer partitions.....	422
G.6.3	Derivation process for reference layer sample locations in resampling	423
G.6.4	SVC derivation process for macroblock and sub-macroblock partition indices.....	425
G.7	Syntax and semantics	425
G.7.1	Method of specifying syntax in tabular form	425
G.7.2	Specification of syntax functions, categories, and descriptors	425
G.7.3	Syntax in tabular form	425
G.7.4	Semantics	437
G.8	SVC decoding process	471
G.8.1	SVC initialisation and decoding processes.....	472
G.8.2	SVC reference picture lists construction and decoded reference picture marking process	492
G.8.3	SVC intra decoding processes	503
G.8.4	SVC Inter prediction process.....	513
G.8.5	SVC transform coefficient decoding and sample array construction processes	526
G.8.6	Resampling processes for prediction data, intra samples, and residual samples	543
G.8.7	SVC deblocking filter processes	573
G.8.8	Specification of bitstream subsets	585
G.9	Parsing process	586
G.9.1	Alternative parsing process for coded block pattern	587
G.9.2	Alternative CAVLC parsing process for transform coefficient levels.....	588
G.9.3	Alternative CABAC parsing process for slice data in scalable extension	592
G.10	Profiles and levels	595
G.10.1	Profiles	595
G.10.2	Levels	598
G.11	Byte stream format	603
G.12	Hypothetical reference decoder	603
G.13	Supplemental enhancement information	603
G.13.1	SEI payload syntax	604
G.13.2	SEI payload semantics.....	610
G.14	Video usability information.....	638
G.14.1	SVC VUI parameters extension syntax	639
G.14.2	SVC VUI parameters extension semantics.....	639
Annex H (normative)	Multiview video coding	642
H.1	Scope	642
H.2	Normative references	642
H.3	Definitions.....	642
H.4	Abbreviations	644
H.5	Conventions.....	644
H.6	Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships.....	644
H.7	Syntax and semantics	645
H.7.1	Method of specifying syntax in tabular form	645
H.7.2	Specification of syntax functions, categories, and descriptors	645
H.7.3	Syntax in tabular form	645
H.7.4	Semantics	649
H.8	MVC decoding process	663
H.8.1	MVC decoding process for picture order count	663
H.8.2	MVC decoding process for reference picture lists construction	664
H.8.3	MVC decoded reference picture marking process.....	668
H.8.4	MVC inter prediction and inter-view prediction process	668
H.8.5	Specification of bitstream subsets	669
H.9	Parsing process.....	673
H.10	Profiles and levels	673
H.10.1	Profiles	673
H.10.2	Levels	674
H.11	Byte stream format	678
H.12	MVC hypothetical reference decoder.....	678
H.13	MVC SEI messages.....	678

H.13.1	SEI message syntax.....	678
H.13.2	SEI message semantics	684
H.14	Video usability information.....	696
H.14.1	MVC VUI parameters extension syntax	697
H.14.2	MVC VUI parameters extension semantics	697

LIST OF FIGURES

Figure 6-1	– Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame.....	18
Figure 6-2	– Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields.....	19
Figure 6-3	– Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame.....	19
Figure 6-4	– Nominal vertical and horizontal sampling locations of 4:2:2 samples top and bottom fields.....	20
Figure 6-5	– Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a frame.....	20
Figure 6-6	– Nominal vertical and horizontal sampling locations of 4:4:4 samples top and bottom fields.....	21
Figure 6-7	– A picture with 11 by 9 macroblocks that is partitioned into two slices.....	22
Figure 6-8	– Partitioning of the decoded frame into macroblock pairs	22
Figure 6-9	– Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans	24
Figure 6-10	– Scan for 4x4 luma blocks.....	25
Figure 6-11	– Scan for 8x8 luma blocks.....	25
Figure 6-12	– Neighbouring macroblocks for a given macroblock	27
Figure 6-13	– Neighbouring macroblocks for a given macroblock in MBAFF frames.....	28
Figure 6-14	– Determination of the neighbouring macroblock, blocks, and partitions (informative)	29
Figure 7-1	– Structure of an access unit not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive.....	70
Figure 8-1	– Intra_4x4 prediction mode directions (informative)	135
Figure 8-2	– Example for temporal direct-mode motion vector inference (informative)	167
Figure 8-3	– Directional segmentation prediction (informative).....	168
Figure 8-4	– Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation	174
Figure 8-5	– Fractional sample position dependent variables in chroma interpolation and surrounding integer position samples A, B, C, and D	177
Figure 8-6	– Assignment of the indices of dcY to luma4x4BlkIdx	183
Figure 8-7	– Assignment of the indices of dcC to chroma4x4BlkIdx: (a) ChromaArrayType equal to 1, (b) ChromaArrayType equal to 2.....	185
Figure 8-8	– 4x4 block scans. (a) Zig-zag scan. (b) Field scan (informative)	187
Figure 8-9	– 8x8 block scans. (a) 8x8 zig-zag scan. (b) 8x8 field scan (informative).....	188
Figure 8-10	– Boundaries in a macroblock to be filtered	208
Figure 8-11	– Convention for describing samples across a 4x4 block horizontal or vertical boundary	212
Figure 9-1	– Illustration of CABAC parsing process for a syntax element SE (informative)	235
Figure 9-2	– Overview of the arithmetic decoding process for a single bin (informative).....	284
Figure 9-3	– Flowchart for decoding a decision	285

Figure 9-4 – Flowchart of renormalization	288
Figure 9-5 – Flowchart of bypass decoding process	289
Figure 9-6 – Flowchart of decoding a decision before termination	290
Figure 9-7 – Flowchart for encoding a decision.....	292
Figure 9-8 – Flowchart of renormalization in the encoder.....	293
Figure 9-9 – Flowchart of PutBit(B).....	294
Figure 9-10 – Flowchart of encoding bypass	295
Figure 9-11 – Flowchart of encoding a decision before termination	296
Figure 9-12 – Flowchart of flushing at termination	296
Figure C-1 – Structure of byte streams and NAL unit streams for HRD conformance checks.....	322
Figure C-2 – HRD buffer model	324
Figure D-1 – Rearrangement and upconversion of checkerboard interleaving (frame_packing_arrangement_type equal to 0).....	390
Figure D-2 – Rearrangement and upconversion of column interleaving with frame_packing_arrangement_type equal to 1, quincunx_sampling_flag equal to 0, and (x, y) equal to (0, 0) or (4, 8) for both constituent frames	390
Figure D-3 – Rearrangement and upconversion of column interleaving with frame_packing_arrangement_type equal to 1, quincunx_sampling_flag equal to 0, (x, y) equal to (0, 0) or (4, 8) for constituent frame 0 and (x, y) equal to (12, 8) for constituent frame 1	391
Figure D-4 – Rearrangement and upconversion of row interleaving with frame_packing_arrangement_type equal to 2, quincunx_sampling_flag equal to 0, and (x, y) equal to (0, 0) or (8, 4) for both constituent frames	391
Figure D-5 – Rearrangement and upconversion of row interleaving with frame_packing_arrangement_type equal to 2, quincunx_sampling_flag equal to 0, (x, y) equal to (0, 0) or (8, 4) for constituent frame 0, and (x, y) equal to (8, 12) for constituent frame 1	392
Figure D-6 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0, and (x, y) equal to (0, 0) or (4, 8) for both constituent frames.....	392
Figure D-7 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0, (x, y) equal to (12, 8) for constituent frame 0, and (x, y) equal to (0, 0) or (4, 8) for constituent frame 1	393
Figure D-8 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0, and (x, y) equal to (0, 0) or (8, 4) for both constituent frames.....	393
Figure D-9 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0, (x, y) equal to (8, 12) for constituent frame 0, and (x, y) equal to (0, 0) or (8, 4) for constituent frame 1	394
Figure D-10 – Rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling (frame_packing_arrangement_type equal to 3 with quincunx_sampling_flag equal to 1).....	394
Figure D-11 – Rearrangement of a temporal interleaving frame arrangement (frame_packing_arrangement_type equal to 5)	395
Figure E-1 – Location of chroma samples for top and bottom fields for chroma_format_idc equal to 1 (4:2:0 chroma format) as a function of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field	407

LIST OF TABLES

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)	13
Table 6-1 – SubWidthC, and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag.....	17
Table 6-2 – Specification of input and output assignments for subclauses 6.4.11.1 to 6.4.11.7.....	29
Table 6-3 – Specification of mbAddrN	34
Table 6-4 – Specification of mbAddrN and yM	35
Table 7-1 – NAL unit type codes, syntax element categories, and NAL unit type classes.....	63
Table 7-2 – Assignment of mnemonic names to scaling list indices and specification of fall-back rule.....	75
Table 7-3 – Specification of default scaling lists Default_4x4_Intra and Default_4x4_Inter	75
Table 7-4 – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter	76
Table 7-5 – Meaning of primary_pic_type.....	84
Table 7-6 – Name association to slice_type	87
Table 7-7 – modification_of_pic_nums_idc operations for modification of reference picture lists.....	93
Table 7-8 – Interpretation of adaptive_ref_pic_marking_mode_flag.....	95
Table 7-9 – Memory management control operation (memory_management_control_operation) values	96
Table 7-10 – Allowed collective macroblock types for slice_type.....	99
Table 7-11 – Macroblock types for I slices	100
Table 7-12 – Macroblock type with value 0 for SI slices	101
Table 7-13 – Macroblock type values 0 to 4 for P and SP slices.....	102
Table 7-14 – Macroblock type values 0 to 22 for B slices	103
Table 7-15 – Specification of CodedBlockPatternChroma values	105
Table 7-16 – Relationship between intra_chroma_pred_mode and spatial prediction modes.....	106
Table 7-17 – Sub-macroblock types in P macroblocks	107
Table 7-18 – Sub-macroblock types in B macroblocks	108
Table 8-1 – Refined slice group map type.....	118
Table 8-2 – Specification of Intra4x4PredMode[luma4x4BlkIdx] and associated names.....	134
Table 8-3 – Specification of Intra8x8PredMode[luma8x8BlkIdx] and associated names.....	141
Table 8-4 – Specification of Intra16x16PredMode and associated names	148
Table 8-5 – Specification of Intra chroma prediction modes and associated names	151
Table 8-6 – Specification of the variable colPic.....	160
Table 8-7 – Specification of PicCodingStruct(X).....	160
Table 8-8 – Specification of mbAddrCol, yM, and vertMvScale.....	162
Table 8-9 – Assignment of prediction utilization flags	164
Table 8-10 – Derivation of the vertical component of the chroma vector in field coding mode	170
Table 8-11 – Differential full-sample luma locations.....	175
Table 8-12 – Assignment of the luma prediction sample predPartLX _L [x _L , y _L]	176
Table 8-13 – Specification of mapping of idx to c _{ij} for zig-zag and field scan.....	187
Table 8-14 – Specification of mapping of idx to c _{ij} for 8x8 zig-zag and 8x8 field scan.....	188

Table 8-15 – Specification of QP_C as a function of qP_I	189
Table 8-16 – Derivation of offset dependent threshold variables α' and β' from indexA and indexB.....	216
Table 8-17 – Value of variable t'_{CO} as a function of indexA and bS	217
Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)	220
Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)	220
Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v).....	221
Table 9-4 – Assignment of codeNum to values of coded_block_pattern for macroblock prediction modes.....	222
Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)	227
Table 9-6 – Codeword table for level_prefix (informative).....	230
Table 9-7 – total_zeros tables for 4x4 blocks with tzVlcIndex 1 to 7.....	231
Table 9-8 – total_zeros tables for 4x4 blocks with tzVlcIndex 8 to 15.....	232
Table 9-9 – total_zeros tables for chroma DC 2x2 and 2x4 blocks	232
Table 9-10 – Tables for run_before.....	233
Table 9-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process	236
Table 9-12 – Values of variables m and n for ctxIdx from 0 to 10	237
Table 9-13 – Values of variables m and n for ctxIdx from 11 to 23	238
Table 9-14 – Values of variables m and n for ctxIdx from 24 to 39	238
Table 9-15 – Values of variables m and n for ctxIdx from 40 to 53	238
Table 9-16 – Values of variables m and n for ctxIdx from 54 to 59, and 399 to 401.....	239
Table 9-17 – Values of variables m and n for ctxIdx from 60 to 69	239
Table 9-18 – Values of variables m and n for ctxIdx from 70 to 104	240
Table 9-19 – Values of variables m and n for ctxIdx from 105 to 165	241
Table 9-20 – Values of variables m and n for ctxIdx from 166 to 226	242
Table 9-21 – Values of variables m and n for ctxIdx from 227 to 275	243
Table 9-22 – Values of variables m and n for ctxIdx from 277 to 337	244
Table 9-23 – Values of variables m and n for ctxIdx from 338 to 398	245
Table 9-24 – Values of variables m and n for ctxIdx from 402 to 459	246
Table 9-25 – Values of variables m and n for ctxIdx from 460 to 483	247
Table 9-26 – Values of variables m and n for ctxIdx from 484 to 571	248
Table 9-27 – Values of variables m and n for ctxIdx from 572 to 659	250
Table 9-28 – Values of variables m and n for ctxIdx from 660 to 717	252
Table 9-29 – Values of variables m and n for ctxIdx from 718 to 775	253
Table 9-30 – Values of variables m and n for ctxIdx from 776 to 863	254
Table 9-31 – Values of variables m and n for ctxIdx from 864 to 951	256
Table 9-32 – Values of variables m and n for ctxIdx from 952 to 1011	258
Table 9-33 – Values of variables m and n for ctxIdx from 1012 to 1023	259
Table 9-34 – Syntax elements and associated types of binarization, maxBinIdxCtx, and ctxIdxOffset.....	261
Table 9-35 – Bin string of the unary binarization (informative)	264

Table 9-36 – Binarization for macroblock types in I slices	266
Table 9-37 – Binarization for macroblock types in P, SP, and B slices	267
Table 9-38 – Binarization for sub-macroblock types in P, SP, and B slices.....	268
Table 9-39 – Assignment of ctxIdxInc to binIdx for all ctxIdxOffset values except those related to the syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1.....	270
Table 9-40 – Assignment of ctxIdxBlockCatOffset to ctxBlockCat for syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1	271
Table 9-41 – Specification of ctxIdxInc for specific values of ctxIdxOffset and binIdx.....	280
Table 9-42 – Specification of ctxBlockCat for the different blocks	281
Table 9-43 – Mapping of scanning position to ctxIdxInc for ctxBlockCat = 5, 9, or 13.....	282
Table 9-44 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx.....	286
Table 9-45 – State transition table	287
Table A-1 – Level limits.....	307
Table A-2 – Specification of cpbBrVclFactor and cpbBrNalFactor.....	310
Table A-3 – Baseline and Constrained Baseline profile level limits	311
Table A-4 – Main, High, Progressive High, High 10, High 4:2:2, High 4:4:4 Predictive, High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, and CAVLC 4:4:4 Intra profile level limits	312
Table A-5 – Extended profile level limits	313
Table A-6 – Maximum frame rates (frames per second) for some example frame sizes.....	314
Table A-7 – Maximum DPB size (frames) for some example frame sizes.....	317
Table D-1 – Interpretation of pic_struct	357
Table D-2 – Mapping of ct_type to source picture scan.....	358
Table D-3 – Definition of counting_type values	359
Table D-4 – scene_transition_type values	367
Table D-5 – model_id values.....	375
Table D-6 – blending_mode_id values.....	376
Table D-7 – filter_hint_type values	383
Table D-8 – Definition of frame_packing_arrangement_type.....	386
Table D-9 – Definition of content_interpretation_type	387
Table E-1 – Meaning of sample aspect ratio indicator	399
Table E-2 – Meaning of video_format	400
Table E-3 – Colour primaries	401
Table E-4 – Transfer characteristics	402
Table E-5 – Matrix coefficients.....	405
Table E-6 – Divisor for computation of $\Delta t_{fi,dpb}(n)$	408
Table F-1 – Organisations providing patent rights licensing notices.....	413
Table G-1 – Name association to slice_type for NAL units with nal_unit_type equal to 20.	456
Table G-2 – Interpretation of adaptive_ref_base_pic_marking_mode_flag.....	465
Table G-3 – Memory management base control operation (memory_management_base_control_operation) values	466

Table G-4 – Allowed collective macroblock types for slice_type.	469
Table G-5 – Inferred macroblock type I_BL for EI slices.	469
Table G-6 – Scale values cS for transform coefficient level scaling.....	534
Table G-7 – Macroblock type predictors mbTypeILPred	552
Table G-8 – Sub-macroblock type predictors subMbTypeILPred[mbPartIdx].....	552
Table G-9 – 16-phase luma interpolation filter for resampling in Intra_Base prediction	562
Table G-10 – Mapping of (nX, nY) to coeffTokenIdx and vice versa	589
Table G-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process	593
Table G-12 – Values of variables m and n for ctxIdx from 1024 to 1026	593
Table G-13 – Values of variables m and n for ctxIdx from 1027 to 1030	593
Table G-14 – Syntax elements and associated types of binarization, maxBinIdxCtx, and ctxIdxOffset	594
Table G-15 – Assignment of ctxIdxInc to binIdx for the ctxIdxOffset values related to the syntax elements base_mode_flag and residual_prediction_flag.....	594
Table G-16 – Scalable Baseline profile level limits.....	603
Table G-17 – Specification of cpbBrVclFactor and cpbBrNalFactor	603
Table H-1 – modification_of_pic_nums_idc operations for modification of reference picture lists.....	662
Table H-2 – Association between camera parameter variables and syntax elements.	694

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-10 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with ITU T.

This part of ISO/IEC 14496 is technically aligned with ITU-T Rec. H.264 but is not published as identical text.

This eighth edition cancels and replaces the seventh edition (ISO/IEC 14496-10:2012), which has been technically revised. It also incorporates the Amendments ISO/IEC 14496-10:2012/Amd.1:2013 and ISO/IEC 14496-10:2012/Amd.2:2013, and the Technical Corrigendum ISO/IEC 14496-10:2012/Cor.1:2013.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference software*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*
- *Part 7: Optimized reference software for coding of audio-visual objects [Technical Report]*
- *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
- *Part 9: Reference hardware description [Technical Report]*
- *Part 10: Advanced Video Coding*
- *Part 11: Scene description and application engine*

- *Part 12: ISO base media file format*
- *Part 13: Intellectual Property Management and Protection (IPMP) extensions*
- *Part 14: MP4 file format*
- *Part 15: Carriage of network abstraction layer (NAL) unit structured video in ISO base media file format*
- *Part 16: Animation Framework eXtension (AFX)*
- *Part 17: Streaming text format*
- *Part 18: Font compression and streaming*
- *Part 19: Synthesized texture stream*
- *Part 20: Lightweight Application Scene Representation (LSeR) and Simple Aggregation Format (SAF)*
- *Part 21: MPEG-J Graphics Framework eXtensions (GFX)*
- *Part 22: Open Font Format*
- *Part 23: Symbolic Music Representation*
- *Part 24: Audio and systems interaction*
- *Part 25: 3D Graphics Compression Model*
- *Part 26: Audio conformance*
- *Part 27: 3D Graphics conformance*
- *Part 28: Composite font representation*
- *Part 29: Web video coding*
- *Part 30: Timed text and other visual overlays in ISO base media file format*

0 Introduction

This clause does not form an integral part of this Recommendation | International Standard.

0.1 Prologue

This subclause does not form an integral part of this Recommendation | International Standard.

As the costs for both processing power and memory have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industry standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Video Team (JVT) in 2001 for development of a new Recommendation | International Standard.

0.2 Purpose

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

0.3 Applications

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

CATV	Cable TV on optical networks, copper, etc.
DBS	Direct broadcast satellite video services
DSL	Digital subscriber line video services
DTTB	Digital terrestrial television broadcasting
ISM	Interactive storage media (optical disks, etc.)
MMM	Multimedia mailing
MSPN	Multimedia services over packet networks
RTC	Real-time conversational services (videoconferencing, videophone, etc.)
RVS	Remote video surveillance
SSM	Serial storage media (digital VTR, etc.)

0.4 Publication and versions of this Specification

This subclause does not form an integral part of this Recommendation | International Standard.

This Specification has been jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group. It is published as technically-aligned twin text in both organizations ITU-T and ISO/IEC.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first approved version of this Recommendation | International Standard.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 2 refers to the integrated text containing the corrections specified in the first technical corrigendum.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 3 refers to the integrated text containing both the first technical corrigendum (2004) and the first amendment, which is referred to as the "Fidelity range extensions".

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 4 refers to the integrated text containing the first technical corrigendum (2004), the first amendment (the "Fidelity range extensions"), and an additional technical corrigendum (2005).

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 5 refers to the integrated version 4 text with its specification of the High 4:4:4 profile removed.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 6 refers to the integrated version 5 text after its amendment to support additional colour space indicators.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 7 refers to the integrated version 6 text after its amendment to define five new profiles intended primarily for professional applications (the High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles) and two new types of supplemental enhancement information (SEI) messages (the post-filter hint SEI message and the tone mapping information SEI message).

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 8 refers to the integrated version 7 text after its amendment to specify scalable video coding in three profiles (Scalable Baseline, Scalable High, and Scalable High Intra profiles).

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 9 refers to the integrated version 8 text after applying the corrections specified in a third technical corrigendum.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 10 refers to the integrated version 9 text after its amendment to specify a profile for multiview video coding (the Multiview High profile) and to define additional SEI messages.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 11 refers to the integrated version 10 text after its amendment to define a new profile (the Constrained Baseline profile) intended primarily to enable implementation of decoders supporting only the common subset of capabilities supported in various previously-specified profiles.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 12 refers to the integrated version 11 text after its amendment to define a new profile (the Stereo High profile) for two-view video coding with support of interlaced coding tools and to specify an additional SEI message specified as the frame packing arrangement SEI message. The changes for versions 11 and 12 were processed as a single amendment in the ISO/IEC approval process.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 13 refers to the integrated version 12 text with various minor corrections and clarifications.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 14 refers to the integrated version 13 text after its amendment to define a new level (Level 5.2) supporting higher processing rates in terms of maximum macroblocks per second and a new profile (the Progressive High profile) to enable implementation of decoders supporting only the frame coding tools of the previously-specified High profile.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 15 refers to the integrated version 14 text with miscellaneous corrections and clarifications as specified in a fifth technical corrigendum.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 16 refers to the integrated version 15 text after its amendment to define three new profiles intended primarily for communication applications (the Constrained High, Scalable Constrained Baseline, and Scalable Constrained High profiles).

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 17 refers to the integrated version 16 text after its amendment to define additional supplemental enhancement information (SEI) message data, including the multiview view position SEI message, the display orientation SEI message, and two additional frame packing arrangement type indication values for the frame packing arrangement SEI message (the 2D and tiled arrangement type indication values).

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 18 refers to the integrated version 17 text after its amendment to specify the coding of depth signals, including the specification of an additional profile, the Multiview Depth High profile.

Rec. ITU-T H.264 | ISO/IEC 14496-10 version 19 refers to the integrated version 18 text after incorporating a correction to the sub-bitstream extraction process for multiview video coding.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 20 refers to the integrated version 19 text after its amendment to specify the combined coding of video view and depth enhancement, including the specification of an additional profile, the Enhanced Multiview Depth High profile.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 21 refers to the integrated version 20 text after its amendment to specify additional colorimetry identifiers and an additional model type in the tone mapping information SEI message.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 22 (the current Specification) refers to the integrated version 21 text after its amendment to specify multi-resolution frame-compatible (MFC) enhancement for stereoscopic video coding, including the specification of an additional profile, the MFC High profile.

0.5 Profiles and levels

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities, and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and these have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

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

A "profile" is a subset of the entire bitstream syntax that is specified by this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. In many applications, it is currently neither practical nor economic to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "levels" are specified within each profile. A level is a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively they may take the form of constraints on arithmetic combinations of values (e.g., picture width multiplied by picture height multiplied by number of pictures decoded per second).

Coded video content conforming to this Recommendation | International Standard uses a common syntax. In order to achieve a subset of the complete syntax, flags, parameters, and other syntax elements are included in the bitstream that signal the presence or absence of syntactic elements that occur later in the bitstream.

0.6 Overview of the design characteristics

This subclause does not form an integral part of this Recommendation | International Standard.

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image quality. With the exception of the transform bypass mode of operation for lossless coding in the High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles, and the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantised, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are combined with the quantised transform coefficient information and encoded using either variable length coding or arithmetic coding.

Scalable video coding is specified in Annex G allowing the construction of bitstreams that contain sub-bitstreams that conform to this Specification. For temporal bitstream scalability, i.e., the presence of a sub-bitstream with a smaller temporal sampling rate than the bitstream, complete access units are removed from the bitstream when deriving the sub-bitstream. In this case, high-level syntax and inter prediction reference pictures in the bitstream are constructed accordingly. For spatial and quality bitstream scalability, i.e., the presence of a sub-bitstream with lower spatial resolution or quality than the bitstream, NAL units are removed from the bitstream when deriving the sub-bitstream. In this case, inter-layer prediction, i.e., the prediction of the higher spatial resolution or quality signal by data of the lower spatial resolution or quality signal, is typically used for efficient coding. Otherwise, the coding algorithm as described in the previous paragraph is used.

Multiview video coding is specified in Annex H allowing the construction of bitstreams that represent multiple views. Similar to scalable video coding, bitstreams that represent multiple views may also contain sub-bitstreams that conform to this Specification. For temporal bitstream scalability, i.e., the presence of a sub-bitstream with a smaller temporal sampling rate than the bitstream, complete access units are removed from the bitstream when deriving the sub-bitstream. In this case, high-level syntax and inter prediction reference pictures in the bitstream are constructed accordingly. For view bitstream scalability, i.e. the presence of a sub-bitstream with fewer views than the bitstream, NAL units are removed from the bitstream when deriving the sub-bitstream. In this case, inter-view prediction, i.e., the prediction of one view signal by data of another view signal, is typically used for efficient coding. Otherwise, the coding algorithm as described in the previous paragraph is used.

An extension of multiview video coding that additionally supports the inclusion of depth maps is specified in Annex I, allowing the construction of bitstreams that represent multiple views with corresponding depth views. In a similar manner as with the multiview video coding specified in Annex H, bitstreams encoded as specified in Annex I may also contain sub-bitstreams that conform to this Specification.

A multiview video coding extension with depth information is specified in Annex J. Sub-bitstreams consisting of a texture base view conform to this Specification, sub-bitstreams consisting of multiple texture views may also conform to Annex H of this Specification, and sub-bitstreams consisting of one or more texture views and one or more depth views may also conform to Annex I of this Specification. Enhanced texture view coding that utilizes the associated depth views and decoding processes for depth views are specified for this extension.

0.6.1 Predictive coding

This subclause does not form an integral part of this Recommendation | International Standard.

Because of the conflicting requirements of random access and highly efficient compression, two main coding types are specified. Intra coding is done without reference to other pictures. Intra coding may provide access points to the coded sequence where decoding can begin and continue correctly, but typically also shows only moderate compression efficiency. Inter coding (predictive or bi-predictive) is more efficient using inter prediction of each block of sample values from some previously decoded picture selected by the encoder. In contrast to some other video coding standards, pictures coded using bi-predictive inter prediction may also be used as references for inter coding of other pictures.

The application of the three coding types to pictures in a sequence is flexible, and the order of the decoding process is generally not the same as the order of the source picture capture process in the encoder or the output order from the decoder for display. The choice is left to the encoder and will depend on the requirements of the application. The decoding order is specified such that the decoding of pictures that use inter-picture prediction follows later in decoding order than other pictures that are referenced in the decoding process.

0.6.2 Coding of progressive and interlaced video

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard specifies a syntax and decoding process for video that originated in either progressive-scan or interlaced-scan form, which may be mixed together in the same sequence. The two fields of an interlaced frame are separated in capture time while the two fields of a progressive frame share the same capture time. Each field may be coded separately or the two fields may be coded together as a frame. Progressive frames are typically coded as a frame. For interlaced video, the encoder can choose between frame coding and field coding. Frame coding or field coding can be adaptively selected on a picture-by-picture basis and also on a more localized basis within a coded frame. Frame coding is typically preferred when the video scene contains significant detail with limited motion. Field coding typically works better when there is fast picture-to-picture motion.

0.6.3 Picture partitioning into macroblocks and smaller partitions

This subclause does not form an integral part of this Recommendation | International Standard.

As in previous video coding Recommendations and International Standards, a macroblock, consisting of a 16x16 block of luma samples and two corresponding blocks of chroma samples, is used as the basic processing unit of the video decoding process.

A macroblock can be further partitioned for inter prediction. The selection of the size of inter prediction partitions is a result of a trade-off between the coding gain provided by using motion compensation with smaller blocks and the quantity of data needed to represent the data for motion compensation. In this Recommendation | International Standard the inter prediction process can form segmentations for motion representation as small as 4x4 luma samples in size, using motion vector accuracy of one-quarter of the luma sample grid spacing displacement. The process for inter prediction of a sample block can also involve the selection of the picture to be used as the reference picture from a number of stored

previously-decoded pictures. Motion vectors are encoded differentially with respect to predicted values formed from nearby encoded motion vectors.

Typically, the encoder calculates appropriate motion vectors and other data elements represented in the video data stream. This motion estimation process in the encoder and the selection of whether to use inter prediction for the representation of each region of the video content is not specified in this Recommendation | International Standard.

0.6.4 Spatial redundancy reduction

This subclause does not form an integral part of this Recommendation | International Standard.

Both source pictures and prediction residuals have high spatial redundancy. This Recommendation | International Standard is based on the use of a block-based transform method for spatial redundancy removal. After inter prediction from previously-decoded samples in other pictures or spatial-based prediction from previously-decoded samples within the current picture, the resulting prediction residual is split into 4x4 blocks. These are converted into the transform domain where they are quantised. After quantisation many of the transform coefficients are zero or have low amplitude and can thus be represented with a small amount of encoded data. The processes of transformation and quantisation in the encoder are not specified in this Recommendation | International Standard.

0.7 How to read this Specification

This subclause does not form an integral part of this Recommendation | International Standard.

It is suggested that the reader starts with clause 1 (Scope) and moves on to clause 3 (Definitions). Clause 6 should be read for the geometrical relationship of the source, input, and output of the decoder. Clause 7 (Syntax and semantics) specifies the order to parse syntax elements from the bitstream. See subclauses 7.1-7.3 for syntactical order and see subclause 7.4 for semantics; i.e., the scope, restrictions, and conditions that are imposed on the syntax elements. The actual parsing for most syntax elements is specified in clause 9 (Parsing process). Finally, clause 8 (Decoding process) specifies how the syntax elements are mapped into decoded samples. Throughout reading this Specification, the reader should refer to clauses 2 (Normative references), 4 (Abbreviations), and 5 (Conventions) as needed. Annexes A through E, G, and H also form an integral part of this Recommendation | International Standard.

Annex A specifies thirteen profiles (Baseline, Constrained Baseline, Main, Extended, High, Progressive High, High 10, High 4:2:2, High 4:4:4 Predictive, High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, and CAVLC 4:4:4 Intra), each being tailored to certain application domains, and defines the so-called levels of the profiles. Annex B specifies syntax and semantics of a byte stream format for delivery of coded video as an ordered stream of bytes. Annex C specifies the hypothetical reference decoder and its use to check bitstream and decoder conformance. Annex D specifies syntax and semantics for supplemental enhancement information message payloads. Annex E specifies syntax and semantics of the video usability information parameters of the sequence parameter set.

Annex G specifies scalable video coding (SVC). The reader is referred to Annex G for the entire decoding process for SVC, which is specified there with references being made to clauses 2-9 and Annexes A-E. Subclause G.10 specifies three profiles for SVC (Scalable Baseline, Scalable High, and Scalable High Intra).

Annex H specifies multiview video coding (MVC) and multi-resolution frame compatible stereo coding (MFC). The reader is referred to Annex H for the entire decoding process for MVC and MFC, which is specified there with references being made to clauses 2-9 and Annexes A-E. Subclause H.10 specifies two profiles for MVC (Multiview High and Stereo High) and one profile for MFC (MFC High).

Annex I specifies MVC extensions for inclusion of depth maps, referred to as multiview video coding with depth (MVCD). The reader is referred to Annex I for the entire decoding process for MVCD, which is specified there with references being made to clauses 2-9 and Annexes A-E and Annex H. Subclause I.10 specifies one profile for MVCD (Multiview and Depth).

Annex J specifies a multiview video coding extension with depth information (3D-AVC). The reader is referred to Annex J for the entire decoding process for 3D-AVC, which is specified there with references being made to clauses 2-9 and Annexes A-E and Annexes H-I. Subclause J.10 specifies one profile for 3D-AVC.

Throughout this Specification, statements appearing with the preamble "NOTE -" are informative and are not an integral part of this Recommendation | International Standard.

Information technology — Coding of audio-visual objects —

Part 10: Advanced Video Coding

1 Scope

This Part of ISO/IEC 14496 specifies advanced video coding for coding of audio-visual objects.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 11578:1996, *Information technology — Open Systems Interconnection — Remote Procedure Call (RPC)*.
- ISO 11664-1, *Colorimetry — Part 1: CIE standard colorimetric observers*.
- ISO 12232:2006, *Photography – Digital still cameras – Determination of exposure index, ISO speed ratings, standard output sensitivity, and recommended exposure index*.

3 Definitions

For the purposes of this document, the following definitions apply:

- 3.1 access unit:** A set of NAL units that are consecutive in *decoding order* and contain exactly one *primary coded picture*. In addition to the *primary coded picture*, an access unit may also contain one or more *redundant coded pictures*, one *auxiliary coded picture*, or other NAL units not containing *slices* or *slice data partitions* of a *coded picture*. The decoding of an access unit always results in a *decoded picture*.
- 3.2 AC transform coefficient:** Any *transform coefficient* for which the *frequency index* in one or both dimensions is non-zero.
- 3.3 adaptive binary arithmetic decoding process:** An *entropy decoding process* that derives the values of *bins* from a *bitstream* produced by an *adaptive binary arithmetic encoding process*.
- 3.4 adaptive binary arithmetic encoding process:** An *entropy encoding process*, not normatively specified in this Recommendation | International Standard, that codes a sequence of *bins* and produces a *bitstream* that can be decoded using the *adaptive binary arithmetic decoding process*.
- 3.5 alpha blending:** A process not specified by this Recommendation | International Standard, in which an *auxiliary coded picture* is used in combination with a *primary coded picture* and with other data not specified by this Recommendation | International Standard in the *display process*. In an alpha blending process, the samples of an *auxiliary coded picture* are interpreted as indications of the degree of opacity (or, equivalently, the degrees of transparency) associated with the corresponding *luma* samples of the *primary coded picture*.
- 3.6 arbitrary slice order (ASO):** A *decoding order* of *slices* in which the *macroblock address* of the first *macroblock* of some *slice* of a *slice group* may be less than the *macroblock address* of the first *macroblock* of

some other preceding *slice* of the same *slice group* or, in the case of a *picture* that is coded using three separate colour planes, some other preceding *slice* of the same *slice group* within the same colour plane, or in which the *slices* of a *slice group* of a *picture* may be interleaved with the *slices* of one or more other *slice groups* of the *picture* or, in the case of a *picture* that is coded using three separate colour planes, with the *slices* of one or more other *slice groups* within the same colour plane.

- 3.7 **auxiliary coded picture:** A *picture* that supplements the *primary coded picture* that may be used in combination with other data not specified by this Recommendation | International Standard in the *display process*. An auxiliary coded picture has the same syntactic and semantic restrictions as a *monochrome redundant coded picture*. An auxiliary coded picture must contain the same number of *macroblocks* as the *primary coded picture*. Auxiliary coded pictures have no normative effect on the *decoding process*. See also *primary coded picture* and *redundant coded picture*.
- 3.8 **B slice:** A *slice* that may be decoded using *intra prediction* or *inter prediction* using at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.
- 3.9 **bin:** One bit of a *bin string*.
- 3.10 **binarization:** A set of *bin strings* for all possible values of a *syntax element*.
- 3.11 **binarization process:** A unique mapping process of all possible values of a *syntax element* onto a set of *bin strings*.
- 3.12 **bin string:** A string of *bins*. A bin string is an intermediate binary representation of values of *syntax elements* from the *binarization* of the *syntax element*.
- 3.13 **bi-predictive slice:** See *B slice*.
- 3.14 **bitstream:** A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.
- 3.15 **block:** An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.
- 3.16 **bottom field:** One of two *fields* that comprise a *frame*. Each row of a *bottom field* is spatially located immediately below a corresponding row of a *top field*.
- 3.17 **bottom macroblock (of a macroblock pair):** The *macroblock* within a *macroblock pair* that contains the samples in the bottom row of samples for the *macroblock pair*. For a *field macroblock pair*, the bottom macroblock represents the samples from the region of the *bottom field* of the *frame* that lie within the spatial region of the *macroblock pair*. For a *frame macroblock pair*, the bottom macroblock represents the samples of the *frame* that lie within the bottom half of the spatial region of the *macroblock pair*.
- 3.18 **broken link:** A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.
- 3.19 **byte:** A sequence of 8 bits, written and read with the most significant bit on the left and the least significant bit on the right. When represented in a sequence of data bits, the most significant bit of a byte is first.
- 3.20 **byte-aligned:** A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*. A bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.
- 3.21 **byte stream:** An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.
- 3.22 **can:** A term used to refer to behaviour that is allowed, but not necessarily required.
- 3.23 **category:** A number associated with each *syntax element*. The category is used to specify the allocation of *syntax elements* to *NAL units* for *slice data partitioning*. It may also be used in a manner determined by the application to refer to classes of *syntax elements* in a manner not specified in this Recommendation | International Standard.
- 3.24 **chroma:** An adjective specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours. The symbols used for a chroma array or sample are Cb and Cr.

NOTE – The term *chroma* is used rather than the term *chrominance* in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term *chrominance*.

- 3.25 **coded field:** A *coded representation* of a *field*.
- 3.26 **coded frame:** A *coded representation* of a *frame*.
- 3.27 **coded picture:** A *coded representation* of a *picture*. A coded picture may be either a *coded field* or a *coded frame*. Coded picture is a collective term referring to a *primary coded picture* or a *redundant coded picture*, but not to both together.
- 3.28 **coded picture buffer (CPB):** A first-in first-out buffer containing *access units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C.
- 3.29 **coded representation:** A data element as represented in its coded form.
- 3.30 **coded slice data partition NAL unit:** A *NAL unit* containing a *slice data partition*.
- 3.31 **coded slice NAL unit:** A *NAL unit* containing a *slice* that is not a *slice* of an *auxiliary coded picture*.
- 3.32 **coded video sequence:** A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *access units* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.
- 3.33 **component:** An array or single sample from one of the three arrays (*luma* and two *chroma*) that make up a *field* or *frame* in 4:2:0, 4:2:2, or 4:4:4 colour format or the array or a single sample of the array that make up a *field* or *frame* in monochrome format.
- 3.34 **complementary field pair:** A collective term for a *complementary reference field pair* or a *complementary non-reference field pair*.
- 3.35 **complementary non-reference field pair:** Two *non-reference fields* that are in consecutive *access units* in *decoding order* as two *coded fields* of opposite parity and share the same value of the *frame_num syntax element*, where the first *field* is not already a *paired field*.
- 3.36 **complementary reference field pair:** Two *reference fields* that are in consecutive *access units* in *decoding order* as two *coded fields* and share the same value of the *frame_num syntax element*, where the second *field* in *decoding order* is not an *IDR picture* and does not include a *memory_management_control_operation syntax element* equal to 5.
- 3.37 **context variable:** A variable specified for the *adaptive binary arithmetic decoding process* of a *bin* by an equation containing recently decoded *bins*.
- 3.38 **DC transform coefficient:** A *transform coefficient* for which the *frequency index* is zero in all dimensions.
- 3.39 **decoded picture:** A *decoded picture* is derived by decoding a *coded picture*. A *decoded picture* is either a *decoded frame*, or a *decoded field*. A *decoded field* is either a *decoded top field* or a *decoded bottom field*.
- 3.40 **decoded picture buffer (DPB):** A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.
- 3.41 **decoder:** An embodiment of a *decoding process*.
- 3.42 **decoder under test (DUT):** A *decoder* that is tested for conformance to this Recommendation | International Standard by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing of the output of the two *decoders*.
- 3.43 **decoding order:** The order in which *syntax elements* are processed by the *decoding process*.
- 3.44 **decoding process:** The process specified in this Recommendation | International Standard that reads a *bitstream* and derives *decoded pictures* from it.
- 3.45 **direct prediction:** An *inter prediction* for a *block* for which no *motion vector* is decoded. Two direct *prediction* modes are specified that are referred to as *spatial direct prediction* and *temporal prediction* mode.
- 3.46 **display process:** A process not specified in this Recommendation | International Standard having, as its input, the cropped *decoded pictures* that are the output of the *decoding process*.

- 3.47 emulation prevention byte:** A *byte* equal to 0x03 that may be present within a *NAL unit*. The presence of emulation prevention bytes ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* contains a *start code prefix*.
- 3.48 encoder:** An embodiment of an *encoding process*.
- 3.49 encoding process:** A process, not specified in this Recommendation | International Standard, that produces a *bitstream* conforming to this Recommendation | International Standard.
- 3.50 field:** An assembly of alternate rows of a *frame*. A *frame* is composed of two *fields*, a *top field* and a *bottom field*.
- 3.51 field macroblock:** A *macroblock* containing samples from a single *field*. All *macroblocks* of a *coded field* are field macroblocks. When *macroblock-adaptive frame/field decoding* is in use, some *macroblocks* of a *coded frame* may be field macroblocks.
- 3.52 field macroblock pair:** A *macroblock pair* decoded as two *field macroblocks*.
- 3.53 field scan:** A specific sequential ordering of *transform coefficients* that differs from the *zig-zag scan* by scanning columns more rapidly than rows. Field scan is used for *transform coefficients* in *field macroblocks*.
- 3.54 flag:** A variable that can take one of the two possible values 0 and 1.
- 3.55 frame:** A *frame* contains an array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format. A *frame* consists of two *fields*, a *top field* and a *bottom field*.
- 3.56 frame macroblock:** A *macroblock* representing samples from the two *fields* of a *coded frame*. When *macroblock-adaptive frame/field decoding* is not in use, all *macroblocks* of a *coded frame* are frame macroblocks. When *macroblock-adaptive frame/field decoding* is in use, some *macroblocks* of a *coded frame* may be frame macroblocks.
- 3.57 frame macroblock pair:** A *macroblock pair* decoded as two *frame macroblocks*.
- 3.58 frequency index:** A one-dimensional or two-dimensional index associated with a *transform coefficient* prior to an *inverse transform* part of the *decoding process*.
- 3.59 hypothetical reference decoder (HRD):** A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.
- 3.60 hypothetical stream scheduler (HSS):** A hypothetical delivery mechanism for the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*. The HSS is used for checking the conformance of a *bitstream* or a *decoder*.
- 3.61 I slice:** A *slice* that is not an *SI slice* that is decoded using *intra prediction* only.
- 3.62 informative:** A term used to refer to content provided in this Recommendation | International Standard that is not an integral part of this Recommendation | International Standard. Informative content does not establish any mandatory requirements for conformance to this Recommendation | International Standard.
- 3.63 instantaneous decoding refresh (IDR) access unit:** An *access unit* in which the *primary coded picture* is an *IDR picture*.
- 3.64 instantaneous decoding refresh (IDR) picture:** A *coded picture* for which the variable *IdrPicFlag* is equal to 1. An *IDR picture* causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after the decoding of the *IDR picture*. All *coded pictures* that follow an *IDR picture* in *decoding order* can be decoded without *inter prediction* from any *picture* that precedes the *IDR picture* in *decoding order*. The first *picture* of each *coded video sequence* in *decoding order* is an *IDR picture*.
- 3.65 inter coding:** Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *inter prediction*.
- 3.66 inter prediction:** A *prediction* derived from decoded samples of *reference pictures* other than the current decoded *picture*.
- 3.67 interpretation sample value:** A possibly-altered value corresponding to a decoded sample value of an *auxiliary coded picture* that may be generated for use in the *display process*. Interpretation sample values are not used in the *decoding process* and have no normative effect on the *decoding process*.

- 3.68 intra coding:** Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *intra prediction*.
- 3.69 intra prediction:** A *prediction* derived from the decoded samples of the same decoded *slice*.
- 3.70 intra slice:** See *I slice*.
- 3.71 inverse transform:** A part of the *decoding process* by which a set of *transform coefficients* are converted into spatial-domain values, or by which a set of *transform coefficients* are converted into *DC transform coefficients*.
- 3.72 layer:** One of a set of syntactical structures in a non-branching hierarchical relationship. Higher layers contain lower layers. The coding layers are the *coded video sequence*, *picture*, *slice*, and *macroblock* layers.
- 3.73 level:** A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this Recommendation | International Standard. The same set of levels is defined for all *profiles*, with most aspects of the definition of each level being in common across different *profiles*. Individual implementations may, within specified constraints, support a different level for each supported *profile*. In a different context, a level is the value of a *transform coefficient* prior to *scaling* (see the definition of *transform coefficient level*).
- 3.74 list:** A one-dimensional array of *syntax elements* or variables.
- 3.75 list 0 (list 1) motion vector:** A *motion vector* associated with a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.76 list 0 (list 1) prediction:** *Inter prediction* of the content of a *slice* using a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.77 luma:** An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colours. The symbol or subscript used for luma is Y or L.
NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.
- 3.78 macroblock:** A 16x16 *block* of *luma* samples and two corresponding *blocks* of *chroma* samples of a *picture* that has three sample arrays, or a 16x16 *block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes. The division of a *slice* or a *macroblock pair* into macroblocks is a *partitioning*.
- 3.79 macroblock-adaptive frame/field decoding:** A *decoding process* for *coded frames* in which some *macroblocks* may be decoded as *frame macroblocks* and others may be decoded as *field macroblocks*.
- 3.80 macroblock address:** When *macroblock-adaptive frame/field decoding* is not in use, a macroblock address is the index of a *macroblock* in a *macroblock raster scan* of the *picture* starting with zero for the top-left *macroblock* in a *picture*. When *macroblock-adaptive frame/field decoding* is in use, the macroblock address of the *top macroblock* of a *macroblock pair* is two times the index of the *macroblock pair* in a *macroblock pair raster scan* of the *picture*, and the macroblock address of the *bottom macroblock* of a *macroblock pair* is the macroblock address of the corresponding *top macroblock* plus 1. The macroblock address of the *top macroblock* of each *macroblock pair* is an even number and the macroblock address of the *bottom macroblock* of each *macroblock pair* is an odd number.
- 3.81 macroblock location:** The two-dimensional coordinates of a *macroblock* in a *picture* denoted by (x, y). For the top left *macroblock* of the *picture* (x, y) is equal to (0, 0). x is incremented by 1 for each *macroblock* column from left to right. When *macroblock-adaptive frame/field decoding* is not in use, y is incremented by 1 for each *macroblock* row from top to bottom. When *macroblock-adaptive frame/field decoding* is in use, y is incremented by 2 for each *macroblock pair* row from top to bottom, and is incremented by an additional 1 when a *macroblock* is a *bottom macroblock*.
- 3.82 macroblock pair:** A pair of vertically contiguous *macroblocks* in a *frame* that is coupled for use in *macroblock-adaptive frame/field decoding*. The division of a *slice* into macroblock pairs is a *partitioning*.
- 3.83 macroblock partition:** A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *macroblock* for *inter prediction* for a *picture* that has three sample arrays or a *block* of *luma* samples resulting from a *partitioning* of a *macroblock* for *inter prediction* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.

- 3.84 macroblock to slice group map:** A means of mapping *macroblocks* of a *picture* into *slice groups*. The macroblock to slice group map consists of a list of numbers, one for each coded *macroblock*, specifying the *slice group* to which each coded *macroblock* belongs.
- 3.85 map unit to slice group map:** A means of mapping *slice group map units* of a *picture* into *slice groups*. The map unit to slice group map consists of a list of numbers, one for each *slice group map unit*, specifying the *slice group* to which each coded *slice group map unit* belongs.
- 3.86 matrix:** A two-dimensional array of *syntax elements* or variables.
- 3.87 may:** A term used to refer to behaviour that is allowed, but not necessarily required. In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.
- 3.88 memory management control operation:** Seven operations that control *reference picture marking*.
- 3.89 motion vector:** A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.
- 3.90 must:** A term used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this Recommendation | International Standard. This term is used exclusively in an *informative* context.
- 3.91 NAL unit:** A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.
- 3.92 NAL unit stream:** A sequence of *NAL units*.
- 3.93 non-paired field:** A collective term for a *non-paired reference field* or a *non-paired non-reference field*.
- 3.94 non-paired non-reference field:** A decoded *non-reference field* that is not part of a *complementary non-reference field pair*.
- 3.95 non-paired reference field:** A decoded *reference field* that is not part of a *complementary reference field pair*.
- 3.96 non-reference field:** A *field* coded with *nal_ref_idc* equal to 0.
- 3.97 non-reference frame:** A *frame* coded with *nal_ref_idc* equal to 0.
- 3.98 non-reference picture:** A *picture* coded with *nal_ref_idc* equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.
- 3.99 note:** A term used to prefix *informative* remarks. This term is used exclusively in an *informative* context.
- 3.100 opposite parity:** The *opposite parity* of *top* is *bottom*, and vice versa.
- 3.101 output order:** The order in which the *decoded pictures* are output from the *decoded picture buffer*.
- 3.102 P slice:** A *slice* that is not an *SP slice* that may be decoded using *intra prediction* or *inter prediction* using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.
- 3.103 parameter:** A *syntax element* of a *sequence parameter set* or a *picture parameter set*. Parameter is also used as part of the defined term *quantisation parameter*.
- 3.104 parity:** The parity of a *field* can be *top* or *bottom*.
- 3.105 partitioning:** The division of a set into subsets such that each element of the set is in exactly one of the subsets.
- 3.106 picture:** A collective term for a *field* or a *frame*.
- 3.107 picture parameter set:** A *syntax structure* containing *syntax elements* that apply to zero or more entire coded *pictures* as determined by the *pic_parameter_set_id syntax element* found in each *slice header*.
- 3.108 picture order count:** A variable that is associated with each coded *field* and each *field* of a coded *frame* and has a value that is non-decreasing with increasing *field* position in *output order* relative to the first output *field* of the previous *IDR picture* in *decoding order* or relative to the first output *field* of the previous *picture*, in *decoding order*, that contains a *memory management control operation* that marks all *reference pictures* as "unused for reference".
- 3.109 prediction:** An embodiment of the *prediction process*.

- 3.110 prediction process:** The use of a *predictor* to provide an estimate of the sample value or data element currently being decoded.
- 3.111 predictive slice:** See *P slice*.
- 3.112 predictor:** A combination of specified values or previously decoded sample values or data elements used in the *decoding process* of subsequent sample values or data elements.
- 3.113 primary coded picture:** The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this Recommendation | International Standard. The primary coded picture contains all *macroblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures. See also *redundant coded picture*.
- 3.114 profile:** A specified subset of the syntax of this Recommendation | International Standard.
- 3.115 quantisation parameter:** A variable used by the *decoding process* for *scaling of transform coefficient levels*.
- 3.116 random access:** The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.
- 3.117 raster scan:** A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.
- 3.118 raw byte sequence payload (RBSP):** A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit*. An RBSP is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and followed by zero or more subsequent bits equal to 0.
- 3.119 raw byte sequence payload (RBSP) stop bit:** A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*. The location of the end of the *string of data bits* within an RBSP can be identified by searching from the end of the RBSP for the *RBSP stop bit*, which is the last non-zero bit in the RBSP.
- 3.120 recovery point:** A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.
- 3.121 redundant coded picture:** A coded representation of a *picture* or a part of a *picture*. The content of a redundant coded picture shall not be used by the *decoding process* for a *bitstream* conforming to this Recommendation | International Standard. A *redundant coded picture* is not required to contain all *macroblocks* in the *primary coded picture*. Redundant coded pictures have no normative effect on the *decoding process*. See also *primary coded picture*.
- 3.122 reference field:** A *reference field* may be used for *inter prediction* when *P*, *SP*, and *B slices* of a *coded field* or *field macroblocks* of a *coded frame* are decoded. See also *reference picture*.
- 3.123 reference frame:** A *reference frame* may be used for *inter prediction* when *P*, *SP*, and *B slices* of a *coded frame* are decoded. See also *reference picture*.
- 3.124 reference index:** An index into a *reference picture list*.
- 3.125 reference picture:** A *picture* with *nal_ref_idc* not equal to 0. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.
- 3.126 reference picture list:** A list of *reference pictures* that is used for *inter prediction* of a *P*, *B*, or *SP slice*. For the *decoding process* of a *P* or *SP slice*, there is one reference picture list. For the *decoding process* of a *B slice*, there are two reference picture lists.
- 3.127 reference picture list 0:** A *reference picture list* used for *inter prediction* of a *P*, *B*, or *SP slice*. All *inter prediction* used for *P* and *SP slices* uses reference picture list 0. Reference picture list 0 is one of two *reference picture lists* used for *inter prediction* for a *B slice*, with the other being *reference picture list 1*.
- 3.128 reference picture list 1:** A *reference picture list* used for *inter prediction* of a *B slice*. Reference picture list 1 is one of two *reference picture lists* used for *inter prediction* for a *B slice*, with the other being *reference picture list 0*.
- 3.129 reference picture marking:** Specifies, in the *bitstream*, how the *decoded pictures* are marked for *inter prediction*.

- 3.130 reserved:** The term reserved, when used in the clauses specifying some values of a particular *syntax element*, are for future use by ITU-T | ISO/IEC. These values shall not be used in *bitstreams* conforming to this Recommendation | International Standard, but may be used in future extensions of this Recommendation | International Standard by ITU-T | ISO/IEC.
- 3.131 residual:** The decoded difference between a *prediction* of a sample or data element and its decoded value.
- 3.132 run:** A number of consecutive data elements represented in the decoding process. In one context, the number of zero-valued *transform coefficient levels* preceding a non-zero *transform coefficient level* in the list of *transform coefficient levels* generated by a *zig-zag scan* or a *field scan*. In other contexts, run refers to a number of *macroblocks*.
- 3.133 sample aspect ratio:** Specifies, for assisting the *display process*, which is not specified in this Recommendation | International Standard, the ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *frame*. Sample aspect ratio is expressed as *h:v*, where *h* is horizontal width and *v* is vertical height (in arbitrary units of spatial distance).
- 3.134 scaling:** The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.
- 3.135 sequence parameter set:** A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded video sequences* as determined by the content of a *seq_parameter_set_id syntax element* found in the *picture parameter set* referred to by the *pic_parameter_set_id syntax element* found in each *slice header*.
- 3.136 shall:** A term used to express mandatory requirements for conformance to this Recommendation | International Standard. When used to express a mandatory constraint on the values of *syntax elements* or on the results obtained by operation of the specified *decoding process*, it is the responsibility of the *encoder* to ensure that the constraint is fulfilled. When used in reference to operations performed by the *decoding process*, any *decoding process* that produces identical results to the *decoding process* described herein conforms to the *decoding process* requirements of this Recommendation | International Standard.
- 3.137 should:** A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Recommendation | International Standard.
- 3.138 SI slice:** A *slice* that is coded using *intra prediction* only and using quantisation of the *prediction* samples. An SI slice can be coded such that its decoded samples can be constructed identically to an *SP slice*.
- 3.139 skipped macroblock:** A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.
- 3.140 slice:** An integer number of *macroblocks* or *macroblock pairs* ordered consecutively in the *raster scan* within a particular *slice group*. For the *primary coded picture*, the division of each *slice group* into slices is a *partitioning*. Although a slice contains *macroblocks* or *macroblock pairs* that are consecutive in the *raster scan* within a *slice group*, these *macroblocks* or *macroblock pairs* are not necessarily consecutive in the *raster scan* within the *picture*. The *macroblock addresses* are derived from the first *macroblock address* in a slice (as represented in the *slice header*) and the *macroblock to slice group map*, and, when a *picture* is coded using three separate colour planes, a colour plane identifier.
- 3.141 slice data partition:** A non-empty subset of the *syntax elements* of the slice data *syntax structure* for a *slice*. The *syntax elements* of a slice data partition are associated with the same *category*.
- 3.142 slice data partitioning:** A method of *partitioning* selected *syntax elements* into *syntax structures* based on a *category* associated with each *syntax element*.
- 3.143 slice group:** A subset of the *macroblocks* or *macroblock pairs* of a *picture*. The division of the *picture* into slice groups is a *partitioning* of the *picture*. The *partitioning* is specified by the *macroblock to slice group map*.
- 3.144 slice group map units:** The units of the *map unit to slice group map*.
- 3.145 slice header:** A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.
- 3.146 source:** Term used to describe the video material or some of its attributes before encoding.
- 3.147 SP slice:** A *slice* that may be coded using *intra prediction* or *inter prediction* with quantisation of the *prediction* samples using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*. An SP

slice can be coded such that its decoded samples can be constructed identically to another SP slice or an SI slice.

- 3.148 start code prefix:** A unique sequence of three bytes equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*. The location of a start code prefix can be used by a *decoder* to identify the beginning of a new *NAL unit* and the end of a previous *NAL unit*. Emulation of start code prefixes is prevented within *NAL units* by the inclusion of *emulation prevention bytes*.
- 3.149 string of data bits (SODB):** A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*. Within an SODB, the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.
- 3.150 sub-macroblock:** One quarter of the samples of a *macroblock*, i.e., an 8x8 *luma block* and two corresponding *chroma blocks* of which one corner is located at a corner of the *macroblock* for a *picture* that has three sample arrays or an 8x8 *luma block* of which one corner is located at a corner of the *macroblock* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.
- 3.151 sub-macroblock partition:** A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *sub-macroblock* for *inter prediction* for a *picture* that has three sample arrays or a *block* of *luma* samples resulting from a *partitioning* of a *sub-macroblock* for *inter prediction* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.
- 3.152 switching I slice:** See *SI slice*.
- 3.153 switching P slice:** See *SP slice*.
- 3.154 syntax element:** An element of data represented in the *bitstream*.
- 3.155 syntax structure:** Zero or more *syntax elements* present together in the *bitstream* in a specified order.
- 3.156 top field:** One of two *fields* that comprise a *frame*. Each row of a *top field* is spatially located immediately above the corresponding row of the *bottom field*.
- 3.157 top macroblock (of a macroblock pair):** The *macroblock* within a *macroblock pair* that contains the samples in the top row of samples for the *macroblock pair*. For a *field macroblock pair*, the top macroblock represents the samples from the region of the *top field* of the *frame* that lie within the spatial region of the *macroblock pair*. For a *frame macroblock pair*, the top macroblock represents the samples of the *frame* that lie within the top half of the spatial region of the *macroblock pair*.
- 3.158 transform coefficient:** A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in an *inverse transform* part of the *decoding process*.
- 3.159 transform coefficient level:** An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.
- 3.160 universal unique identifier (UUID):** An identifier that is unique with respect to the space of all universal unique identifiers.
- 3.161 unspecified:** The term unspecified, when used in the clauses specifying some values of a particular *syntax element*, indicates that the values have no specified meaning in this Recommendation | International Standard and will not have a specified meaning in the future as an integral part of this Recommendation | International Standard.
- 3.162 variable length coding (VLC):** A reversible procedure for entropy coding that assigns shorter bit strings to *symbols* expected to be more frequent and longer bit strings to *symbols* expected to be less frequent.
- 3.163 VCL NAL unit:** A collective term for *coded slice NAL units* and *coded slice data partition NAL units*.
- 3.164 zig-zag scan:** A specific sequential ordering of *transform coefficient levels* from (approximately) the lowest spatial frequency to the highest. Zig-zag scan is used for *transform coefficient levels* in *frame macroblocks*.

4 Abbreviations

For the purposes of this International Standard, the following abbreviations apply:

CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CBR	Constant Bit Rate
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
DUT	Decoder under test
FIFO	First-In, First-Out
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
IDR	Instantaneous Decoding Refresh
LSB	Least Significant Bit
MB	Macroblock
MBAFF	Macroblock-Adaptive Frame-Field Coding
MSB	Most Significant Bit
MVC	Multiview Video Coding
MVCD	Multiview Video Coding with Depth
NAL	Network Abstraction Layer
RBSP	Raw Byte Sequence Payload
SEI	Supplemental Enhancement Information
SODB	String Of Data Bits
SVC	Scalable Video Coding
UUID	Universal Unique Identifier
VBR	Variable Bit Rate
VCL	Video Coding Layer
VLC	Variable Length Coding
VUI	Video Usability Information

5 Conventions

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, integer division and arithmetic shift operations are specifically defined. Numbering and counting conventions generally begin from 0.

5.1 Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
–	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication

x^y	Exponentiation. Specifies x to the power of y . In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
$/$	Integer division with truncation of the result toward zero. For example, $7/4$ and $-7/-4$ are truncated to 1 and $-7/4$ and $7/-4$ are truncated to -1 .
$\div$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^y f(i)$	The summation of $f(i)$ with i taking all integer values from x up to and including y .
$x \% y$	Modulus. Remainder of x divided by y , defined only for integers x and y with $x \geq 0$ and $y > 0$.

5.2 Logical operators

The following logical operators are defined as follows:

$x \ \&\& \ y$	Boolean logical "and" of x and y .
$x \ \ y$	Boolean logical "or" of x and y .
$!$	Boolean logical "not".
$x \ ? \ y : z$	If x is TRUE or not equal to 0, evaluates to the value of y ; otherwise, evaluates to the value of z .

5.3 Relational operators

The following relational operators are defined as follows:

$>$	Greater than.
$>=$	Greater than or equal to.
$<$	Less than.
$<=$	Less than or equal to.
$=$	Equal to.
$!=$	Not equal to.

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.4 Bit-wise operators

The following bit-wise operators are defined as follows:

$\&$	Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
$ $	Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
$\wedge$	Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
$x \gg y$	Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y . Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

$x \ll y$ Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y . Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

5.5 Assignment operators

The following arithmetic operators are defined as follows:

- $=$ Assignment operator.
- $++$ Increment, i.e., $x++$ is equivalent to $x = x + 1$; when used in an array index, evaluates to the value of the variable prior to the increment operation.
- $--$ Decrement, i.e., $x--$ is equivalent to $x = x - 1$; when used in an array index, evaluates to the value of the variable prior to the decrement operation.
- $+=$ Increment by amount specified, i.e., $x += 3$ is equivalent to $x = x + 3$, and $x += (-3)$ is equivalent to $x = x + (-3)$.
- $-=$ Decrement by amount specified, i.e., $x -= 3$ is equivalent to $x = x - 3$, and $x -= (-3)$ is equivalent to $x = x - (-3)$.

5.6 Range notation

The following notation is used to specify a range of values:

$x = y..z$ x takes on integer values starting from y to z , inclusive, with x , y , and z being integer numbers.

5.7 Mathematical functions

The following mathematical functions are defined as follows:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases} \quad (5-1)$$

$$\text{Ceil}(x) \quad \text{the smallest integer greater than or equal to } x. \quad (5-2)$$

$$\text{Clip1}_V(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_V) - 1, x) \quad (5-3)$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_C) - 1, x) \quad (5-4)$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \quad (5-5)$$

$$\text{Floor}(x) \quad \text{the greatest integer less than or equal to } x. \quad (5-6)$$

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a \% (d/b)) * b & ; \quad e == 0 \\ (a / (d/b)) * c & ; \quad e == 1 \end{cases} \quad (5-7)$$

$$\text{Log2}(x) \quad \text{returns the base-2 logarithm of } x. \quad (5-8)$$

$\text{Log10}(x)$ returns the base-10 logarithm of x . (5-9)

$\text{Median}(x, y, z) = x + y + z - \text{Min}(x, \text{Min}(y, z)) - \text{Max}(x, \text{Max}(y, z))$ (5-10)

$\text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases}$ (5-11)

$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases}$ (5-12)

$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5)$ (5-13)

$\text{Sign}(x) = \begin{cases} 1 & ; \quad x \geq 0 \\ -1 & ; \quad x < 0 \end{cases}$ (5-14)

$\text{Sqrt}(x) = \sqrt{x}$ (5-15)

5.8 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- operations of a higher precedence are evaluated before any operation of a lower precedence,
- operations of the same precedence are evaluated sequentially from left to right.

Table 5-1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x--"
"!x", "-x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", " $\frac{x}{y}$ ", "x % y"
"x + y", "x - y" (as a two-argument operator), " $\sum_{i=x}^y f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = y", "x != y"
"x & y"
"x y"
"x && y"
"x y"
"x ? y : z"
"x = y", "x += y", "x -= y"

5.9 Variables, syntax elements, and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the bitstream are referred to as syntax functions. These functions are specified in subclause 7.2 and assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream. Syntax functions are described by their names, which are constructed as syntax element names and end with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in subclause 5.7) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any

underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing. Thus, an element of a matrix *s* at horizontal position *x* and vertical position *y* may be denoted either as *s*[*x*, *y*] or as *s*_{*yx*}.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.10 Text description of logical operations

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
    statement 0
else if ( condition 1 )
    statement 1
...
else /* informative remark on remaining condition */
    statement n
```

may be described in the following manner:

... as follows / ... the following applies:

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- ...
- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0a && condition 0b )
    statement 0
else if ( condition 1a || condition 1b )
    statement 1
...
else
    statement n
```

may be described in the following manner:

... as follows / ... the following applies:

- If all of the following conditions are true, statement 0
- condition 0a

- condition 0b
- Otherwise, if any of the following conditions are true, statement 1
 - condition 1a
 - condition 1b
- ...
- Otherwise, statement n

In the text, a statement of logical operations as would be described in pseudo-code as:

```
if( condition 0 )
    statement 0
if ( condition 1 )
    statement 1
```

may be described in the following manner:

When condition 0, statement 0
When condition 1, statement 1

5.11 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as the input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

In the specification of a process, a specific macroblock may be referred to by the variable name having a value equal to the address of the specific macroblock.

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

6.1 Bitstream formats

This subclause specifies the relationship between the NAL unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this Recommendation | International Standard. The byte stream format is specified in Annex B.

6.2 Source, decoded, and output picture formats

This subclause specifies the relationship between source and decoded frames and fields that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of either or both frames or fields (called collectively pictures) in decoding order.

The source and decoded pictures (frames or fields) are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome), with or without an auxiliary array.
- Luma and two Chroma (YCbCr or YCgCo), with or without an auxiliary array.
- Green, Blue and Red (GBR, also known as RGB), with or without an auxiliary array.
- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ), with or without an auxiliary array.

For convenience of notation and terminology in this Specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E. The (monochrome) auxiliary arrays, which may or may not be present as auxiliary pictures in a coded video sequence, are optional for decoding and can be used for such purposes as alpha blending.

The variables SubWidthC, and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. An entry marked as "-" in Table 6-1 denotes an undefined value for SubWidthC or SubHeightC. Other values of chroma_format_idc, SubWidthC, and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 – SubWidthC, and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma Format	SubWidthC	SubHeightC
0	0	monochrome	-	-
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	-	-

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, depending on the value of separate_colour_plane_flag, the following applies:

- If separate_colour_plane_flag is equal to 0, each of the two chroma arrays has the same height and width as the luma array.
- Otherwise (separate_colour_plane_flag is equal to 1), the three colour planes are separately processed as monochrome sampled pictures.

The width and height of the luma sample arrays are each an integer multiple of 16. In coded video sequences using 4:2:0 chroma sampling, the width and height of chroma sample arrays are each an integer multiple of 8. In coded video sequences using 4:2:2 sampling, the width of the chroma sample arrays is an integer multiple of 8 and the height is an integer multiple of 16. The height of a luma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 32. In coded video sequences using 4:2:0 chroma sampling, the height of each chroma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 16. The width or height of pictures output from the decoding process need not be an integer multiple of 16 and can be specified using a cropping rectangle.

The syntax for the luma and (when present) chroma arrays are ordered such when data for all three colour components is present, the data for the luma array is first, followed by any data for the Cb array, followed by any data for the Cr array, unless otherwise specified.

The width of fields coded referring to a specific sequence parameter set is the same as that of frames coded referring to the same sequence parameter set (see below). The height of fields coded referring to a specific sequence parameter set is half that of frames coded referring to the same sequence parameter set (see below).

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 14, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of `chroma_format_idc` is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

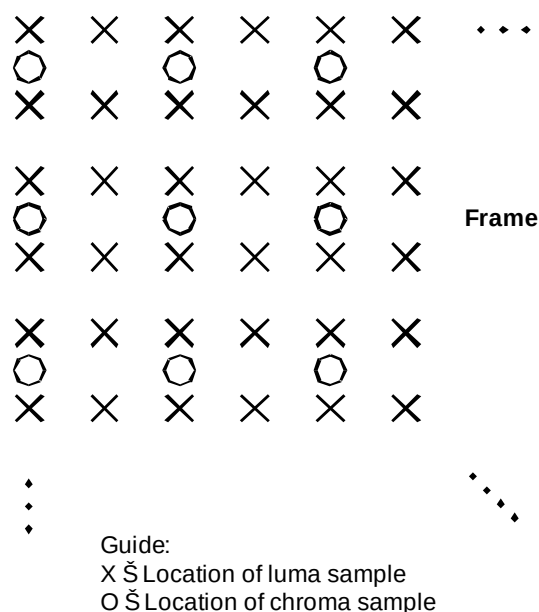


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

A frame consists of two fields as described below. A coded picture may represent a coded frame or an individual coded field. A coded video sequence conforming to this Recommendation | International Standard may contain arbitrary combinations of coded frames and coded fields. The decoding process is also specified in a manner that allows smaller regions of a coded frame to be coded either as a frame or field region, by use of macroblock-adaptive frame-field coding.

Source and decoded fields are one of two types: top field or bottom field. When two fields are output at the same time, or are combined to be used as a reference frame (see below), the two fields (which shall be of opposite parity) are interleaved. The first (i.e., top), third, fifth, etc., rows of a decoded frame are the top field rows. The second, fourth, sixth, etc., rows of a decoded frame are the bottom field rows. A top field consists of only the top field rows of a decoded frame. When the top field or bottom field of a decoded frame is used as a reference field (see below) only the even rows (for a top field) or the odd rows (for a bottom field) of the decoded frame are used.

When the value of `chroma_format_idc` is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in top and bottom fields are shown in Figure 6-2. The nominal vertical sampling relative locations of the chroma samples in a top field are specified as shifted up by one-quarter luma sample height relative to the field-sampling grid. The vertical sampling locations of the chroma samples in a bottom field are specified as shifted down by one-quarter luma sample height relative to the field-sampling grid. Alternative chroma sample relative locations may be indicated in the video usability information (see Annex E).

NOTE – The shifting of the chroma samples is in order for these samples to align vertically to the usual location relative to the full-frame sampling grid as shown in Figure 6-1.

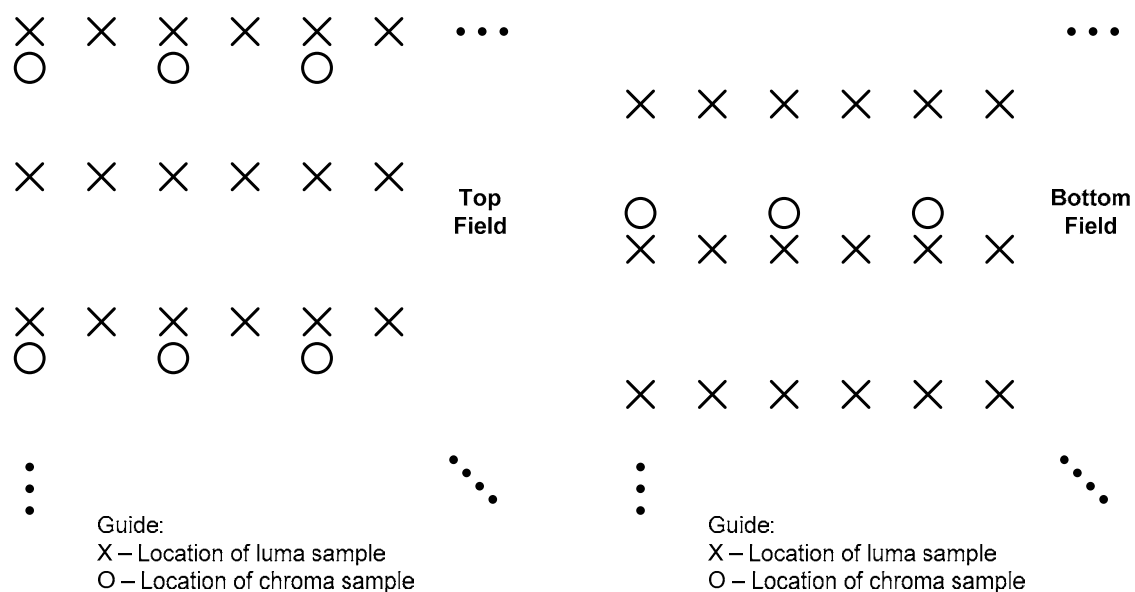


Figure 6-2 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields

When the value of `chroma_format_idc` is equal to 2, the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a frame and in fields are as shown in Figures 6-3 and 6-4, respectively.

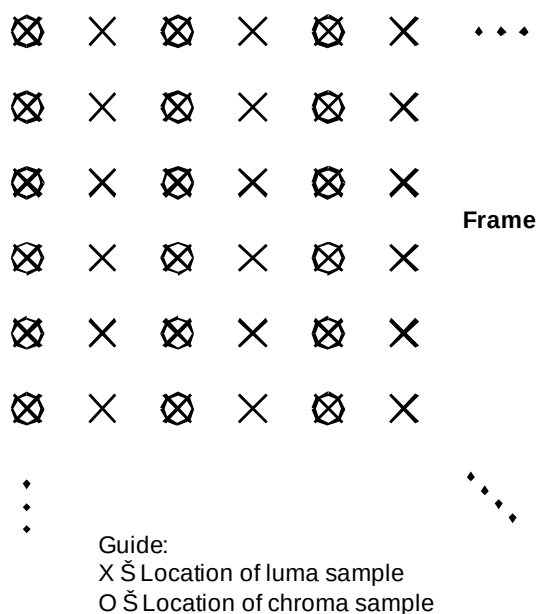


Figure 6-3 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame

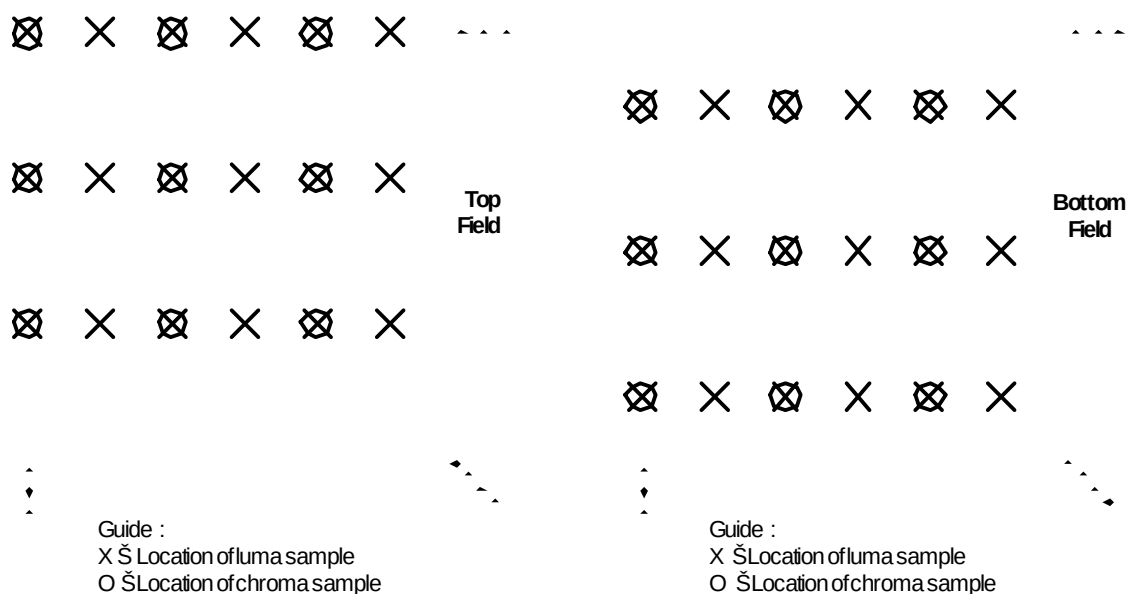


Figure 6-4 – Nominal vertical and horizontal sampling locations of 4:2:2 samples top and bottom fields

When the value of `chroma_format_idc` is equal to 3, all array samples are co-sited for all cases of frames and fields and the nominal locations in a frame and in fields are as shown in Figures 6-5 and 6-6, respectively.

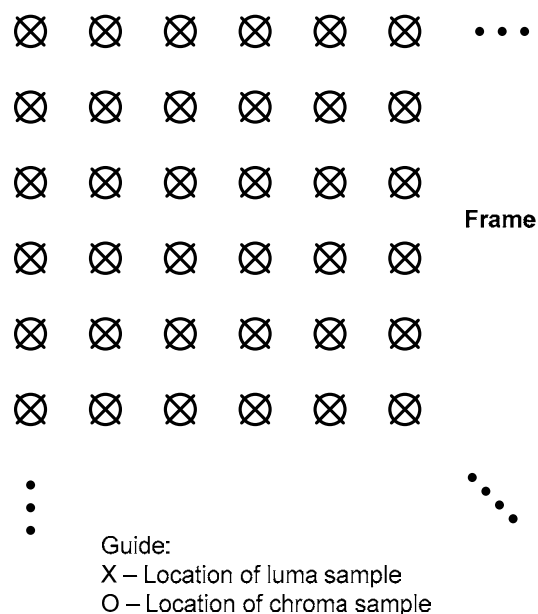


Figure 6-5 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a frame

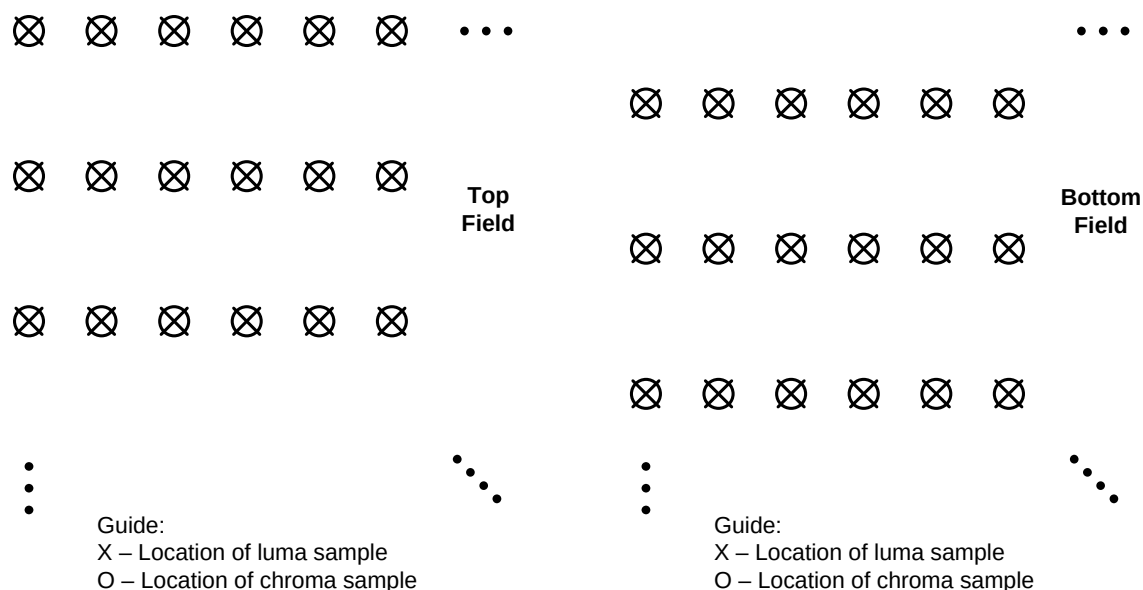


Figure 6-6 – Nominal vertical and horizontal sampling locations of 4:4:4 samples top and bottom fields

The samples are processed in units of macroblocks. The luma array for each macroblock is 16 samples in both width and height. The variables *MbWidthC* and *MbHeightC*, which specify the width and height, respectively, of the chroma arrays for each macroblock, are derived as follows:

- If *chroma_format_idc* is equal to 0 (monochrome) or *separate_colour_plane_flag* is equal to 1, *MbWidthC* and *MbHeightC* are both equal to 0.
- Otherwise, *MbWidthC* and *MbHeightC* are derived as

$$MbWidthC = 16 / SubWidthC \quad (6-1)$$

$$MbHeightC = 16 / SubHeightC \quad (6-2)$$

6.3 Spatial subdivision of pictures and slices

This subclause specifies how a picture is partitioned into slices and macroblocks. Pictures are divided into slices. A slice is a sequence of macroblocks, or, when macroblock-adaptive frame/field decoding is in use, a sequence of macroblock pairs.

Each macroblock is comprised of one 16x16 luma array and, when the chroma sampling format is not equal to 4:0:0 and *separate_colour_plane_flag* is equal to 0, two corresponding chroma sample arrays. When *separate_colour_plane_flag* is equal to 1, each macroblock is comprised of one 16x16 luma or chroma sample array. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-7.

When a picture is coded using three separate colour planes (*separate_colour_plane_flag* is equal to 1), a slice contains only macroblocks of one colour component being identified by the corresponding value of *colour_plane_id*, and each colour component array of a picture consists of slices having the same *colour_plane_id* value. Coded slices with different values of *colour_plane_id* within an access unit can be interleaved with each other under the constraint that for each value of *colour_plane_id*, the coded slice NAL units with that value *colour_plane_id* shall be in the order of increasing macroblock address for the first macroblock of each coded slice NAL unit.

NOTE – When *separate_colour_plane_flag* is equal to 0, each macroblock of a picture is contained in exactly one slice. When *separate_colour_plane_flag* is equal to 1, each macroblock of a colour component is contained in exactly one slice (i.e., information for each macroblock of a picture is present in exactly three slices and these three slices have different values of *colour_plane_id*).

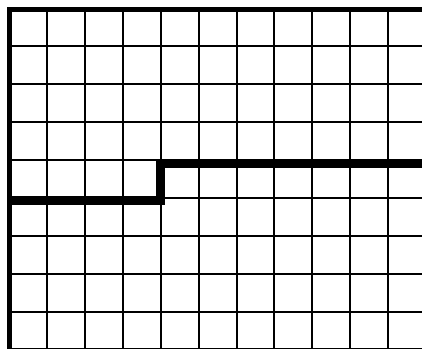


Figure 6-7 – A picture with 11 by 9 macroblocks that is partitioned into two slices

When macroblock-adaptive frame/field decoding is in use, the picture is partitioned into slices containing an integer number of macroblock pairs as shown in Figure 6-8. Each macroblock pair consists of two macroblocks.

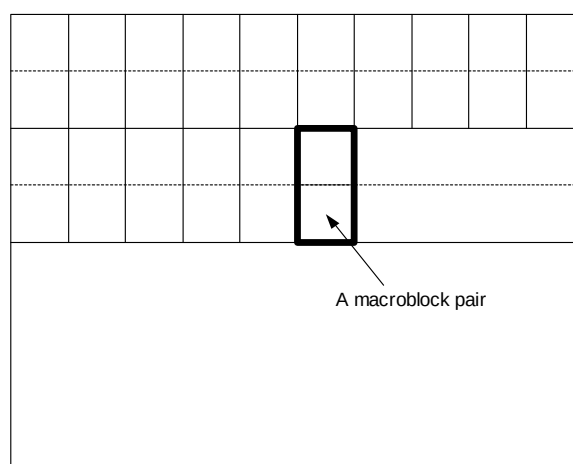


Figure 6-8 – Partitioning of the decoded frame into macroblock pairs

6.4 Inverse scanning processes and derivation processes for neighbours

This subclause specifies inverse scanning processes; i.e., the mapping of indices to locations, and derivation processes for neighbours.

6.4.1 Inverse macroblock scanning process

Input to this process is a macroblock address `mbAddr`.

Output of this process is the location (x, y) of the upper-left luma sample for the macroblock with address `mbAddr` relative to the upper-left sample of the picture.

The inverse macroblock scanning process is specified as follows:

- If `MbaffFrameFlag` is equal to 0,

$$x = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6-3)$$

$$y = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6-4)$$

- Otherwise (MbaffFrameFlag is equal to 1), the following ordered steps are specified:

1. The luma location (xO, yO) is derived by

$$xO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 0) \quad (6-5)$$

$$yO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 1) \quad (6-6)$$

2. Depending on the current macroblock the following applies:

- If the current macroblock is a frame macroblock

$$x = xO \quad (6-7)$$

$$y = yO + (\text{mbAddr} \% 2) * 16 \quad (6-8)$$

- Otherwise (the current macroblock is a field macroblock),

$$x = xO \quad (6-9)$$

$$y = yO + (\text{mbAddr} \% 2) \quad (6-10)$$

6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process

Macroblocks or sub-macroblocks may be partitioned, and the partitions are scanned for inter prediction as shown in Figure 6-9. The outer rectangles refer to the samples in a macroblock or sub-macroblock, respectively. The rectangles refer to the partitions. The number in each rectangle specifies the index of the inverse macroblock partition scan or inverse sub-macroblock partition scan.

The functions MbPartWidth(), MbPartHeight(), SubMbPartWidth(), and SubMbPartHeight() describing the width and height of macroblock partitions and sub-macroblock partitions are specified in Tables 7-13, 7-14, 7-17, and 7-18. MbPartWidth() and MbPartHeight() are set to appropriate values for each macroblock, depending on the macroblock type. SubMbPartWidth() and SubMbPartHeight() are set to appropriate values for each sub-macroblock of a macroblock with mb_type equal to P_8x8, P_8x8ref0, or B_8x8, depending on the sub-macroblock type.

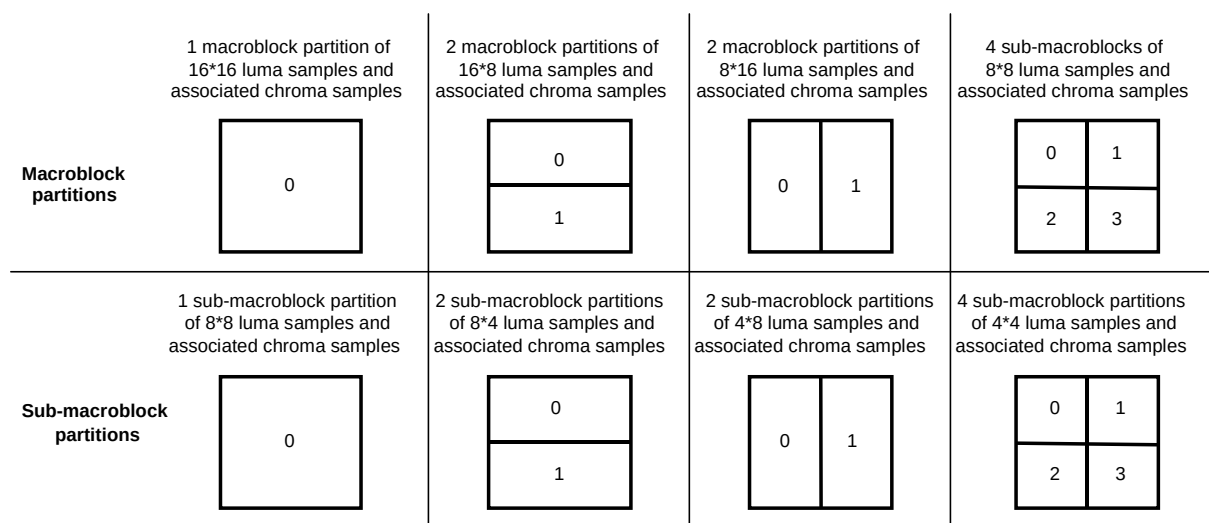


Figure 6-9 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans

6.4.2.1 Inverse macroblock partition scanning process

Input to this process is the index of a macroblock partition `mbPartIdx`.

Output of this process is the location (`x`, `y`) of the upper-left luma sample for the macroblock partition `mbPartIdx` relative to the upper-left sample of the macroblock.

The inverse macroblock partition scanning process is specified by

$$x = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb_type}), \text{MbPartHeight}(\text{mb_type}), 16, 0) \quad (6-11)$$

$$y = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb_type}), \text{MbPartHeight}(\text{mb_type}), 16, 1) \quad (6-12)$$

6.4.2.2 Inverse sub-macroblock partition scanning process

Inputs to this process are the index of a macroblock partition `mbPartIdx` and the index of a sub-macroblock partition `subMbPartIdx`.

Output of this process is the location (`x`, `y`) of the upper-left luma sample for the sub-macroblock partition `subMbPartIdx` relative to the upper-left sample of the sub-macroblock.

The inverse sub-macroblock partition scanning process is specified as follows:

- If `mb_type` is equal to `P_8x8`, `P_8x8ref0`, or `B_8x8`,

$$x = \text{InverseRasterScan}(\text{subMbPartIdx}, \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]), \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]), 8, 0) \quad (6-13)$$

$$y = \text{InverseRasterScan}(\text{subMbPartIdx}, \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]), \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]), 8, 1) \quad (6-14)$$

- Otherwise (`mb_type` is not equal to `P_8x8`, `P_8x8ref0`, or `B_8x8`),

$$x = \text{InverseRasterScan}(\text{subMbPartIdx}, 4, 4, 8, 0) \quad (6-15)$$

$$y = \text{InverseRasterScan}(\text{subMbPartIdx}, 4, 4, 8, 1) \quad (6-16)$$

6.4.3 Inverse 4x4 luma block scanning process

Input to this process is the index of a 4x4 luma block `luma4x4BlkIdx`.

Output of this process is the location (`x`, `y`) of the upper-left luma sample for the 4x4 luma block with index `luma4x4BlkIdx` relative to the upper-left luma sample of the macroblock.

Figure 6-10 shows the scan for the 4x4 luma blocks.

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 6-10 – Scan for 4x4 luma blocks

The inverse 4x4 luma block scanning process is specified by

$$x = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (6-17)$$

$$y = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (6-18)$$

6.4.4 Inverse 4x4 Cb or Cr block scanning process for ChromaArrayType equal to 3

This process is only invoked when `ChromaArrayType` is equal to 3.

The inverse 4x4 chroma block scanning process is identical to inverse 4x4 luma block scanning process as specified in subclause 6.4.3 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma4x4BlkIdx" with the term "cb4x4BlkIdx" or the term "cr4x4BlkIdx" in all places in subclause 6.4.3.

6.4.5 Inverse 8x8 luma block scanning process

Input to this process is the index of an 8x8 luma block `luma8x8BlkIdx`.

Output of this process is the location (`x`, `y`) of the upper-left luma sample for the 8x8 luma block with index `luma8x8BlkIdx` relative to the upper-left luma sample of the macroblock.

Figure 6-11 shows the scan for the 8x8 luma blocks.

0	1
2	3

Figure 6-11 – Scan for 8x8 luma blocks

The inverse 8x8 luma block scanning process is specified by:

$$x = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 0) \quad (6-19)$$

$$y = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 1) \quad (6-20)$$

6.4.6 Inverse 8x8 Cb or Cr block scanning process for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The inverse 8x8 chroma block scanning process is identical to inverse 8x8 luma block scanning process as specified in subclause 6.4.5 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma8x8BlkIdx" with the term "cb8x8BlkIdx" or the term "cr8x8BlkIdx" in all places in subclause 6.4.5.

6.4.7 Inverse 4x4 chroma block scanning process

Input to this process is the index of a 4x4 chroma block chroma4x4BlkIdx.

Output of this process is the location (x, y) of the upper-left chroma sample for a 4x4 chroma block with index chroma4x4BlkIdx relative to the upper-left chroma sample of the macroblock.

The inverse 4x4 chroma block scanning process is specified by

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (6-21)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (6-22)$$

6.4.8 Derivation process of the availability for macroblock addresses

Input to this process is a macroblock address mbAddr.

Output of this process is the availability of the macroblock mbAddr.

NOTE – The meaning of availability is determined when this process is invoked.

The macroblock is marked as available, unless any of the following conditions are true, in which case the macroblock is marked as not available:

- mbAddr < 0,
- mbAddr > CurrMbAddr,
- the macroblock with address mbAddr belongs to a different slice than the macroblock with address CurrMbAddr.

6.4.9 Derivation process for neighbouring macroblock addresses and their availability

This process can only be invoked when MbaffFrameFlag is equal to 0.

The outputs of this process are:

- mbAddrA: the address and availability status of the macroblock to the left of the current macroblock,
- mbAddrB: the address and availability status of the macroblock above the current macroblock,
- mbAddrC: the address and availability status of the macroblock above-right of the current macroblock,
- mbAddrD: the address and availability status of the macroblock above-left of the current macroblock.

Figure 6-12 shows the relative spatial locations of the macroblocks with mbAddrA, mbAddrB, mbAddrC, and mbAddrD relative to the current macroblock with CurrMbAddr.

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr	

Figure 6-12 – Neighbouring macroblocks for a given macroblock

Input to the process in subclause 6.4.8 is $\text{mbAddrA} = \text{CurrMbAddr} - 1$ and the output is whether the macroblock mbAddrA is available. In addition, mbAddrA is marked as not available when $\text{CurrMbAddr} \% \text{PicWidthInMbs}$ is equal to 0.

Input to the process in subclause 6.4.8 is $\text{mbAddrB} = \text{CurrMbAddr} - \text{PicWidthInMbs}$ and the output is whether the macroblock mbAddrB is available.

Input to the process in subclause 6.4.8 is $\text{mbAddrC} = \text{CurrMbAddr} - \text{PicWidthInMbs} + 1$ and the output is whether the macroblock mbAddrC is available. In addition, mbAddrC is marked as not available when $(\text{CurrMbAddr} + 1) \% \text{PicWidthInMbs}$ is equal to 0.

Input to the process in subclause 6.4.8 is $\text{mbAddrD} = \text{CurrMbAddr} - \text{PicWidthInMbs} - 1$ and the output is whether the macroblock mbAddrD is available. In addition, mbAddrD is marked as not available when $\text{CurrMbAddr} \% \text{PicWidthInMbs}$ is equal to 0.

6.4.10 Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames

This process can only be invoked when MbaffFrameFlag is equal to 1.

The outputs of this process are:

- mbAddrA: the address and availability status of the top macroblock of the macroblock pair to the left of the current macroblock pair,
- mbAddrB: the address and availability status of the top macroblock of the macroblock pair above the current macroblock pair,
- mbAddrC: the address and availability status of the top macroblock of the macroblock pair above-right of the current macroblock pair,
- mbAddrD: the address and availability status of the top macroblock of the macroblock pair above-left of the current macroblock pair.

Figure 6-13 shows the relative spatial locations of the macroblocks with mbAddrA, mbAddrB, mbAddrC, and mbAddrD relative to the current macroblock with CurrMbAddr.

mbAddrA, mbAddrB, mbAddrC, and mbAddrD have identical values regardless whether the current macroblock is the top or the bottom macroblock of a macroblock pair.

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr or	
	CurrMbAddr	

Figure 6-13 – Neighbouring macroblocks for a given macroblock in MBAFF frames

Input to the process in subclause 6.4.8 is $mbAddrA = 2 * (CurrMbAddr / 2 - 1)$ and the output is whether the macroblock mbAddrA is available. In addition, mbAddrA is marked as not available when $(CurrMbAddr / 2) \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.8 is $mbAddrB = 2 * (CurrMbAddr / 2 - PicWidthInMbs)$ and the output is whether the macroblock mbAddrB is available.

Input to the process in subclause 6.4.8 is $mbAddrC = 2 * (CurrMbAddr / 2 - PicWidthInMbs + 1)$ and the output is whether the macroblock mbAddrC is available. In addition, mbAddrC is marked as not available when $(CurrMbAddr / 2 + 1) \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.8 is $mbAddrD = 2 * (CurrMbAddr / 2 - PicWidthInMbs - 1)$ and the output is whether the macroblock mbAddrD is available. In addition, mbAddrD is marked as not available when $(CurrMbAddr / 2) \% PicWidthInMbs$ is equal to 0.

6.4.11 Derivation processes for neighbouring macroblocks, blocks, and partitions

Subclause 6.4.11.1 specifies the derivation process for neighbouring macroblocks.

Subclause 6.4.11.2 specifies the derivation process for neighbouring 8x8 luma blocks.

Subclause 6.4.11.3 specifies the derivation process for neighbouring 8x8 chroma blocks for ChromaArrayType equal to 3.

Subclause 6.4.11.4 specifies the derivation process for neighbouring 4x4 luma blocks.

Subclause 6.4.11.5 specifies the derivation process for neighbouring 4x4 chroma blocks.

Subclause 6.4.11.6 specifies the derivation process for neighbouring 4x4 chroma blocks for ChromaArrayType equal to 3.

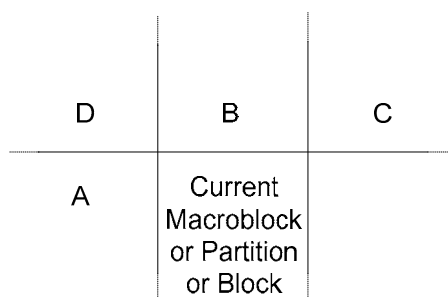
Subclause 6.4.11.7 specifies the derivation process for neighbouring partitions.

Table 6-2 specifies the values for the difference of luma location (xD, yD) for the input and the replacement for N in mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, cb8x8BlkIdxN, cr8x8BlkIdxN, luma4x4BlkIdxN, cb4x4BlkIdxN, cr4x4BlkIdxN, and chroma4x4BlkIdxN for the output. These input and output assignments are used in subclauses 6.4.11.1 to 6.4.11.7. The variable predPartWidth is specified when Table 6-2 is referred to.

Table 6-2 – Specification of input and output assignments for subclauses 6.4.11.1 to 6.4.11.7

N	xD	yD
A	–1	0
B	0	–1
C	predPartWidth	–1
D	–1	–1

Figure 6-14 illustrates the relative location of the neighbouring macroblocks, blocks, or partitions A, B, C, and D to the current macroblock, partition, or block, when the current macroblock, partition, or block is in frame coding mode.

**Figure 6-14 – Determination of the neighbouring macroblock, blocks, and partitions (informative)**

6.4.11.1 Derivation process for neighbouring macroblocks

Outputs of this process are:

- mbAddrA: the address of the macroblock to the left of the current macroblock and its availability status,
- mbAddrB: the address of the macroblock above the current macroblock and its availability status.

mbAddrN (with N being A or B) is derived as specified by the following ordered steps:

1. The difference of luma location (xD, yD) is set according to Table 6-2.
2. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) equal to (xD, yD), and the output is assigned to mbAddrN.

6.4.11.2 Derivation process for neighbouring 8x8 luma block

Input to this process is an 8x8 luma block index luma8x8BlkIdx.

The luma8x8BlkIdx specifies the 8x8 luma blocks of a macroblock in a raster scan.

Outputs of this process are:

- mbAddrA: either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock and its availability status,
- luma8x8BlkIdxA: the index of the 8x8 luma block to the left of the 8x8 block with index luma8x8BlkIdx and its availability status,
- mbAddrB: either equal to CurrMbAddr or the address of the macroblock above the current macroblock and its availability status,

- luma8x8BlkIdxB: the index of the 8x8 luma block above the 8x8 block with index luma8x8BlkIdx and its availability status.

mbAddrN and luma8x8BlkIdxN (with N being A or B) are derived as specified by the following ordered steps:

1. The difference of luma location (xD, yD) is set according to Table 6-2.
2. The luma location (xN, yN) is specified by

$$xN = (\text{luma8x8BlkIdx} \% 2) * 8 + xD \quad (6-23)$$

$$yN = (\text{luma8x8BlkIdx} / 2) * 8 + yD \quad (6-24)$$

3. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
4. The variable luma8x8BlkIdxN is derived as follows:
 - If mbAddrN is not available, luma8x8BlkIdxN is marked as not available.
 - Otherwise (mbAddrN is available), the derivation process for 8x8 luma block indices as specified in subclause 6.4.13.3 is invoked with the luma location (xW, yW) as the input and the output is assigned to luma8x8BlkIdxN.

6.4.11.3 Derivation process for neighbouring 8x8 chroma blocks for ChromaArrayType equal to 3

This process is only invoked when ChromaArrayType is equal to 3.

The derivation process for neighbouring 8x8 chroma block is identical to the derivation process for neighbouring 8x8 luma block as specified in subclause 6.4.11.2 when substituting the term "luma" with the term "Cb" or the term "Cr", and substituting the term "luma8x8BlkIdx" with the term "cb8x8BlkIdx" or the term "cr8x8BlkIdx" in all places in subclause 6.4.11.2.

6.4.11.4 Derivation process for neighbouring 4x4 luma blocks

Input to this process is a 4x4 luma block index luma4x4BlkIdx.

Outputs of this process are:

- mbAddrA: either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock and its availability status,
- luma4x4BlkIdxA: the index of the 4x4 luma block to the left of the 4x4 block with index luma4x4BlkIdx and its availability status,
- mbAddrB: either equal to CurrMbAddr or the address of the macroblock above the current macroblock and its availability status,
- luma4x4BlkIdxB: the index of the 4x4 luma block above the 4x4 block with index luma4x4BlkIdx and its availability status.

mbAddrN and luma4x4BlkIdxN (with N being A or B) are derived as specified by the following ordered steps:

1. The difference of luma location (xD, yD) is set according to Table 6-2.
2. The inverse 4x4 luma block scanning process as specified in subclause 6.4.3 is invoked with luma4x4BlkIdx as the input and (x, y) as the output.
3. The luma location (xN, yN) is specified by

$$xN = x + xD \quad (6-25)$$

$$yN = y + yD \quad (6-26)$$

4. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).