INTERNATIONAL STANDARD

ISO/IEC 14496-10

Third edition
2005-12-15

Corrected version
2006-03-01

AMENDMENT 2
2007-12-01

Information technology — Coding of audio-visual objects

Part 10: Advanced Video Coding

AMENDMENT 2: New profiles for professional applications

Technologies de l'information — Codage des objets audiovisuels

Partie 10: Codage visuel avancé

AMENDEMENT 2: Nouveaux profils pour applications professionnelles

Reference number
ISO/IEC 14496-10:2005/Amd.2:2007(E)

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

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.

Amendment 2 to ISO/IEC 14496-10:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Amendment 2 to ISO/IEC 14496-10:2005 is technically aligned with ITU-T Rec. H.264 (2005)/Amd.2 (2007), but is not published as identical text.

Information technology — Coding of audio-visual objects

Part 10: Advanced Video Coding

AMENDMENT 2: New profiles for professional applications

In subclause 0.4, replace the following:

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first (2003) 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 (the current specification) refers to the integrated text containing the first technical corrigendum (2004), the first amendment (the "Fidelity range extensions"), and an additional technical corrigendum (2005). In the ITU-T, the next published version after version 2 was version 4 (due to the completion of the drafting work for version 4 prior to the approval opportunity for a final version 3 text).

with:

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first (2003) approved version of this Recommendation | International Standard. The first published version in ISO/IEC corresponded to version 1.

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. The first fully-published version in the ITU-T was version 2, due to the development of the corrigendum during the publication process. Version 2 was also published in integrated form by ISO/IEC.

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). In both ITU-T and ISO/IEC, the next complete published version after version 2 was version 4.

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. In ISO/IEC, the changes from version 4 to version 5 were published as a corrigendum text.

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. In the ITU-T, the changes for versions 5 and 6 were approved and published as a single amendment.

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).

In the second paragraph of subclause 0.6, replace the sentence:

With the exception of the transform bypass mode of operation for lossless coding in 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.

with:

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.

In the third paragraph of subclause 0.7, replace the sentence:

Annex A specifies six profiles (Baseline, Main, Extended, High, High 10 and High 4:2:2), each being tailored to certain application domains, and defines the so-called levels of the profiles.

with:

Annex A specifies eleven profiles (Baseline, Main, Extended, 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.

Replace subclause 3.6 with the following:

3.6 **arbitrary slice order**: A *decoding order* of *slices* in which the *macroblock address* of the first *macroblock* of some *slice* of a *picture* may be less than the *macroblock address* of the first *macroblock* of some other preceding *slice* of the same *coded picture* or, in the case of a *picture* that is coded using three separate colour planes, some other preceding *slice* of the same colour plane.

Replace subclause 3.75 with the following:

3.75 **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*.

Replace subclause 3.136 with the following:

3.136 **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 addresses of the *macroblocks* are derived from the address of the first *macroblock* in a slice (as represented in the *slice header*), the *macroblock to slice group map*, and, when a *picture* is coded using three separate colour planes, a colour plane identifier.

In subclause 6.2 "Source, decoded, and output picture formats", make the following changes.

Replace the following:

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. 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

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

with:

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	-	-

Replace the following paragraph:

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

with:

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.

Replace the following paragraph:

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 12, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

with:

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.

Replace the following paragraph:

– If chroma_format_idc is equal to 0 (monochrome), MbWidthC and MbHeightC are both equal to 0 (as no chroma arrays are specified for monochrome video).

with:

– If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, MbWidthC and MbHeightC are both set equal to 0.

In subclause 6.3 "Spatial subdivision of pictures and slices", make the following changes:

Replace the following:

Each macroblock is comprised of one 16x16 luma array and, when the video format is not monochrome, two corresponding chroma sample arrays. 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.

with:

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.

Insert a new subclause 6.4.3.1 as follows:

6.4.3.1 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.

Insert a new subclause 6.4.4.1 as follows:

6.4.4.1 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 6.4.4 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 6.4.4.

In subclause 6.4.8 "Derivation processes for neighbouring macroblocks, blocks, and partitions", make the following changes.

Insert the following sentence after the paragraph starting with "Subclause 6.4.8.2 specifies*":*

Subclause 6.4.8.2.1 specifies the derivation process for neighbouring 8x8 chroma blocks for ChromaArrayType equal to 3.

Insert the following sentence after the paragraph starting with "Subclause 6.4.8.4 specifies".

Subclause 6.4.8.4.1 specifies the derivation process for neighbouring 4x4 chroma blocks for ChromaArrayType equal to 3.

Replace the paragraph starting with "Table 6-2 specifies*" with the following:*

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 6.4.8.1 to 6.4.8.5. The variable predPartWidth is specified when Table 6-2 is referred to.

Insert a new subclause 6.4.8.2.1 as follows:

6.4.8.2.1 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 6.4.8.2 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 6.4.8.2.

In subclause 6.4.8.4 "Derivation process for neighbouring 4x4 chroma blocks", make the following changes.

*Replace the paragraph starting with "*Depending on chroma_format_idc*" with the following:*

The position (x, y) of the upper-left sample of the 4x4 chroma block with index chroma4x4BlkIdx is derived by

x = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (6-25)
y = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (6-26)

Update all equation numbers in the subsequent subclauses due to the removal of equation 6-27 and 6-28.

Insert a new subclause 6.4.8.4.1 as follows:

6.4.8.4.1 Derivation process for neighbouring 4x4 chroma blocks for ChromaArrayType equal to 3

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

The derivation process for neighbouring 4x4 chroma block in 4:4:4 chroma format is identical to the derivation process for neighbouring 4x4 luma block as specified in subclause 6.4.8.3 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 6.4.8.3.

In subclause 7.3.2.1 "Sequence parameter set RBSP syntax", make the following changes.

*Replace the syntax element "***residual_colour_transform_flag***" with the syntax element "***separate_colour_plane_flag***".*

*Replace the expression "*profile_idc == 144*" with "*profile_idc == 44 || profile_idc == 244*".*

*Replace the expression "*i < 8*" with "*i < ((chroma_format_idc != 3) ? 8 : 12)*".*

*In subclause 7.3.2.2 "Picture parameter set RBSP syntax", replace the expression "*i < 6 + 2* transform_8x8_mode_flag*" with "*i < 6 + ((chroma_format_idc != 3) ? 2 : 6) * transform_8x8_mode_flag".

Replace the slice data partition B RBSP syntax table of subclause 7.3.2.9.2 "Slice data partition B RBSP syntax" with the following:

slice_data_partition_b_layer_rbsp( ) {	C	Descriptor
slice_id	All	ue(v)
if(separate_colour_plane_flag == 1)		
colour_plane_id	All	u(2)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data( ) /* only category 3 parts of slice_data( ) syntax */	3	
rbsp_slice_trailing_bits( )	3	
}		

Replace the slice data partition C RBSP syntax table of subclause 7.3.2.9.3 "Slice data partition C RBSP syntax" with the following:

slice_data_partition_c_layer_rbsp( ) {	C	Descriptor
slice_id	All	ue(v)
if(separate_colour_plane_flag = = 1)		
colour_plane_id	All	u(2)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data( ) /* only category 4 parts of slice_data( ) syntax */	4	
rbsp_slice_trailing_bits( )	4	
}		

Replace the slice header syntax table of subclause 7.3.3 "Slice header syntax" with the following:

slice_header( ) {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
if(separate_colour_plane_flag = = 1)		
colour_plane_id	2	u(2)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type = = 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type = = 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0 **]**	2	se(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1 **]**	2	se(v)
}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type = = B)		
direct_spatial_mv_pred_flag	2	u(1)
if(slice_type = = P | | slice_type = = SP | | slice_type = = B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		

num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering( )	2	
if((weighted_pred_flag && (slice_type == P || slice_type == SP)) || (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table( )	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
if(slice_type == SP || slice_type == SI) {		
if(slice_type == SP)		
sp_for_switch_flag	2	u(1)
slice_qs_delta	2	se(v)
}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
if(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)		
slice_group_change_cycle	2	u(v)
}		

In the syntax table of subclause 7.3.3.2 "Prediction weight table syntax", replace the syntax expression:

chroma_format_idc != 0

with:

ChromaArrayType != 0

(in all 3 occurrences)

In the syntax table of subclause 7.3.5.1 "Macroblock prediction syntax", replace the syntax expression

chroma_format_idc != 0

with:

ChromaArrayType == 1 || ChromaArrayType == 2

(in one occurrence)

Replace the content of subclause 7.3.5.3 "Residual data syntax" with the following:

residual( ) {	C	Descriptor
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
Intra16x16DCLevel = i16x16DClevel		
Intra16x16ACLevel = i16x16AClevel		
LumaLevel = level		
LumaLevel8x8 = level8x8		
if(ChromaArrayType == 1 || ChromaArrayType == 2) {		
NumC8x8 = 4 / (SubWidthC * SubHeightC)		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
if(CodedBlockPatternChroma & 3) /* chroma DC residual present*/		
residual_block(ChromaDCLevel[iCbCr], 4 * NumC8x8)	3 | 4	
else		
for(i = 0; i < 4 * NumC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
for(i8x8 = 0; i8x8 < NumC8x8; i8x8++)		
for(i4x4 = 0; i4x4 < 4; i4x4++)		
if(CodedBlockPatternChroma & 2) /* chroma AC residual present */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4],15)	3 | 4	
else		
for(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
} else if(ChromaArrayType == 3) {		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
CbIntra16x16DCLevel = i16x16DClevel		
CbIntra16x16ACLevel = i16x16AClevel		
CbLevel = level		
CbLevel8x8 = level8x8		
residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8)		
CrIntra16x16DCLevel = i16x16DClevel		

CrIntra16x16ACLevel = i16x16AClevel		
CrLevel = level		
CrLevel8x8 = level8x8		
}		
}		

Replace the content of subclause 7.3.5.3.2 "Residual block CABAC syntax" with the following:

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	Descriptor
if(maxNumCoeff != 64 || (ChromaArrayType == 3))		
coded_block_flag	3 | 4	ae(v)
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i **]**	3 | 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i **]**	3 | 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff - 1)		
coeff_abs_level_minus1[numCoeff - 1 **]**	3 | 4	ae(v)
coeff_sign_flag[numCoeff - 1 **]**	3 | 4	ae(v)
coeffLevel[numCoeff - 1] = (coeff_abs_level_minus1[numCoeff – 1] + 1) * (1 – 2 * coeff_sign_flag[numCoeff – 1])		
for(i = numCoeff - 2; i >= 0; i--)		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i **]**	3 | 4	ae(v)
coeff_sign_flag[i **]**	3 | 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) * (1 – 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

Insert a new subclause 7.3.5.3.3 as follows:

7.3.5.3.3 Residual luma data syntax

residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8) {	C	Descriptor
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(i16x16DClevel, 16)	3	
for(i8x8 = 0; i8x8 < 4; i8x8++)		
if(!transform_size_8x8_flag || !entropy_coding_mode_flag)		
for(i4x4 = 0; i4x4 < 4; i4x4++) {		
if(CodedBlockPatternLuma & (1 << i8x8))		
if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(i16x16AClevel[i8x8*4+ i4x4], 15)	3	
else		
residual_block(level[i8x8 * 4 + i4x4], 16)	3 | 4	
else if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
for(i = 0; i < 15; i++)		
i16x16AClevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
level[i8x8 * 4 + i4x4][i] = 0		
if(!entropy_coding_mode_flag && transform_size_8x8_flag)		
for(i = 0; i < 16; i++)		
level8x8[i8x8][4 * i + i4x4] = level[i8x8 * 4 + i4x4][i]		
}		
else if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(level8x8[i8x8], 64)	3 | 4	
else		
for(i = 0; i < 64; i++)		
level8x8[i8x8][i] = 0		
}		

In subclause 7.4.1.2.5 "Order of VCL NAL units and association to coded pictures", replace the following paragraph:

- Otherwise (arbitrary slice order is not allowed), the order of coded slice of an IDR picture NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of an IDR picture NAL unit.

with:

- Otherwise (arbitrary slice order is not allowed), the following applies.
  - If separate_colour_plane_flag is equal to 0, the order of coded slices of IDR picture NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of an IDR picture NAL unit.

– Otherwise (separate_colour_plane_flag is equal to 1), the order of coded slices of IDR picture NAL units for each value of colour_plane_id shall be in the order of increasing macroblock address for the first macroblock of each coded slice of the particular value of colour_plane_id of an IDR picture NAL unit.

NOTE – When separate_colour_plane_flag is equal to 1, the relative ordering of coded slices having different values of colour_plane_id is not constrained.

In subclause 7.4.2.1 "Sequence parameter set RBSP semantics", make the following changes.

Replace the paragraphs that state as follows:

NOTE – When more than one of constraint_set0_flag, constraint_set1_flag, or constraint_set2_flag are equal to 1, the bitstream obeys the constraints of all of the indicated subclauses of subclause A.2.

constraint_set3_flag indicates the following.

– If profile_idc is equal to 66, 77, or 88 and level_idc is equal to 11, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for level 1b and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in Annex A for level 1b.

– Otherwise (profile_idc is equal to 100, 110, 122, or 144 or level_idc is not equal to 11), the value of 1 for constraint_set3_flag is reserved for future use by ITU-T | ISO/IEC. constraint_set3_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard when profile_idc is equal to 100, 110, 122, or 144 or level_idc is not equal to 11. Decoders conforming to this Recommendation | International Standard shall ignore the value of constraint_set3_flag when profile_idc is equal to 100, 110, 122, or 144 or level_idc is not equal to 11.

with:

NOTE – When one or more than one of constraint_set0_flag, constraint_set1_flag, or constraint_set2_flag are equal to 1, the bitstream must obey the constraints of all of the indicated subclauses of subclause A.2. When profile_idc is equal to 44, 100, 110, 122, or 244, the values of constraint_set0_flag, constraint_set1_flag, and constraint_set2_flag must all be equal to 0.

constraint_set3_flag indicates the following.

– If profile_idc is equal to 66, 77, or 88 and level_idc is equal to 11, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for level 1b and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in Annex A for level 1b.

– Otherwise, if profile_idc is equal to 100 or 110, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for the High 10 Intra profile, and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey these corresponding constraints.

– Otherwise, if profile_idc is equal to 122, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for the High 4:2:2 Intra profile, and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey these corresponding constraints.

– Otherwise, if profile_idc is equal to 44, constraint_set3_flag shall be equal to 1. When profile_idc is equal to 44, the value of 0 for constraint_set3_flag is forbidden.

– Otherwise, if profile_idc is equal to 244, constraint_set3_flag equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for the High 4:4:4 Intra profile, and constraint_set3_flag equal to 0 indicates that the bitstream may or may not obey these corresponding constraints.

– Otherwise (profile_idc is equal to 66, 77, 88 and level_idc is not equal to 11), the value of 1 for constraint_set3_flag is reserved for future use by ITU-T | ISO/IEC. constraint_set3_flag shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard when profile_idc is equal

to 66, 77, or 88 and level_idc is not equal to 11. Decoders conforming to this Recommendation | International Standard shall ignore the value of constraint_set3_flag when profile_idc is equal to 66, 77, or 88 and level_idc is not equal to 11.

Delete the paragraph starting with "**residual_colour_transform_flag".**

*Add the following paragraph before the paragraph starting with "***bit_depth_luma_minus8***".*

separate_colour_plane_flag equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. separate_colour_plane_flag equal to 0 specifies that the three colour components of the 4:4:4 chroma format are not coded separately. When separate_colour_plane_flag is not present, it shall be inferred to be equal to 0. When separate_colour_plane_flag is equal to 1, the primary coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y or Cb or Cr) that conforms to the monochrome coding syntax. In this case, each picture is associated with a specific colour_plane_id value.

NOTE – There is no dependency in decoding processes between the colour planes having different colour_plane_id values. For example, the decoding process of a monochrome picture with one value of colour_plane_id does not use any date from monochrome pictures having different values of colour_plane_id for inter prediction.

Depending on the value of separate_colour_plane_flag, assign the value of the variable ChromaArrayType as follows.

– If separate_colour_plane_flag is equal to 0, ChromaArrayType is set equal to chroma_format_idc.

– Otherwise (separate_colour_plane_flag is equal to 1), ChromaArrayType is set equal to 0.

Replace the paragraph starting with "**bit_depth_luma_minus8"** *with the following:*

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quantisation parameter range offset $QpBdOffset_Y$, as specified by

$$BitDepth_Y = 8 + bit\_depth\_luma\_minus8 \quad (7\text{-}1)$$

$$QpBdOffset_Y = 6 * bit\_depth\_luma\_minus8 \quad (7\text{-}2)$$

When bit_depth_luma_minus8 is not present, it shall be inferred to be equal to 0. bit_depth_luma_minus8 shall be in the range of 0 to 6, inclusive.

Replace the paragraph starting with "**bit_depth_chroma_minus8"** *with the following.*

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset $QpBdOffset_C$, as specified by

$$BitDepth_C = 8 + bit\_depth\_chroma\_minus8 \quad (7\text{-}3)$$

$$QpBdOffset_C = 6 * bit\_depth\_chroma\_minus8 \quad (7\text{-}4)$$

When bit_depth_chroma_minus8 is not present, it shall be inferred to be equal to 0. bit_depth_chroma_minus8 shall be in the range of 0 to 6, inclusive.

NOTE – The value of bit_depth_chroma_minus8 is not used in the decoding process when ChromaArrayType is equal to 0. In particular, when separate_colour_plane_flag is equal to 1, each colour plane is decoded as a distinct monochrome picture using the luma component decoding process (except for the selection of scaling matrices) and the luma bit depth is used for all three colour components.

*Replace the paragraph starting with "***seq_scaling_matrix_present_flag equal to 1***" with the following:*

seq_scaling_matrix_present_flag equal to 1 specifies that the flags seq_scaling_list_present_flag[i] for i = 0..11 are present. seq_scaling_matrix_present_flag equal to 0 specifies that these flags are not present and the sequence-level scaling list specified by Flat_4x4_16 shall be inferred for i = 0..5 and the sequence-level scaling list specified by Flat_8x8_16 shall be inferred for i = 6..11. When seq_scaling_matrix_present_flag is not present, it shall be inferred to be equal to 0.

Replace Table 7–2 with the following:

Table 7-2 – Assignment of mnemonic names to scaling list indices and specification of fall-back rule

Value of scaling list index	Mnemonic name	Block size	MB prediction type	Component	Scaling list fall-back rule set A	Scaling list fall-back rule set B	Default scaling list
0	Sl_4x4_Intra_Y	4x4	Intra	Y	default scaling list	sequence-level scaling list	Default_4x4_Intra
1	Sl_4x4_Intra_Cb	4x4	Intra	Cb	scaling list for i = 0	scaling list for i = 0	Default_4x4_Intra
2	Sl_4x4_Intra_Cr	4x4	Intra	Cr	scaling list for i = 1	scaling list for i = 1	Default_4x4_Intra
3	Sl_4x4_Inter_Y	4x4	Inter	Y	default scaling list	sequence-level scaling list	Default_4x4_Inter
4	Sl_4x4_Inter_Cb	4x4	Inter	Cb	scaling list for i = 3	scaling list for i = 3	Default_4x4_Inter
5	Sl_4x4_Inter_Cr	4x4	Inter	Cr	scaling list for i = 4	scaling list for i = 4	Default_4x4_Inter
6	Sl_8x8_Intra_Y	8x8	Intra	Y	default scaling list	sequence-level scaling list	Default_8x8_Intra
7	Sl_8x8_Inter_Y	8x8	Inter	Y	default scaling list	sequence-level scaling list	Default_8x8_Inter
8	Sl_8x8_Intra_Cb	8x8	Intra	Cb	scaling list for i = 6	scaling list for i = 6	Default_8x8_Intra
9	Sl_8x8_Inter_Cb	8x8	Inter	Cb	scaling list for i = 7	scaling list for i = 7	Default_8x8_Inter
10	Sl_8x8_Intra_Cr	8x8	Intra	Cr	scaling list for i = 8	scaling list for i = 8	Default_8x8_Intra
11	Sl_8x8_Inter_Cr	8x8	Inter	Cr	scaling list for i = 9	scaling list for i = 9	Default_8x8_Inter

*Replace the paragraph starting with "***frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, frame_crop_bottom_offset"** *with the following.*

frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, frame_crop_bottom_offset specify the samples of the pictures in the coded video sequence that are output from the decoding process, in terms of a rectangular region specified in frame coordinates for output.

The variables CropUnitX and CropUnitY are derived as follows:

– If ChromaArrayType is equal to 0, CropUnitX and CropUnitY are derived as

CropUnitX = 1 (7-16)
CropUnitY = 2 – frame_mbs_only_flag (7-17)

– Otherwise (ChromaArrayType is equal to 1, 2, or 3), CropUnitX and CropUnitY are derived as

CropUnitX = SubWidthC (7-18)
CropUnitY = SubHeightC * (2 – frame_mbs_only_flag) (7-19)

The frame cropping rectangle contains luma samples with horizontal frame coordinates from CropUnitX * frame_crop_left_offset to $PicWidthInSamples_L$ – (CropUnitX * frame_crop_right_offset + 1) and vertical frame coordinates from CropUnitY * frame_crop_top_offset to (16 * FrameHeightInMbs) – (CropUnitY * frame_crop_bottom_offset + 1), inclusive. The value of frame_crop_left_offset shall be in the range of 0 to ($PicWidthInSamples_L$ / CropUnitX) – (frame_crop_right_offset + 1), inclusive; and the value of frame_crop_top_offset shall be in the range of 0 to (16 * FrameHeightInMbs / CropUnitY) – (frame_crop_bottom_offset + 1), inclusive.

When frame_cropping_flag is equal to 0, the values of frame_crop_left_offset, frame_crop_right_offset, frame_crop_top_offset, and frame_crop_bottom_offset shall be inferred to be equal to 0.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having frame coordinates (x / SubWidthC, y / SubHeightC), where (x, y) are the frame coordinates of the specified luma samples.

For decoded fields, the specified samples of the decoded field are the samples that fall within the rectangle specified in frame coordinates.

In subclause 7.4.2.1.2 "Sequence parameter set extension RBSP semantics", replace the text:

The value of chroma_format_idc

with:

The value of chroma_format_idc and the value of ChromaArrayType

(in one occurrence)

Add the following note at the end of subclause 7.4.2.2 "Picture parameter set RBSP semantics":

NOTE – The values of bit_depth_chroma_minus8, chroma_qp_index_offset and second_chroma_qp_index_offset are not used in the decoding process when ChromaArrayType is equal to 0. In particular, when separate_colour_plane_flag is equal to 1, each colour plane is decoded as a distinct monochrome picture using the luma component decoding process (except for the selection of scaling matrices), including the application of the luma quantisation parameter derivation process without application of an offset for the decoding of the pictures having colour_plane_id not equal to 0.

In subclause 7.4.2.9.1 "Slice data partition A RBSP semantics", replace the paragraph that states as follows:

slice_id identifies the slice associated with the data partition. Each slice shall have a unique slice_id value within the coded picture that contains the slice. When arbitrary slice order is not allowed as specified in Annex A, the first slice of a

coded picture, in decoding order, shall have slice_id equal to 0 and the value of slice_id shall be incremented by one for each subsequent slice of the coded picture in decoding order.

with:

slice_id identifies the slice associated with the data partition. The value of slice_id is constrained as follows.

- If separate_colour_plane_flag is equal to 0, the following applies.
  - If arbitrary slice order is not allowed as specified in Annex A, the first slice of a coded picture, in decoding order, shall have slice_id equal to 0 and the value of slice_id shall be incremented by one for each subsequent slice of the coded picture in decoding order.
  - Otherwise (arbitrary slice order is allowed), each slice shall have a unique slice_id value within the set of slices of the coded picture.
- Otherwise (separate_colour_plane_flag is equal to 1), the following applies.
  - If arbitrary slice order is not allowed as specified in Annex A, the first slice of a coded picture having each value of colour_plane_id, in decoding order, shall have slice_id equal to 0 and the value of slice_id shall be incremented by one for each subsequent slice of the coded picture having the same value of colour_plane_id, in decoding order.
  - Otherwise (arbitrary slice order is allowed) each slice shall have a unique slice_id value within each set of slices of the coded picture that have the same value of colour_plane_id.

In subclause 7.4.2.9.2 "Slice data partition B RBSP semantics", insert the following after the paragraph starting with "**slice_id**".

colour_plane_id specifies the colour plane associated with the current slice RBSP when separate_colour_plane_flag is equal to 1. The value of colour_plane_id shall be in the range of 0 to 2, inclusive. colour_plane_id equal to 0, 1, and 2 correspond to the Y, Cb, and Cr planes, respectively.

NOTE – There is no dependency between the decoding processes of pictures having different values of colour_plane_id.

In subclause 7.4.2.9.3 "Slice data partition C RBSP semantics", insert the following after the paragraph starting with "**slice_id**":

colour_plane_id has the same semantics as specified in subclause 7.4.2.9.2.

In subclause 7.4.3 "Slice header semantics", make the following changes.

Replace the following paragraph:

first_mb_in_slice specifies the address of the first macroblock in the slice. When arbitrary slice order is not allowed as specified in Annex A, the value of first_mb_in_slice shall not be less than the value of first_mb_in_slice for any other slice of the current picture that precedes the current slice in decoding order.

with:

first_mb_in_slice specifies the address of the first macroblock in the slice. When arbitrary slice order is not allowed as specified in Annex A, the value of first_mb_in_slice is constrained as follows.

- If separate_colour_plane_flag is equal to 0, the value of first_mb_in_slice shall not be less than the value of first_mb_in_slice for any other slice of the current picture that precedes the current slice in decoding order.

– Otherwise (separate_colour_plane_flag is equal to 1), the value of first_mb_in_slice shall not be less than the value of first_mb_in_slice for any other slice of the current picture that precedes the current slice in decoding order and has the same value of colour_plane_id.

Insert the following after the paragraph starting with "**pic_parameter_set_id**".

colour_plane_id has the same semantics as specified in subclause 7.4.2.9.2.

In subclause 7.4.5 "Macroblock layer semantics", make the following changes.

*Replace the paragraph starting with "***transform_size_8x8_flag** equal to 1" *with the following.*

transform_size_8x8_flag equal to 1 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 8x8 blocks shall be invoked for luma samples, and when ChromaArrayType == 3 also for Cb and Cr samples. transform_size_8x8_flag equal to 0 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 4x4 blocks shall be invoked for luma samples, and when ChromaArrayType == 3 also for Cb and Cr samples. When transform_size_8x8_flag is not present in the bitstream, it shall be inferred to be equal to 0.

Replace the paragraph starting with **"**To each Intra_16x16*" with the following:*

To each Intra_16x16 prediction macroblock, an Intra16x16PredMode is assigned, which specifies the Intra_16x16 prediction mode.

The meaning of the variables CodedBlockPatternChroma and CodedBlockPatternLuma is specified with the semantics of coded_block_pattern.

Replace the following paragraphs:

pcm_sample_luma[i] is a sample value. The first pcm_sample_luma[i] values represent luma sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepth_Y$. When profile_idc is not equal to 100, 110, 122, or 144, pcm_sample_luma[i] shall not be equal to 0.

pcm_sample_chroma[i] is a sample value. The first MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cb sample values in the raster scan within the macroblock and the remaining MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cr sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepth_C$. When profile_idc is not equal to 100, 110, 122, or 144, pcm_sample_chroma[i] shall not be equal to 0.

with:

pcm_sample_luma[i] is a sample value. The first pcm_sample_luma[i] values represent luma sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepth_Y$. When profile_idc is not equal to 44, 100, 110, 122, or 244, pcm_sample_luma[i] shall not be equal to 0.

pcm_sample_chroma[i] is a sample value. The first MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cb sample values in the raster scan within the macroblock and the remaining MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cr sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is $BitDepth_C$. When profile_idc is not equal to 44, 100, 110, 122, or 244, pcm_sample_chroma[i] shall not be equal to 0.

*Delete the paragraph starting with "*When coded_block_pattern is present, CodedBlockPatternLuma specifies*".*

*Insert the following paragraph before the paragraph starting with "*mb_qp_delta"*:*

Depending on the value of ChromaArrayType, the following meaning is assigned to CodedBlockPatternLuma and the following restrictions are imposed on CodedBlockPatternChroma.

– If ChromaArrayType is equal to 0, CodedBlockPatternLuma specifies whether and where, for the luma component, non-zero AC transform coefficient levels are present and the derivation of the variable CodedBlockPatternChroma shall result in a value that is equal to 0.
– Otherwise, if ChromaArrayType is equal to 1 or 2, CodedBlockPatternLuma specifies whether, for the luma component, non-zero AC transform coefficient levels are present and CodedBlockPatternChroma may be in the range of 0 to 2, inclusive.
– Otherwise (ChromaArrayType is equal to 3), CodedBlockPatternLuma specifies whether and where, for the luma and the Cb and Cr component, non-zero AC transform coefficient levels are present and the derivation of the variable CodedBlockPatternChroma shall result in a value that is equal to 0.

In subclause 7.4.5.1 "Macroblock prediction semantics", make the following changes.

Replace the paragraph starting with **"prev_intra4x4_pred_mode_flag"** *with the following:*

prev_intra4x4_pred_mode_flag[luma4x4BlkIdx] and **rem_intra4x4_pred_mode**[luma4x4BlkIdx] specify the Intra_4x4 prediction of the 4x4 luma block with index luma4x4BlkIdx = 0..15. When ChromaArrayType is equal to 3, prev_intra4x4_pred_mode_flag[luma4x4BlkIdx] and rem_intra4x4_pred_mode[luma4x4BlkIdx] also specify the Intra_4x4 prediction of the 4x4 Cb block and 4x4 Cr block with luma4x4BlkIdx equal to index cb4x4BlkIdx = 0..15 or cr4x4BlkIdx = 0..15.

Replace the paragraph starting with **"prev_intra8x8_pred_mode_flag"** *with the following:*

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] and **rem_intra8x8_pred_mode**[luma8x8BlkIdx] specify the Intra_8x8 prediction of the 8x8 luma block with index luma8x8BlkIdx = 0..3. When ChromaArrayType is equal to 3, prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] and rem_intra8x8_pred_mode[luma8x8BlkIdx] also specify the Intra_8x8 prediction of the 8x8 Cb block and 8x8 Cr block with index luma8x8BlkIdx equal to cb8x8BlkIdx = 0..3 or cr8x8BlkIdx = 0..3.

Replace the content of subclause 7.4.5.3 "Residual data semantics" with the following:

The syntax structure residual_block( ), which is used for parsing the transform coefficient levels, is assigned as follows.

– If entropy_coding_mode_flag is equal to 0, residual_block is set equal to residual_block_cavlc, which is used for parsing the syntax elements for transform coefficient levels.
– Otherwise (entropy_coding_mode_flag is equal to 1), residual_block is set equal to residual_block_cabac, which is used for parsing the syntax elements for transform coefficient levels.

The syntax structure residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8) is used with the variables in brackets being its output and being assigned as follows. Intra16x16DCLevel is set equal to i16x16DClevel, Intra16x16ACLevel is set equal to i16x16AClevel, LumaLevel is set equal to level, and LumaLevel8x8 is set equal to level8x8.

When ChromaArrayType is equal to 1 or 2, the following applies.

– For each chroma component, indexed by iCbCr = 0..1, the DC transform coefficient levels of the 4 * NumC8x8 4x4 chroma blocks are parsed into the iCbCr-th list ChromaDCLevel[iCbCr].

– For each of the 4x4 chroma blocks, indexed by i4x4 = 0..3 and i8x8 = 0..NumC8x8 – 1, of each chroma component, indexed by iCbCr = 0..1, the 15 AC transform coefficient levels are parsed into the (i8x8*4 + i4x4)-th list of the iCbCr-th chroma component ChromaACLevel[iCbCr][i8x8*4 + i4x4].

When ChromaArrayType is equal to 3, the following applies.

– The syntax structure residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8) is used for the Cb component with the variables in brackets being its output and being assigned as follows. CbIntra16x16DCLevel is set equal to i16x16DClevel, CbIntra16x16ACLevel is set equal to i16x16AClevel, CbLevel is set equal to level, and CbLevel8x8 is set equal to level8x8.

– The syntax structure residual_luma(i16x16DClevel, i16x16AClevel, level, level8x8) is used for the Cr component with the variables in brackets being its output and being assigned as follows. CrIntra16x16DCLevel is set equal to i16x16DClevel, CrIntra16x16ACLevel is set equal to i16x16AClevel, CrLevel is set equal to level, and CrLevel8x8 is set equal to level8x8.

In subclause 7.4.5.3.2 "Residual block CABAC semantics", replace the paragraph starting with "**coded_block_flag**" *with the following:*

coded_block_flag specifies whether the block contains non-zero transform coefficient levels as follows.

– If coded_block_flag is equal to 0, the block contains no non-zero transform coefficient levels.

– Otherwise (coded_block_flag is equal to 1), the block contains at least one non-zero transform coefficient level.

When maxNumCoeff is equal to 64 and ChromaArrayType is not equal to 3, coded_block_flag is set equal to 1.

Insert a new subclause 7.4.5.3.3 as follows:

7.3.5.3.3 Residual luma data semantics

Output of this syntax structure are the variables i16x16DClevel, i16x16AClevel, level, and level8x8.

The syntax structure residual_block( ), which is used for parsing the transform coefficient levels, is assigned as follows.

– If entropy_coding_mode_flag is equal to 0, residual_block is set equal to residual_block_cavlc, which is used for parsing the syntax elements for transform coefficient levels.

– Otherwise (entropy_coding_mode_flag is equal to 1), residual_block is set equal to residual_block_cabac, which is used for parsing the syntax elements for transform coefficient levels.

Depending on mb_type the syntax structure residual_block(coeffLevel, maxNumCoeff) is used with the arguments coeffLevel, which is a list containing the maxNumCoeff transform coefficient levels that are parsed in residual_block( ) and maxNumCoeff as follows. Depending on MbPartPredMode(mb_type, 0), the following applies.

– If MbPartPredMode(mb_type, 0) is equal to Intra_16x16, the transform coefficient levels are parsed into the list i16x16DClevel and into the 16 lists i16x16AClevel[i]. i16x16DClevel contains the 16 transform coefficient levels of the DC transform coefficient levels for each 4x4 luma block. For each of the 16 4x4 luma blocks indexed by i = 0..15, the 15 AC transform coefficients levels of the i-th block are parsed into the i-th list i16x16AClevel[i].

– Otherwise (MbPartPredMode(mb_type, 0) is not equal to Intra_16x16), the following applies.

  – If transform_size_8x8_flag is equal to 0, for each of the 16 4x4 luma blocks indexed by i = 0..15, the 16 transform coefficient levels of the i-th block are parsed into the i-th list level[i].

– Otherwise (transform_size_8x8_flag is equal to 1), for each of the 4 8x8 luma blocks indexed by i8x8 = 0..3, the following applies.

  – If entropy_coding_mode_flag is equal to 0, first for each of the 4 4x4 luma blocks indexed by i4x4 = 0..3, the 16 transform coefficient levels of the i4x4-th block are parsed into the (i8x8 * 4 + i4x4)-th list level[i8x8 * 4 + i4x4]. Then, the 64 transform coefficient levels of the i8x8-th 8x8 luma block which are indexed by 4 * i + i4x4, where i = 0..15 and i4x4 = 0..3, are derived as level8x8[i8x8][4 * i + i4x4] = level[i8x8 * 4 + i4x4][i].

    NOTE – The 4x4 luma blocks with luma4x4BlkIdx = i8x8 * 4 + i4x4 containing every fourth transform coefficient level of the corresponding i8x8-th 8x8 luma block with offset i4x4 are assumed to represent spatial locations given by the inverse 4x4 luma block scanning process in subclause 6.4.3.

  – Otherwise (entropy_coding_mode_flag is equal to 1), the 64 transform coefficient levels of the i8x8-th block are parsed into the i8x8-th list level8x8[i8x8].

In clause 8 and all subclauses of clause 8, make the following changes.

*Replace all occurrences of "*chroma_format_idc*" with "*ChromaArrayType*".*

Delete all occurrences of "(monochrome)*".*

In clause 8, make the following changes.

After the first sentence, insert the following.

Depending on the value of chroma_format_idc, the number of sample arrays of the current picture is as follows.

– If chroma_format_idc is equal to 0, the current picture consists of 1 sample array S_L.

– Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

*Replace the paragraph that starts with "*Each picture referred*" with the following:*

Each picture referred to in this clause is a complete or part of a primary coded picture. Each slice referred to in this clause is a slice of a primary picture. Each slice data partition referred to in this clause is a slice data partition of a primary picture.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows.

– If separate_colour_plane_flag is equal to 0, the decoding process is invoked a single time with the current picture being the output.

– Otherwise (separate_colour_plane_flag is equal to 1), the decoding process is invoked three times. Inputs to the decoding process are all NAL units of the primary coded picture with identical value of colour_plane_id. The decoding process of NAL units with a particular value of colour_plane_id is specified as if only a coded video sequence with monochrome colour format with that particular value of colour_plane_id would be present in the bitstream. The output of each of the three decoding processes is assigned to the 3 sample arrays of the current picture with the NAL units with colour_plane_id equal to 0 being assigned to S_L, the NAL units with colour_plane_id equal to 1 being assigned to S_{Cb}, and the NAL units with colour_plane_id equal to 2 being assigned to S_{Cr}.

NOTE – The variable ChromaArrayType is derived as 0 when separate_colour_plane_flag is equal to 1 and chroma_format_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures with chroma_format_idc being equal to 0.

In subclause 8.3.4 "Intra prediction process for chroma samples", after the second paragraph, insert the following:

Depending on the value of ChromaArrayType, the following applies.

– If ChromaArrayType is equal to 3, the Intra prediction chroma samples for the current macroblock $pred_{Cb}$[x, y] and $pred_{Cr}$[x, y] are derived using the Intra prediction process for chroma samples with ChromaArrayType equal to 3 as specified in subclause 8.3.4.5.

– Otherwise (ChromaArrayType is equal to 1 or 2), the following text specifies the Intra prediction chroma samples for the current macroblock $pred_{Cb}$[x, y] and $pred_{Cr}$[x, y].

In subclause 8.3.4.1 "Specification of Intra_Chroma_DC prediction mode", make the following changes:

Replace the following:

– Depending on chroma_format_idc, the position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx is derived as follows

  – If chroma_format_idc is equal to 1 or 2, the following applies

$$xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) \quad (8\text{-}124)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) \quad (8\text{-}125)$$

  – Otherwise (chroma_format_idc is equal to 3), the following applies

$$xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 0) \quad (8\text{-}126)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 1) \quad (8\text{-}127)$$

with:

– The position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx is derived as

$$xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) \quad (8\text{-}124)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) \quad (8\text{-}125)$$

– Depending on the values of xO and yO, the following applies.

Indent all succeeding If/Otherwise statements as necessary.

Update all following equation numbers in clause 8 as necessary.

Insert a new subclause 8.3.4.5 as follows:

8.3.4.5 Intra prediction for chroma samples with ChromaArrayType equal to 3

This process is invoked when ChromaArrayType is equal to 3. This process is invoked for I and SI macroblock types. It specifies how the Intra prediction chroma samples for the current macroblock are derived when ChromaArrayType is equal to 3.

Inputs to this process are constructed samples prior to the deblocking filter process from neighbouring Cb and Cr blocks and for Intra_NxN (where NxN is equal to 4x4 or 8x8) prediction mode, the associated values of IntraNxNPredMode from neighbouring macroblocks.

Outputs of this process are the Intra prediction samples of the Cb and Cr components of the macroblock or in case of the Intra_NxN prediction process, the outputs are NxN Cb sample arrays as part of the 16x16 Cb array of prediction samples of the macroblock, and NxN Cb sample arrays as part of the 16x16 Cb array of prediction samples of the macroblock.

Each Cb, Cr, and luma block with the same block index of the macroblock shall use the same prediction mode. The prediction mode is applied to each of the Cb and Cr blocks separately. The process specified in this subclause is invoked for each Cb and Cr block.

Depending on the macroblock prediction mode, the following applies.

- If the macroblock prediction mode is equal to Intra_4x4, the following applies.
  - The same process described in subclause 8.3.1 shall also be applied to Cb or Cr samples, substituting luma with Cb or Cr, substituting luma4x4BlkIdx with cb4x4BlkIdx or cr4x4BlkIdx, and substituting $pred4x4_L$ with $pred4x4_{Cb}$ or $pred4x4_{Cr}$.
  - The output variable Intra4x4PredMode[luma4x4BlkIdx] from the process described in subclause 8.3.1.1 shall also be used for the 4x4 Cb or 4x4 Cr blocks with index luma4x4BlkIdx equal to index cb4x4BlkIdx or cr4x4BlkIdx.
  - The process to derive prediction Cb or Cr samples shall be identical to the process described in subclause 8.3.1.2 and its subsequent subclauses when substituting luma with Cb or Cr, and substituting $pred4x4_L$ with $pred4x4_{Cb}$ or $pred4x4_{Cr}$.
- Otherwise, if the macroblock prediction mode is equal to Intra_8x8, the following applies.
  - The same process described in subclause 8.3.2 shall also be applied to Cb or Cr samples, substituting luma with Cb or Cr, substituting luma8x8BlkIdx with cb8x8BlkIdx or cr8x8BlkIdx, and substituting $pred8x8_L$ with $pred8x8_{Cb}$ or $pred8x8_{Cr}$.
  - The output variable Intra8x8PredMode[luma8x8BlkIdx] from the process described in subclause 8.3.2.1 shall also be used for the 8x8 Cb or 8x8 Cr blocks with index luma8x8BlkIdx equal to index cb8x8BlkIdx or cr8x8BlkIdx.
  - The process to derive prediction Cb or Cr samples shall be identical to the process described in subclause 8.3.2.2 and its subsequent subclauses when substituting luma with Cb or Cr, and substituting $pred8x8_L$ with $pred8x8_{Cb}$ or $pred8x8_{Cr}$.
- Otherwise, if the macroblock prediction mode is equal to Intra_16x16, the following applies.
  - The same process described in subclause 8.3.3 and in the subsequent subclause 8.3.3.1 to 8.3.3.4 shall also be applied to Cb or Cr samples, substituting luma with Cb or Cr, and substituting $pred_L$ with $pred_{Cb}$ or $pred_{Cr}$.

*In subclause 8.4 "Inter prediction process", replace the paragraph starting with "*Outputs of this process are*" with the following:*

Outputs of this process are Inter prediction samples for the current macroblock that are a 16x16 array $pred_L$ of luma samples and when ChromaArrayType is not equal to 0 two (MbWidthC)x(MbHeightC) arrays $pred_{Cr}$ and $pred_{Cb}$ of chroma samples, one for each of the chroma components Cb and Cr.

In subclause 8.4.1 "Derivation process for motion vector components and reference indices", replace the paragraph starting with "Outputs of this process are" *with the following:*

Outputs of this process are

– luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1

Replace the paragraph starting with "For the derivation of the variables for the chroma motion vectors" with the following:

When ChromaArrayType is not equal to 0 and predFlagLX (with X being either 0 or 1) is equal to 1, the derivation process for chroma motion vectors in subclause 8.4.1.4 is invoked with mvLX and refIdxLX as input and the output being mvCLX.

In subclause 8.4.2.1 "Reference picture selection process", make the following changes.

Replace the paragraph starting with "Output of this process" *with the following:*

Output of this process is a reference picture consisting of a two-dimensional array of luma samples $refPicLX_L$ and when ChromaArrayType is not equal to 0, two two-dimensional arrays of chroma samples $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

Replace the paragraph starting with "The reference picture sample arrays" *with the following.*

Depending on separate_colour_plane_flag the following applies.

- If separate_colour_plane_flag is equal to 0, the reference picture sample arrays $refPicLX_L$, $refPicLX_{Cb}$ (if available), and $refPicLX_{Cr}$ (if available) correspond to decoded sample arrays S_L, S_{Cb} (if available), S_{Cr} (if available) derived in subclause 8.7 for a previously-decoded reference field or reference frame or complementary reference field pair or field of a reference frame.
- Otherwise (separate_colour_plane_flag is equal to 1), the following applies.
  - If colour_plane_id is equal to 0, the reference picture sample array $refPicLX_L$ corresponds to the decoded sample array S_L derived in subclause 8.7 for a previously-decoded reference field or reference frame or complementary reference field pair or field of a reference frame.
  - Otherwise, if colour_plane_id is equal to 1, the reference picture sample array $refPicLX_L$ corresponds to the decoded sample array S_{Cb} derived in subclause 8.7 for a previously-decoded reference field or reference frame or complementary reference field pair or field of a reference frame.
  - Otherwise (colour_plane_id is equal to 2), the reference picture sample array $refPicLX_L$ corresponds to the decoded sample array S_{Cb} derived in subclause 8.7 for a previously-decoded reference field or reference frame or complementary reference field pair or field of a reference frame.

In subclause 8.4.2.2 "Fractional sample interpolation process", make the following changes.

Replace the paragraphs:

- a chroma motion vector mvCLX given in eighth-chroma-sample units, and
- the selected reference picture sample arrays refPicLX$_L$, and refPicLX$_{Cb}$, and refPicLX$_{Cb}$

with:

- when ChromaArrayType is not equal to 0, a chroma motion vector mvCLX with a precision of (4*SubWidthC) chroma-sample units horizontally and (4*SubHeightC) chroma-sample units vertically, and
- the selected reference picture sample arrays refPicLX$_L$, and when ChromaArrayType is not equal to 0, refPicLX$_{Cb}$, and refPicLX$_{Cb}$

Replace the paragraph starting with "Let (xInt$_C$, yInt$_C$) be a chroma location" *with the following:*

Let (xInt$_C$, yInt$_C$) be a chroma location given in full-sample units and (xFrac$_C$, yFrac$_C$) be an offset given in (4*SubWidthC) chroma-sample units horizontally and (4*SubHeightC) chroma-sample units vertically. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays refPicLX$_{Cb}$, and refPicLX$_{Cr}$.

Replace the 4 equations following the paragraph staring with "Otherwise (chroma_format_idc is equal to 3)" *with the following:*

$$xIntC = xAC + (mvLX[0] >> 2) + xC \quad (8\text{-}232)$$
$$yIntL = yAC + (mvLX[1] >> 2) + yC \quad (8\text{-}233)$$

$$xFrac_C = mvLX[0] \,\&\, 3 \quad (8\text{-}234)$$
$$yFrac_C = mvLX[1] \,\&\, 3 \quad (8\text{-}235)$$

Replace the following paragraphs:

– The prediction sample value predPartLX$_{Cb}$[x$_C$, y$_C$] is derived by invoking the process specified in subclause 8.4.2.2.2 with (xInt$_C$, yInt$_C$), (xFrac$_C$, yFrac$_C$) and refPicLX$_{Cb}$ given as input.

– The prediction sample value predPartLX$_{Cr}$[x$_C$, y$_C$] is derived by invoking the process specified in subclause 8.4.2.2.2 with (xInt$_C$, yInt$_C$), (xFrac$_C$, yFrac$_C$) and refPicLX$_{Cr}$ given as input.

with:

– Depedending on ChromaArrayType the following applies.

– If ChromaArrayType is not equal to 3, the following applies.

- The prediction sample value predPartLX$_{Cb}$[x$_C$, y$_C$] is derived by invoking the process specified in subclause 8.4.2.2.2 with (xInt$_C$, yInt$_C$), (xFrac$_C$, yFrac$_C$) and refPicLX$_{Cb}$ given as input.
- The prediction sample value predPartLX$_{Cr}$[x$_C$, y$_C$] is derived by invoking the process specified in subclause 8.4.2.2.2 with (xInt$_C$, yInt$_C$), (xFrac$_C$, yFrac$_C$) and refPicLX$_{Cr}$ given as input.

– Otherwise (ChromaArrayType is equal to 3), the following applies.

  - The prediction sample value predPartLX$_{Cb}$[x_C, y_C] is derived by invoking the process specified in subclause 8.4.2.2.1 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and refPicLX$_{Cb}$ given as input.

  - The prediction sample value predPartLX$_{Cr}$[x_C, y_C] is derived by invoking the process specified in subclause 8.4.2.2.1 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and refPicLX$_{Cr}$ given as input.

In subclause 8.4.2.2.2 "Chroma sample interpolation process", replace the first paragraph, which states as follows:

This process shall only be invoked when chroma_format_idc is not equal to 0 (monochrome).

with:

This process shall only be invoked when ChromaArrayType is equal to 1 or 2.

Replace the content of subclause 8.5 "Transform coefficient decoding process and picture construction process prior to deblocking filter process" with the following:

Inputs to this process are Intra16x16DCLevel (if available), Intra16x16ACLevel (if available), CbIntra16x16DCLevel (if available), CbIntra16x16ACLevel (if available), CrIntra16x16DCLevel (if available), CrIntra16x16ACLevel (if available), LumaLevel (if available), LumaLevel8x8 (if available), ChromaDCLevel (if available), ChromaACLevel (if available), CbLevel (if available), CrLevel (if available), CbLevel8x8 (if available), CrLevel8x8 (if available), and available Inter or Intra prediction sample arrays for the current macroblock for the applicable components $pred_L$, $pred_{Cb}$, or $pred_{Cr}$

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock prediction array may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or Intra_8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated. For the same reason, the Cb and Cr component of the macroblock prediction array may not be complete when ChromaArrayType is equal to 3.

Outputs of this process are the constructed sample arrays prior to the deblocking filter process for the applicable components S'_L, S'_{Cb}, S'_{Cr}.

NOTE – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock constructed sample arrays prior to the deblocking filter process may not be complete, since for each 4x4 (or 8x8) luma block, the Intra_4x4 (or Intra_8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated. For the same reason, the Cb and Cr component of the macroblock constructed sample arrays prior to the deblocking filter process may not be complete when ChromaArrayType is equal to 3. .

This subclause specifies transform coefficient decoding and picture construction prior to the deblocking filter process.

When the current macroblock is coded as P_Skip or B_Skip, all values of LumaLevel, LumaLevel8x8, CbLevel, CbLevel8x8, CrLevel, CrLevel8x8, ChromaDCLevel, ChromaACLevel are set equal to 0 for the current macroblock.

*In subclause 8.5.1 "Specification of transform decoding process for 4x4 luma residual blocks", replace the paragraph starting with "*When residual_colour_transform_flag*" with the following:*

When qpprime_y_zero_transform_bypass_flag is equal to 1, QP'_Y is equal to 0, the macroblock prediction mode is equal to Intra_4x4, and Intra4x4PredMode[luma4x4BlkIdx] is equal to 0 or 1, the intra residual transform-bypass decoding process as specified in subclause 8.5.13 is invoked with nMax equal to 4, horPredFlag equal to Intra4x4PredMode[luma4x4BlkIdx], and the 4x4 array r as inputs, and the output is a modified version of the 4x4 array r.

*In subclause 8.5.2 "Specification of transform decoding process for luma samples of Intra_16x16 macroblock prediction mode", replace the paragraph starting with "*When residual_colour_transform_flag*" with the following:*

When qpprime_y_zero_transform_bypass_flag is equal to 1, QP'_Y is equal to 0, the macroblock prediction mode is equal to Intra_16x16, and Intra16x16PredMode is equal to 0 or 1, intra residual transform-bypass decoding process as specified in subclause 8.5.13 is invoked with nMax equal to 4, horPredFlag equal to Intra16x16PredMode, and the 4x4 array r as inputs, and the output is a modified version of the 4x4 array r.

*In subclause 8.5.3 "Specification of transform decoding process for 8x8 luma residual blocks", replace the paragraph starting with "*When residual_colour_transform_flag*" with the following:*

When qpprime_y_zero_transform_bypass_flag is equal to 1, QP'_Y is equal to 0, the macroblock prediction mode is equal to Intra_8x8, and Intra8x8PredMode[luma8x8BlkIdx] is equal to 0 or 1, the intra residual transform-bypass decoding process as specified in subclause 8.5.13 is invoked with nMax equal to 8, horPredFlag equal to Intra8x8PredMode[luma8x8BlkIdx], and the 8x8 array r as inputs, and the output is a modified version of the 8x8 array r.

In subclause 8.5.4 "Specification of transform decoding process for chroma samples", make the following changes.

Replace the first paragraph of the subclause with the following:

This process is invoked for each chroma component Cb and Cr separately when ChromaArrayType is not equal to 0.

Depending on ChromaArrayType, the following applies.

- If ChromaArrayType is equal to 3, the transform decoding process for chroma samples with ChromaArrayType equal to 3 as specified in subclause 8.5.4.1 is invoked.
- Otherwise (ChromaArrayType is not equal to 3), the following text specifies the transform decoding process for chroma samples.

Replace the sentence:

Otherwise, if chroma_format_idc is equal to 2, the 2x4 array c is derived using the inverse raster scanning process applied to ChromaDCLevel as follows

with:

Otherwise (ChromaArrayType is equal to 2), the 2x4 array c is derived using the inverse raster scanning process applied to ChromaDCLevel as follows

Delete the paragraph starting with "- Otherwise (chroma_format_idc is equal to 3),*".*

Delete the case (c) from Figure 8-7 including the corresponding part of the caption.

Replace the following:

Depending on the variable chroma_format_idc, the The position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx inside the macroblock is derived as follows.

– If chroma_format_idc is equal to 1 or 2, the following applies.

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8\text{-}302)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8\text{-}303)$$

– Otherwise (chroma_format_idc is equal to 3), the following applies.

$$xO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 0) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 0) \quad (8\text{-}304)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx / 4, 8, 8, 16, 1) + InverseRasterScan(chroma4x4BlkIdx \% 4, 4, 4, 8, 1) \quad (8\text{-}305)$$

with:

The position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx inside the macroblock is derived by.

$$xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) \quad (8\text{-}302)$$

$$yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) \quad (8\text{-}303)$$

Update all following equation numbers in clause 8 as necessary.

*Replace the paragraph starting with "*When residual_colour_transform_flag*" with the following:*

When qpprime_y_zero_transform_bypass_flag is equal to 1, QP'_Y is equal to 0, the macroblock prediction mode is equal to Intra_4x4, Intra_8x8, or Intra_16x16, and intra_chroma_pred_mode is equal to 1 or 2, the intra residual transform-bypass decoding process as specified in subclause 8.5.13 is invoked with nMax equal to 4, horPredFlag equal to (intra_chroma_pred_mode – 1), and the 4x4 array r as inputs, and the output is a modified version of the 4x4 array r.

Insert a new subclause 8.5.4.1 as follows:

8.5.4.1 Specification of transform decoding process for chroma samples with ChromaArrayType equal to 3

This process is invoked for each chroma component Cb and Cr separately when ChromaArrayType is equal to 3.

Depending on the macroblock prediction mode and transform_size_8x8_flag, the following applies.

– If the macroblock prediction mode is equal to Intra_16x16, the transform decoding process for Cb or Cr residual blocks shall be identical to the process described in subclause 8.5.2 when substituting luma with Cb or Cr, substituting Intra16x16DCLevel with CbIntra16x16DCLevel or CrIntra16x16DCLevel, substituting Intra16x16ACLevel with CbIntra16x16ACLevel or CrIntra16x16ACLevel, and substituting $pred_L$ with $pred_{Cb}$ or $pred_{Cr}$, substituting luma4x4BlkIdx with cb4x4BlkIdx or cr4x4BlkIdx, substituting lumaList with CbList or CrList, and substituting $Clip1_Y$ with $Clip1_C$.

– Otherwise, if transform_size_8x8_flag is equal to 1, the transform decoding process for 8x8 Cb or 8x8 Cr residual blocks shall be identical to the process described in subclause 8.5.3 when substituting luma with Cb or Cr, substituting LumaLevel8x8 with CbLevel8x8 or CrLevel8x8, substituting $pred_L$ with $pred_{Cb}$ or $pred_{Cr}$, substituting luma8x8BlkIdx with cb8x8BlkIdx or cr8x8BlkIdx, and substituting $Clip1_Y$ with $Clip1_C$.

– Otherwise (the macroblock prediction mode is not equal to Intra_16x16 and transform_size_8x8_flag is equal to 0), the transform decoding process for 4x4 Cb or 4x4 Cr residual blocks shall be identical to the process described in subclause 8.5.1 when substituting luma with Cb or Cr, substituting LumaLevel with CbLevel or CrLevel, substituting $pred_L$ with $pred_{Cb}$ or $pred_{Cr}$, substituting luma4x4BlkIdx with cb4x4BlkIdx or cr4x4BlkIdx, and substituting $Clip1_Y$ with $Clip1_C$.

Replace the title of subclause 8.5.6 with the following:

8.5.6 Inverse scanning process for 8x8 transform coefficients

In subclause 8.5.7 "Derivation process for the chroma quantisation parameters and scaling function", make the following changes.

Delete the following:

The value of $BitDepth'_C$ for the chroma components is derived as

$$BitDepth'_C = BitDepth_C + residual\_colour\_transform\_flag \quad (8\text{-}311)$$

Replace the following:

- The variable iYCbCr derived as follows.
  - If the input array c relates to a luma residual block, iYCbCr is set equal to 0.
  - Otherwise, if the input array c relates to a chroma residual block and the chroma component is equal to Cb, iYCbCr is set equal to 1.
  - Otherwise (the input array c relates to a chroma residual block and the chroma component is equal to Cr), iYCbCr is set equal to 2.

with:

- The variable iYCbCr derived as follows.
  - If separate_colour_plane_flag is equal to 1, iYCbCr is set equal to colour_plane_id.
  - Otherwise (separate_colour_plane_flag is equal to 0), the following applies.
    - If the input array c relates to a luma residual block, iYCbCr is set equal to 0.
    - Otherwise, if the input array c relates to a chroma residual block and the chroma component is equal to Cb, iYCbCr is set equal to 1.
    - Otherwise (the input array c relates to a chroma residual block and the chroma component is equal to Cr), iYCbCr is set equal to 2.

Replace the paragraph starting with "The inverse scanning process for 8x8 luma transform coefficients" *with the following:*

- The variable iYCbCr is derived as follows.
  - If separate_colour_plane_flag is equal to 1, iYCbCr is set equal to colour_plane_id.
  - Otherwise (separate_colour_plane_flag is equal to 0), the following applies.
    - If the input array c relates to a luma residual block, iYCbCr is set equal to 0.
    - Otherwise, if the input array c relates to a chroma residual block and the chroma component is equal to Cb, iYCbCr is set equal to 1.
    - Otherwise (the input array c relates to a chroma residual block and the chroma component is equal to Cr), iYCbCr is set equal to 2.
- The inverse scanning process for 8x8 luma transform coefficients as specified in subclause 8.5.6 is invoked with ScalingList8x8[2 * iYCbCr + mbIsInterFlag] as the input and the output is assigned to the 8x8 matrix weightScale8x8.

In subclause 8.5.8 "Scaling and transformation process for luma DC transform coefficients for Intra_16x16 macroblock type", make the following changes.

Change the title of the subclause to:

8.5.8 Scaling and transformation process for DC transform coefficients for Intra_16x16 macroblock type

Replace the following:

Inputs to this process are transform coefficient level values for luma DC transform coefficients of Intra_16x16 macroblocks as a 4x4 array c with elements c_{ij}, where i and j form a two-dimensional frequency index.

Outputs of this process are 16 scaled DC values for luma 4x4 blocks of Intra_16x16 macroblocks as a 4x4 array dcY with elements dcY_{ij}.

with:

Inputs to this process are transform coefficient level values for DC transform coefficients of Intra_16x16 macroblocks as a 4x4 array c with elements c_{ij}, where i and j form a two-dimensional frequency index.

Outputs of this process are 16 scaled DC values for 4x4 blocks of Intra_16x16 macroblocks as a 4x4 array dcY with elements dcY_{ij}.

The variables bitDepth and qP are derived as follows.

– If the input array c relates to a luma residual block, bitDepth is set equal to $BitDepth_Y$ and QP is set equal to QP'_Y.

– Otherwise (the input array c relates to a chroma residual block), bitDepth is set equal to $BitDepth_C$ and QP is set equal to QP'_C.

NOTE – When separate_colour_plane_flag is equal to 1, all residual blocks are considered to be associated with the luma component for purposes of the decoding process of each colour component of a picture.

Replace the following:

The bitstream shall not contain data that results in any element f_{ij} of f with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)}-1$, inclusive.

After the inverse transform, scaling is performed as follows.

– If qP is greater than or equal to 36, the scaled result is derived as

$$dcY_{ij} = (f_{ij} * LevelScale(QP'_Y \% 6, 0, 0)) << (QP'_Y / 6 - 6), \quad \text{with } i, j = 0...3. \tag{8-320}$$

– Otherwise (QP'_Y is less than 36), the scaled result is derived as

$$dcY_{ij} = (f_{ij} * LevelScale(QP'_Y \% 6, 0, 0) + (1 << (5 - QP'_Y / 6))) << (qP / 6 - 6), \text{ with } i, j = 0...3 \tag{8-321}$$

The bitstream shall not contain data that results in any element dcY_{ij} of dcY with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)}-1$, inclusive.

NOTE – When entropy_coding_mode_flag is equal to 0 and QP'_Y is less than 10 and profile_idc is equal to 66, 77, or 88, the range of values that can be represented for the elements c_{ij} of c is not sufficient to represent the full range of values of the elements dcY_{ij} of dcY that could be necessary to form a close approximation of the content of any possible source picture by use of the Intra_16x16 macroblock type.

with:

The bitstream shall not contain data that results in any element f_{ij} of f with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + bitDepth)}$ to $2^{(7 + bitDepth)}-1$, inclusive.

After the inverse transform, scaling is performed as follows.

– If qP is greater than or equal to 36, the scaled result is derived as

$$dcY_{ij} = (f_{ij} * LevelScale(qP \% 6, 0, 0)) << (qP / 6 - 6), \quad \text{with } i, j = 0...3. \tag{8-320}$$

– Otherwise (qP is less than 36), the scaled result is derived as

$$dcY_{ij} = (f_{ij} * LevelScale(qP \% 6, 0, 0) + (1 << (5 - qP / 6))) << (qP / 6 - 6), \quad \text{with } i, j = 0...3 \tag{8-321}$$

The bitstream shall not contain data that results in any element dcY_{ij} of dcY with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + bitDepth)}$ to $2^{(7 + bitDepth)}-1$, inclusive.

NOTE – When entropy_coding_mode_flag is equal to 0 and qP is less than 10 and profile_idc is equal to 66, 77, or 88, the range of values that can be represented for the elements c_{ij} of c is not sufficient to represent the full range of values of the elements dcY_{ij} of dcY that could be necessary to form a close approximation of the content of any possible source picture by use of the Intra_16x16 macroblock type.

In subclause 8.5.9 "Scaling and transformation process for chroma DC transform coefficients", make the following changes.

Add the following paragraph to the beginning of this subclause:

This process is only invoked when ChromaArrayType is equal to 1 or 2.

Replace the following:

– Otherwise, if chroma_format_idc is equal to 2, the inverse transform for the 2x4 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{8-324}$$

– Otherwise (chroma_format_idc is equal to 3), the inverse transform for the 4x4 chroma DC transform coefficients is specified as follows.

  – If residual_colour_transform_flag is equal to 1 and the current macroblock prediction mode MbPartPredMode(mb_type, 0) is Intra_4x4 or Intra_8x8, the inverse transform for the 4x4 chroma DC transform coefficients is specified as

$$f_{ij} = c_{ij} << 2 \quad \text{with } i, j = 0..3 \tag{8-325}$$

  – Otherwise, the inverse transform for the 4x4 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{8-326}$$

The bitstream shall not contain data that results in any element f_{ij} of f with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + BitDepth'_C)}$ to $2^{(7 + BitDepth'_C)}-1$, inclusive.

with:

– Otherwise (ChromaArrayType is equal to 2), the inverse transform for the 2x4 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8\text{-}324)$$

The bitstream shall not contain data that results in any element f_{ij} of f with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + bitDepth_C)}$ to $2^{(7 + bitDepth_C)}-1$, inclusive.

Update all following equation numbers in clause 8 as necessary.

Replace the following:

– If chroma_format_idc is equal to 2, the following applies.
  – The variable $QP'_{C,DC}$ is derived as

$$QP'_{C,DC} = QP'_C + 3 \quad (8\text{-}328)$$

  – Depending on the value of $QP'_{C,DC}$, the following applies.
    – If $QP'_{C,DC}$ is greater than or equal to 36, the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_{C,DC} \%6, 0, 0)) << (QP'_{C,DC} / 6 - 6), \quad \text{with} \quad i = 0..3, j = 0,1 \quad (8\text{-}329)$$

    – Otherwise ($QP'_{C,DC}$ is less than 36), the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_{C,DC} \% 6, 0, 0) + 2^{5-QP'_{C,DC}/6}) >> (6 - QP'_{C,DC} / 6), \text{ with } i = 0..3, j = 0,1 \quad (8\text{-}330)$$

– Otherwise (chroma_format_idc is equal to 3), the following applies.
  – If QP'_C is greater than or equal to 36, the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_C \%6, 0, 0)) << (QP'_C / 6 - 6), \quad \text{with} \quad i, j = 0..3. \quad (8\text{-}331)$$

  – Otherwise (QP'_C is less than 36), the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_C \% 6, 0, 0) + 2^{5-QP'_C/6}) >> (6 - QP'_C / 6), \text{ with } i, j = 0..3. \quad (8\text{-}332)$$

The bitstream shall not contain data that results in any element dcC_{ij} of dcC with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + BitDepth'_C)}$ to $2^{(7 + BitDepth'_C)}-1$, inclusive.

with:

– Otherwise (ChromaArrayType is equal to 2), the following applies.
  – The variable $QP'_{C,DC}$ is derived as

$$QP'_{C,DC} = QP'_C + 3 \quad (8\text{-}328)$$

– Depending on the value of $QP'_{C,DC}$, the following applies.
  – If $QP'_{C,DC}$ is greater than or equal to 36, the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_{C,DC} \% 6, 0, 0)) << (QP'_{C,DC} / 6 - 6), \quad \text{with} \quad i = 0..3, j = 0,1 \quad (8\text{-}329)$$

  – Otherwise ($QP'_{C,DC}$ is less than 36), the scaled result is derived as

$$dcC_{ij} = (f_{ij} * LevelScale(QP'_{C,DC} \% 6, 0, 0) + 2^{5 - QP'_{C,DC}/6}) >> (6 - QP'_{C,DC} / 6), \quad \text{with} \quad i = 0..3, j = 0,1 \quad (8\text{-}330)$$

The bitstream shall not contain data that results in any element dcC_{ij} of dcC with i, j = 0..3 that exceeds the range of integer values from $-2^{(7 + bitDepth_C)}$ to $2^{(7 + bitDepth_C)}-1$, inclusive.

Update all following equation numbers in clause 8 as necessary.

In subclause 8.5.10 "Scaling and transformation process for residual 4x4 blocks", replace the following:

– Otherwise (the input array c relates to a chroma residual block), bitDepth is set equal to $BitDepth'_C$.

with:

– Otherwise (the input array c relates to a chroma residual block), bitDepth is set equal to $BitDepth_C$.

In subclause 8.5.11 "Scaling and transformation process for residual 8x8 luma blocks", make the following changes.

Replace the title of the subclause with the following:

8.5.11 Scaling and transformation process for residual 8x8 blocks

Replace the following:

Input to this process is an 8x8 array c with elements c_{ij} which is an array relating to an 8x8 residual block of the luma component.

Outputs of this process are residual sample values as 8x8 array r with elements r_{ij}.

with:

Input to this process is an 8x8 array c with elements c_{ij} which is either an array relating to a residual block of the luma component or an array relating to a residual block of a chroma component when ChromaArrayType is equal to 3.

NOTE – When separate_colour_plane_flag is equal to 1, all residual blocks are considered to be associated with the luma component for purposes of the decoding process of each coded picture (prior to the final assignment of the decoded picture to a particular luma or chroma picture array according to the value of colour_plane_id).

Outputs of this process are residual sample values as 8x8 array r with elements r_{ij}.

The variables bitDepth and qP are derived as follows.

– If the input array c relates to a luma residual block, bitDepth is set equal to $BitDepth_Y$ and QP is set equal to QP'_Y.

– Otherwise (the input array c relates to a chroma residual block), bitDepth is set equal to $BitDepth_C$ and QP is set equal to QP'_C.

NOTE – When separate_colour_plane_flag is equal to 1, all residual blocks are considered to be associated with the luma component for purposes of the decoding process of each colour component of a picture.

Replace the following:

The bitstream shall not contain data that results in any element c_{ij} of c with i, j = 0..7 that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)}-1$, inclusive.

The scaling process for 8x8 block transform coefficient levels c_{ij} proceeds as follows.

– If QP'_Y is greater than or equal to 36, the scaled result is derived as

$$d_{ij} = (c_{ij} * LevelScale8x8(QP'_Y \% 6, i, j)) << (QP'_Y / 6 - 6), \quad \text{with} \quad i, j = 0..7. \tag{8-359}$$

– Otherwise (QP'_Y is less than 36), the scaled result is derived as

$$d_{ij} = (c_{ij} * LevelScale(QP'_Y \% 6, i, j) + 2^{5 - QP'_Y / 6}) >> (6 - QP'_Y / 6), \text{ with } i, j = 0..7. \tag{8-360}$$

The bitstream shall not contain data that results in any element d_{ij} of d with i, j = 0..7 that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)}-1$, inclusive.

with:

The bitstream shall not contain data that results in any element c_{ij} of c with i, j = 0..7 that exceeds the range of integer values from $-2^{(7 + BitDepth)}$ to $2^{(7 + BitDepth)}-1$, inclusive.

The scaling process for 8x8 block transform coefficient levels cij proceeds as follows.

– If qP is greater than or equal to 36, the scaled result shall be derived as

$$d_{ij} = (c_{ij} * LevelScale8x8(qP \% 6, i, j)) << (qP / 6 - 6), \quad \text{with} \quad i, j = 0..7. \tag{8-359}$$

– Otherwise (qP is less than 36), the scaled result shall be derived as

$$d_{ij} = (c_{ij} * LevelScale(qP \% 6, i, j) + 2^{5 - qP/6}) >> (6 - qP / 6), \text{ with } i, j = 0..7. \tag{8-360}$$

The bitstream shall not contain data that results in any element d_{ij} of d with i, j = 0..7 that exceeds the range of integer values from $-2^{(7 + BitDepth)}$ to $2^{(7 + BitDepth)}-1$, inclusive.

Replace the following paragraphs:

The bitstream shall not contain data that results in any element e_{ij}, f_{ij}, g_{ij}, h_{ij}, or k_{ij} for i and j in the range of 0..7, inclusive, that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)} - 1$, inclusive.

The bitstream shall not contain data that results in any element m_{ij} for i and j in the range of 0..7, inclusive, that exceeds the range of integer values from $-2^{(7 + BitDepth_Y)}$ to $2^{(7 + BitDepth_Y)} - 33$, inclusive.

with:

The bitstream shall not contain data that results in any element e_{ij}, f_{ij}, g_{ij}, h_{ij}, or k_{ij} for i and j in the range of 0..7, inclusive, that exceeds the range of integer values from $-2^{(7 + BitDepth)}$ to $2^{(7 + BitDepth)} - 1$, inclusive.

The bitstream shall not contain data that results in any element m_{ij} for i and j in the range of 0..7, inclusive, that exceeds the range of integer values from $-2^{(7 + BitDepth)}$ to $2^{(7 + BitDepth)} - 33$, inclusive.

In subclause 8.5.12 "Picture construction process prior to deblocking filter process", make the following changes.

Replace the first paragraph with the following:

Inputs to this process are

- luma4x4BlkIdx or chroma4x4BlkIdx or luma8x8BlkIdx or cb4x4BlkIdx or cr4x4BlkIdx or cb8x8BlkIdx or cr8x8BlkIdx
- a sample array u with elements u_{ij} which is either a 4x4 luma block or a 4x4 chroma block or an 8x8 luma block or an 8x8 chroma block when ChromaArrayType is equal to 3.

Replace the paragraph starting with "When u is a chroma block" *and all subparagraphs with the following:*

When u is a chroma block, for each sample uij of the chroma block, the following applies.

- The subscript C in the variable S'C is replaced with Cb for the Cb chroma component and with Cr for the Cr chroma component.
- Depending on the size of the block u, the following applies.
  - If u is an 4x4 luma block, the variable nE is set equal to 4 and depending on the variable ChromaArrayType, the position of the upper-left sample of a 4x4 chroma block inside the macroblock is derived as follows.
    - If ChromaArrayType is equal to 1 or 2, the following applies.

      xO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 0) (8-411)

      yO = InverseRasterScan(chroma4x4BlkIdx, 4, 4, 8, 1) (8-412)

    - Otherwise (ChromaArrayType is equal to 3), the position of the upper-left sample of the 4x4 Cb block with index cb4x4BlkIdx or the 4x4 Cr block with index cr4x4BlkIdx inside the macroblock is derived by invoking the inverse 4x4 Cb or Cr block scanning process in subclause 6.4.3.1 with cb4x4BlkIdx or cr4x4BlkIdx as the input and the output being assigned to (xO, yO).
  - Otherwise (u is an 8x8 Cb or Cr block when ChromaArrayType is equal to 3), the position of the upper-left sample of the 8x8 Cb block with index cb8x8BlkIdx or the Cr block with index cr8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 Cb or Cr block scanning process in subclause 6.4.4.1 with cb8x8BlkIdx or cr8x8BlkIdx as the input and the output being assigned to (xO, yO), and the variable nE is set equal to 8.
- Depending on the variable MbaffFrameFlag and the current macroblock, the following applies.
  - If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

    $$S'_C[(xP / SubWidthC) + xO + j, ((yP + SubHeightC - 1) / SubHeightC) + 2 * (yO + i)] = u_{ij} \quad \text{with } i, j = 0..nE - 1 \qquad (8\text{-}416)$$

  - Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

    $$S'_C[(xP/ SubWidthC) + xO + j, (yP / SubHeightC) + yO + i] = u_{ij} \text{ with } i, j = 0..nE - 1 \qquad (8\text{-}417)$$

Replace subclause 8.5.13 with the following:

8.5.13 Intra residual transform-bypass decoding process

This process is invoked when qpprime_y_zero_transform_bypass_flag is equal to 1, QP'_Y is equal to 0, the macroblock prediction mode is equal to Intra_4x4, Intra_8x8 or Intra_16x16, and the applicable intra prediction mode is equal to the vertical or horizontal mode. The process for the Cb and Cr components is applied in the same way as for the luma (L or Y) component.

Inputs to this process are

– a variable nMax

– a variable horPredFlag

– an (nMax)x(nMax) array r with elements $r_{i,j}$ which is either an array relating to a residual transform-bypass block of the luma component or an array relating to a residual transform-bypass block of the Cb and Cr component.

Output of this process is a modified version of the (nMax)x(nMax) array r with elements $r_{i,j}$ containing the result of the intra residual transform-bypass decoding process.

Let f be a temporary (nMax)x(nMax) array with elements f_{ij}, which are derived by

$$f_{ij} = r_{ij} \quad \text{with } i, j = 0..nMax - 1 \tag{8-417}$$

Depending on horPredFlag, the following applies.

– If horPredFlag is equal to 0, the modified array r is derived by:

$$r_{ij} = \sum_{k=0}^{i} f_{kj} \quad \text{with } i, j = 0..nMax - 1 \tag{8-417}$$

– Otherwise (horPredFlag is equal to 1), the modified array r is derived by

$$r_{ij} = \sum_{k=0}^{j} f_{ik} \quad \text{with } i, j = 0..nMax - 1 \tag{8-417}$$

Update all following equation numbers in clause 8 as necessary.

In subclause 8.7 "Deblocking filter process", make the following changes.

Replace the first paragraph, which states as follows.

A conditional filtering process is applied to all NxN (where N = 4 or N = 8 for luma, and N = 4 for chroma) block edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by disable_deblocking_filter_idc, as specified below. This filtering process is performed on a macroblock basis after the completion of the picture construction process prior to deblocking filter process (as specified in subclauses 8.5 and 8.6) for the entire decoded picture, with all macroblocks in a picture processed in order of increasing macroblock addresses.

with:

A conditional filtering process is specified in this subclause that is an integral part of the decoding process which shall be applied by decoders conforming to the Baseline, Extended, Main, High, High 10, High 4:2:2, and High 4:4:4 Predictive profiles. For decoders conforming to the High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra, and CAVLC 4:4:4 Intra profiles, the filtering process specified in this subclause, or one similar to it, should be applied but is not required.

The conditional filtering is applied to all NxN (where N = 4 or N = 8 for luma and for chroma when ChromaArrayType is equal to 3, and N = 4 for chroma when ChromaArrayType is equal to 1 or 2) block edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by disable_deblocking_filter_idc, as specified below. This filtering process is performed on a macroblock basis after the completion of the picture construction process prior to deblocking filter process (as specified in subclauses 8.5 and 8.6) for the entire decoded picture, with all macroblocks in a picture processed in order of increasing macroblock addresses.

Replace the following:

– Otherwise, if chroma_format_idc is equal to 3 (4:4:4 format), both types, the solid bold and dashed bold chroma edges are filtered.

with:

– Otherwise, if ChromaArrayType is equal to 3, depending on the transform_size_8x8_flag, the following applies.
  – If transform_size_8x8_flag is equal to 0, both types, the solid bold and dashed bold chroma edges are filtered.
  – Otherwise (transform_size_8x8_flag is equal to 1), only the solid bold chroma edges are filtered.

Replace the paragraph starting with "For the filtering of both chroma components" *with the following:*

When ChromaArrayType is not equal to 0, for the filtering of both chroma components with iCbCr = 0 for Cb and iCbCr = 1 for Cr, the following applies.

Replace the paragraph starting with "When filterInternalEdgesFlag is equal to 1, the filtering of the internal vertical chroma edge is specified as follows" *with the following*:

– When filterInternalEdgesFlag is equal to 1, the filtering of the internal vertical chroma edge is specified as follows.
  – When ChromaArrayType is not equal to 3 or when transform_size_8x8_flag is equal to 0, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and (xEk, yEk) = (4, k) with k = 0..MbHeightC – 1 as input and S'C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.
  – When ChromaArrayType is equal to 3, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and (xEk, yEk) = (8, k) with k = 0..MbHeightC – 1 as input and S'C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.
  – When ChromaArrayType is equal to 3 and when transform_size_8x8_flag is equal to 0, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, and (xEk, yEk) = (12, k) with k = 0..MbHeightC – 1 as input and S'C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.

Replace the paragraph starting with "When filterInternalEdgesFlag is equal to 1, the filtering of the internal horizontal chroma edge is specified as follows" *with the following:*

– When filterInternalEdgesFlag is equal to 1, the filtering of the internal horizontal chroma edge is specified as follows.
  – When ChromaArrayType is not equal to 3 or when transform_size_8x8_flag is equal to 0, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and (xE_k, yE_k) = (k, 4) with k = 0..MbWidthC – 1 as input and S'$_C$ with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.

– When ChromaArrayType is not equal to 1, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 8)$ with k = 0..MbWidthC – 1 as input and S'_C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.

– When ChromaArrayType is equal to 2, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 12)$ with k = 0..MbWidthC – 1 as input and S'_C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.

– When ChromaArrayType is equal to 3 and when transform_size_8x8_flag is equal to 0, the process specified in subclause 8.7.1 is invoked with mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, and $(xE_k, yE_k) = (k, 12)$ with k = 0..MbWidthC – 1 as input and S'_C with C being replaced by Cb for iCbCr = 0 and C being replaced by Cr for iCbCr = 1 as output.

Replace the paragraph that begins with "Finally, the arrays" *with the following:*

Depending on separate_colour_plane_flag the following applies.

- If separate_colour_plane_flag is equal to 0, the arrays S'_L, S'_{Cb}, S'_{Cr} are assigned to the arrays S_L, S_{Cb}, S_{Cr} (which represent the decoded picture), respectively.
- Otherwise (separate_colour_plane_flag is equal to 1), the following applies.
  - If colour_plane_id is equal to 0, the arrays S'_L is assigned to the array S_L (which represent the luma component of the decoded picture).
  - Otherwise, if colour_plane_id is equal to 1, the arrays S'_L is assigned to the array S_{Cb} (which represents the Cb component of the decoded picture).
  - Otherwise (colour_plane_id is equal to 2), the arrays S'_L is assigned to the array S_{Cr} (which represents the Cr component of the decoded picture).

In subclause 8.7.2.3 "Filtering process for edges with bS less than 4", make the following changes.

Replace the paragraph starting with "The filtered result samples p'_0 and q'_0 are derived by" *with the following:*

The filtered result samples p'_0 and q'_0 are derived by

$$\Delta = \text{Clip3}(-t_C, t_C, ((((q_0 - p_0) << 2) + (p_1 - q_1) + 4) >> 3)) \quad (8\text{-}469)$$

$$p'_0 = \text{Clip1}(p_0 + \Delta) \quad (8\text{-}470)$$

$$q'_0 = \text{Clip1}(q_0 - \Delta) \quad (8\text{-}471)$$

where the threshold t_C is determined as follows.

– If chromaEdgeFlag is equal to 0,

$$t_C = t_{C0} + ((a_p < \beta) \ ? \ 1 : 0) + ((a_q < \beta) \ ? \ 1 : 0) \quad (8\text{-}472)$$

– Otherwise (chromaEdgeFlag is equal to 1), the following applies.

  – When ChromaArrayType is equal to 3,

$$t_C = t_{C0} + ((a_p < \beta) ? 1 : 0) + ((a_q < \beta) ? 1 : 0) \quad (8\text{-}472a)$$

– Otherwise

$$t_C = t_{C0} + 1 \quad (8\text{-}473)$$

Update all following equation numbers in clause 8 as necessary.

*Replace the paragraph starting with "*The filtered result sample p'_1*" with the following:*

The filtered result sample p'_1 is derived as follows:

– If chromaEdgeFlag is equal to 0 and a_p is less than β,

$$p'_1 = p_1 + \text{Clip3}(-t_{C0}, t_{C0}, (p_2 + ((p_0 + q_0 + 1) >> 1) - (p_1 << 1)) >> 1) \quad (8\text{-}478)$$

– Otherwise, if chromaEdgeFlag is equal to 1, ChromaArrayType is equal to 3, and a_p is less than β,

$$p'_1 = p_1 + \text{Clip3}(-t_{C0}, t_{C0}, (p_2 + ((p_0 + q_0 + 1) >> 1) - (p_1 << 1)) >> 1) \quad (8\text{-}478a)$$

– Otherwise (chromaEdgeFlag is equal to 1 and ChromaArrayType is not equal to 3, or a_p is greater than or equal to β),

$$p'_1 = p_1 \quad (8\text{-}479)$$

*Replace the paragraph starting with "*The filtered result sample q'_1*" with the following:*

The filtered result sample q'_1 is derived as follows:

– If chromaEdgeFlag is equal to 0 and a_q is less than β,

$$q'_1 = q_1 + \text{Clip3}(-t_{C0}, t_{C0}, (q_2 + ((p_0 + q_0 + 1) >> 1) - (q_1 << 1)) >> 1) \quad (8\text{-}480)$$

– Otherwise, if chromaEdgeFlag is equal to 1, ChromaArrayType is equal to 3, and a_q is less than β,

$$q'_1 = q_1 + \text{Clip3}(-t_{C0}, t_{C0}, (q_2 + ((p_0 + q_0 + 1) >> 1) - (q_1 << 1)) >> 1) \quad (8\text{-}480a)$$

– Otherwise (chromaEdgeFlag is equal to 1 and ChromaArrayType is not equal to 3, or a_q is greater than or equal to β),

$$q'_1 = q_1 \quad (8\text{-}481)$$

Update all following equation numbers in clause 8 as necessary.

In subclause 8.7.2.4 "Filtering process for edges for bS equal to 4", make the following changes.

Replace the paragraph starting with "The filtered result samples p'_i (i = 0..2) are derived as follows" *with the following:*

The filtered result samples p'i (i = 0..2) are derived as follows.

– If chromaEdgeFlag is equal to 0 and the following condition holds,

$a_p < \beta$ && $Abs(p_0 - q_0) < ((\alpha >> 2) + 2)$ (8-484)

then the variables p'_0, p'_1, and p'_2 are derived by

$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3$ (8-485)

$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2$ (8-486)

$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3$ (8-487)

– Otherwise, if chromaEdgeFlag is equal to 1, ChromaArrayType is equal to 3, and the condition in Equation 8-484 holds, the variables p'0, p'1, and p'2 are derived by

$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3$ (8-485a)

$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2$ (8-486a)

$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3$ (8-487a)

– Otherwise (chromaEdgeFlag is equal to 1 and ChromaArrayType is not equal to 3, or the condition in Equation 8-484 does not hold), the variables p'0, p'1, and p'2 are derived by

$p'_0 = (2*p_1 + p_0 + q_1 + 2) >> 2$ (8-488)

$p'_1 = p_1$ (8-489)

$p'_2 = p_2$ (8-490)

Update all following equation numbers in clause 8 as necessary.

Replace the paragraph starting with "The filtered result samples q'_i (i = 0..2) are derived as follows" *with the following:*

The filtered result samples q'_i (i = 0..2) are derived as follows.

– If chromaEdgeFlag is equal to 0 and the following condition holds,

$a_q < \beta$ && $Abs(p_0 - q_0) < ((\alpha >> 2) + 2)$ (8-491)

then the variables q'_0, q'_1, and q'_2 are derived by

$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3$ (8-492)

$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2$ (8-493)

$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) >> 3$ (8-494)

– If chromaEdgeFlag is equal to 1, ChromaArrayType is equal to 3, and the condition in Equation 8-491 holds, the variables q'_0, q'_1, and q'_2 are derived by

$$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3 \quad (8\text{-}492a)$$

$$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2 \quad (8\text{-}493a)$$

$$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) >> 3 \quad (8\text{-}494a)$$

– Otherwise (chromaEdgeFlag is equal to 1 and ChromaArrayType is not equal to 3, or the condition in Equation 8-491 does not hold), the variables q'_0, q'_1, and q'_2 are derived by

$$q'_0 = (2*q_1 + q_0 + p_1 + 2) >> 2 \quad (8\text{-}495)$$

$$q'_1 = q_1 \quad (8\text{-}496)$$

$$q'_2 = q_2 \quad (8\text{-}497)$$

Update all following equation numbers in clause 8 as necessary.

In clause 9 and all subclauses of clause 9, make the following changes.

*Replace all occurrences of "*chroma_format_idc*" with "*ChromaArrayType*".*

Delete all occurrences of "(monochrome)*".*

In subclause 9.1.2 "Mapping process for coded block pattern", make the following changes.

Replace the title of the first part of Table 9-4 "(a) chroma_format_idc is not equal to 0*" with the following:*

(a) ChromaArrayType is equal to 1, or 2

Replace the title of the second part of Table 9-4 "(b) chroma_format_idc is equal to 0*" with the following:*

(b) ChromaArrayType is equal to 0, or 3

In subclause 9.2 "CAVLC parsing process for transform coefficient levels", make the following changes.

*Replace the paragraph starting with "*Inputs to this process*" with the following:*

Inputs to this process are bits from slice data, a maximum number of non-zero transform coefficient levels maxNumCoeff, the luma block index luma4x4BlkIdx or the chroma block index chroma4x4BlkIdx or index cb4x4BlkIdx or index cr4x4BlkIdx of the current block of transform coefficient levels.

*Replace the paragraph starting with "*Output of this process*" with the following:*

Output of this process is the list coeffLevel containing transform coefficient levels of the luma block with block index luma4x4BlkIdx or the chroma block with block index chroma4x4BlkIdx or index cb4x4BlkIdx or index cr4x4BlkIdx.

In subclause 9.2.1 "Parsing process for total number of transform coefficient levels and trailing ones", make the following changes.

Replace the paragraph starting with "Inputs to this process" *with the following:*

Inputs to this process are bits from slice data, a maximum number of non-zero transform coefficient levels maxNumCoeff, the luma block index luma4x4BlkIdx or the chroma block index chroma4x4BlkIdx or index cb4x4BlkIdx or index cr4x4BlkIdx of the current block of transform.

*Delete the paragraph "*Otherwise (chroma_format_idc is equal to 3), nC is set equal to 0".

Add the following two new paragraphs after the paragraph starting with "When the CAVLC parsing process is invoked for Intra16x16DCLevel*":*

When the CAVLC parsing process is invoked for CbIntra16x16DCLevel, cb4x4BlkIdx is set equal to 0.

When the CAVLC parsing process is invoked for CrIntra16x16DCLevel, cr4x4BlkIdx is set equal to 0.

*Add the following two new paragraphs after the paragraph starting with "*If the CAVLC parsing process is invoked for Intra16x16DCLevel*":*

Otherwise, if the CAVLC parsing process is invoked for CbIntra16x16DCLevel, CbIntra16x16ACLevel, or CbLevel, the process specified in subclause 6.4.8.4.1 is invoked with cb4x4BlkIdx as the input, and the output is assigned to mbAddrA, mbAddrB, cb4x4BlkIdxA, and cb4x4BlkIdxB. The 4x4 Cb block specified by mbAddrA\cb4x4BlkIdxA is assigned to blkA, and the 4x4 Cb block specified by mbAddrB\cb4x4BlkIdxB is assigned to blkB.

Otherwise, if the CAVLC parsing process is invoked for CrIntra16x16DCLevel, CrIntra16x16ACLevel, or CrLevel, the process specified in subclause 6.4.8.4.1 is invoked with cr4x4BlkIdx as the input, and the output is assigned to mbAddrA, mbAddrB, cr4x4BlkIdxA, and cr4x4BlkIdxB. The 4x4 Cr block specified by mbAddrA\cr4x4BlkIdxA is assigned to blkA, and the 4x4 Cr block specified by mbAddrB\cr4x4BlkIdxB is assigned to blkB.

*Replace the paragraph starting with "*NOTE - When parsing for Intra16x16DCLevel*," with the following:*

> NOTE – When parsing for Intra16x16DCLevel, CbIntra16x16DCLevel, or CrIntra16x16DCLevel, the values nA and nB are based on the number of non-zero transform coefficient levels in adjacent 4x4 blocks and not on the number of non-zero DC transform coefficient levels in adjacent 16x16 blocks.

In subclause 9.3.1.1 "Initialisation process for context variables", make the following changes.

Replace Table 9-11 with the following:

Table 9-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	Table	Slice type			
			SI	I	P, SP	B
slice_data( )	mb_skip_flag	Table 9-13 Table 9-14			11-13	24-26
	mb_field_decoding_flag	Table 9-18	70-72	70-72	70-72	70-72
macroblock_layer( )	mb_type	Table 9-12 Table 9-13 Table 9-14	0-10	3-10	14-20	27-35
	transform_size_8x8_flag	Table 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	Table 9-18	73-76	73-76	73-76	73-76

	coded_block_pattern (chroma)	Table 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Table 9-17	60-63	60-63	60-63	60-63
mb_pred( )	prev_intra4x4_pred_mode_flag	Table 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Table 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	Table 9-17	na	68	68	68
	rem_intra8x8_pred_mode	Table 9-17	na	69	69	69
	intra_chroma_pred_mode	Table 9-17	64-67	64-67	64-67	64-67
mb_pred( ) and sub_mb_pred( )	ref_idx_l0	Table 9-16			54-59	54-59
	ref_idx_l1	Table 9-16				54-59
	mvd_l0[ ][ ][0]	Table 9-15			40-46	40-46
	mvd_l1[ ][ ][0]	Table 9-15				40-46
	mvd_l0[ ][ ][1]	Table 9-15			47-53	47-53
	mvd_l1[ ][ ][1]	Table 9-15				47-53
sub_mb_pred( )	sub_mb_type	Table 9-13 Table 9-14			21-23	36-39
residual_block_cabac( )	coded_block_flag	Table 9-18 Table 9-24a Table 9-24i	85-104 460-483	85-104 460-483 1012-1023	85-104 460-483 1012-1023	85-104 460-483 1012-1023
	significant_coeff_flag[ ]	Table 9-19 Table 9-22 Table 9-24 Table 9-24 Table 9-24b Table 9-22f Table 9-24d Table 9-24e	105-165 277-337	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747	105-165 277-337 402-416 436-450 484-571 776-863 660-689 718-747
	last_significant_coeff_flag[ ]	Table 9-20 Table 9-23 Table 9-24 Table 9-24 Table 9-24c Table 9-24g Table 9-24d Table 9-24e	166-226 338-398	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765	166-226 338-398 417-425 451-459 572-659 864-951 690-707 748-765
	coeff_abs_level_minus1[ ]	Table 9-21 Table 9-24 Table 9-24h Table 9-24d Table 9-24e	227-275	227-275 426-435 952-1011 708-717 766-775	227-275 426-435 952-1011 708-717 766-775	227-275 426-435 952-1011 708-717 766-775

Insert a new Table 9-24a as follows:

Table 9-24a – Values of variables m and n for ctxIdx from 460 to 483

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
460	-17	123	-7	92	0	80	11	80	**472**	-17	123	-7	92	0	80	11	80
461	-12	115	-5	89	-5	89	5	76	**473**	-12	115	-5	89	-5	89	5	76
462	-16	122	-7	96	-7	94	2	84	**474**	-16	122	-7	96	-7	94	2	84
463	-11	115	-13	108	-4	92	5	78	**475**	-11	115	-13	108	-4	92	5	78
464	-12	63	-3	46	0	39	-6	55	**476**	-12	63	-3	46	0	39	-6	55
465	-2	68	-1	65	0	65	4	61	**477**	-2	68	-1	65	0	65	4	61
466	-15	84	-1	57	-15	84	-14	83	**478**	-15	84	-1	57	-15	84	-14	83
467	-13	104	-9	93	-35	127	-37	127	**479**	-13	104	-9	93	-35	127	-37	127
468	-3	70	-3	74	-2	73	-5	79	**480**	-3	70	-3	74	-2	73	-5	79
469	-8	93	-9	92	-12	104	-11	104	**481**	-8	93	-9	92	-12	104	-11	104
470	-10	90	-8	87	-9	91	-11	91	**482**	-10	90	-8	87	-9	91	-11	91
471	-30	127	-23	126	-31	127	-30	127	**483**	-30	127	-23	126	-31	127	-30	127

Insert a new Table 9-24b as follows:

Table 9-24b – Values of variables m and n for ctxIdx from 484 to 571

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
484	-7	93	-2	85	-13	103	-4	86	**528**	-7	93	-2	85	-13	103	-4	86
485	-11	87	-6	78	-13	91	-12	88	**529**	-11	87	-6	78	-13	91	-12	88
486	-3	77	-1	75	-9	89	-5	82	**530**	-3	77	-1	75	-9	89	-5	82
487	-5	71	-7	77	-14	92	-3	72	**531**	-5	71	-7	77	-14	92	-3	72
488	-4	63	2	54	-8	76	-4	67	**532**	-4	63	2	54	-8	76	-4	67
489	-4	68	5	50	-12	87	-8	72	**533**	-4	68	5	50	-12	87	-8	72
490	-12	84	-3	68	-23	110	-16	89	**534**	-12	84	-3	68	-23	110	-16	89
491	-7	62	1	50	-24	105	-9	69	**535**	-7	62	1	50	-24	105	-9	69
492	-7	65	6	42	-10	78	-1	59	**536**	-7	65	6	42	-10	78	-1	59
493	8	61	-4	81	-20	112	5	66	**537**	8	61	-4	81	-20	112	5	66
494	5	56	1	63	-17	99	4	57	**538**	5	56	1	63	-17	99	4	57
495	-2	66	-4	70	-78	127	-4	71	**539**	-2	66	-4	70	-78	127	-4	71
496	1	64	0	67	-70	127	-2	71	**540**	1	64	0	67	-70	127	-2	71
497	0	61	2	57	-50	127	2	58	**641**	0	61	2	57	-50	127	2	58
498	-2	78	-2	76	-46	127	-1	74	**542**	-2	78	-2	76	-46	127	-1	74
499	1	50	11	35	-4	66	-4	44	**543**	1	50	11	35	-4	66	-4	44
500	7	52	4	64	-5	78	-1	69	**544**	7	52	4	64	-5	78	-1	69
501	10	35	1	61	-4	71	0	62	**545**	10	35	1	61	-4	71	0	62
502	0	44	11	35	-8	72	-7	51	**546**	0	44	11	35	-8	72	-7	51
503	11	38	18	25	2	59	-4	47	**547**	11	38	18	25	2	59	-4	47
504	1	45	12	24	-1	55	-6	42	**548**	1	45	12	24	-1	55	-6	42
505	0	46	13	29	-7	70	-3	41	**549**	0	46	13	29	-7	70	-3	41
506	5	44	13	36	-6	75	-6	53	**550**	5	44	13	36	-6	75	-6	53
507	31	17	-10	93	-8	89	8	76	**551**	31	17	-10	93	-8	89	8	76
508	1	51	-7	73	-34	119	-9	78	**552**	1	51	-7	73	-34	119	-9	78
509	7	50	-2	73	-3	75	-11	83	**553**	7	50	-2	73	-3	75	-11	83
510	28	19	13	46	32	20	9	52	**554**	28	19	13	46	32	20	9	52
511	16	33	9	49	30	22	0	67	**555**	16	33	9	49	30	22	0	67
512	14	62	-7	100	-44	127	-5	90	**556**	14	62	-7	100	-44	127	-5	90

513	-13	108	9	53	0	54	1	67	**557**	-13	108	9	53	0	54	1	67
514	-15	100	2	53	-5	61	-15	72	**558**	-15	100	2	53	-5	61	-15	72
515	-13	101	5	53	0	58	-5	75	**559**	-13	101	5	53	0	58	-5	75
516	-13	91	-2	61	-1	60	-8	80	**560**	-13	91	-2	61	-1	60	-8	80
517	-12	94	0	56	-3	61	-21	83	**561**	-12	94	0	56	-3	61	-21	83
518	-10	88	0	56	-8	67	-21	64	**562**	-10	88	0	56	-8	67	-21	64
519	-16	84	-13	63	-25	84	-13	31	**563**	-16	84	-13	63	-25	84	-13	31
520	-10	86	-5	60	-14	74	-25	64	**564**	-10	86	-5	60	-14	74	-25	64
521	-7	83	-1	62	-5	65	-29	94	**565**	-7	83	-1	62	-5	65	-29	94
522	-13	87	4	57	5	52	9	75	**566**	-13	87	4	57	5	52	9	75
523	-19	94	-6	69	2	57	17	63	**567**	-19	94	-6	69	2	57	17	63
524	1	70	4	57	0	61	-8	74	**568**	1	70	4	57	0	61	-8	74
525	0	72	14	39	-9	69	-5	35	**569**	0	72	14	39	-9	69	-5	35
526	-5	74	4	51	-11	70	-2	27	**570**	-5	74	4	51	-11	70	-2	27
527	18	59	13	68	18	55	13	91	**571**	18	59	13	68	18	55	13	91

Insert a new Table 9-24c as follows:

Table 9-24c – Values of variables m and n for ctxIdx from 572 to 659

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
572	24	0	11	28	4	45	4	39	**616**	24	0	11	28	4	45	4	39
573	15	9	2	40	10	28	0	42	**617**	15	9	2	40	10	28	0	42
574	8	25	3	44	10	31	7	34	**618**	8	25	3	44	10	31	7	34
575	13	18	0	49	33	-11	11	29	**619**	13	18	0	49	33	-11	11	29
576	15	9	0	46	52	-43	8	31	**620**	15	9	0	46	52	-43	8	31
577	13	19	2	44	18	15	6	37	**621**	13	19	2	44	18	15	6	37
578	10	37	2	51	28	0	7	42	**622**	10	37	2	51	28	0	7	42
579	12	18	0	47	35	-22	3	40	**623**	12	18	0	47	35	-22	3	40
580	6	29	4	39	38	-25	8	33	**624**	6	29	4	39	38	-25	8	33
581	20	33	2	62	34	0	13	43	**625**	20	33	2	62	34	0	13	43
582	15	30	6	46	39	-18	13	36	**626**	15	30	6	46	39	-18	13	36
583	4	45	0	54	32	-12	4	47	**627**	4	45	0	54	32	-12	4	47
584	1	58	3	54	102	-94	3	55	**628**	1	58	3	54	102	-94	3	55

585	0	62	2	58	0	0	2	58	**629**	0	62	2	58	0	0	2	58
586	7	61	4	63	56	-15	6	60	**630**	7	61	4	63	56	-15	6	60
587	12	38	6	51	33	-4	8	44	**631**	12	38	6	51	33	-4	8	44
588	11	45	6	57	29	10	11	44	**632**	11	45	6	57	29	10	11	44
589	15	39	7	53	37	-5	14	42	**633**	15	39	7	53	37	-5	14	42
590	11	42	6	52	51	-29	7	48	**634**	11	42	6	52	51	-29	7	48
591	13	44	6	55	39	-9	4	56	**635**	13	44	6	55	39	-9	4	56
592	16	45	11	45	52	-34	4	52	**636**	16	45	11	45	52	-34	4	52
593	12	41	14	36	69	-58	13	37	**637**	12	41	14	36	69	-58	13	37
594	10	49	8	53	67	-63	9	49	**638**	10	49	8	53	67	-63	9	49
595	30	34	-1	82	44	-5	19	58	**639**	30	34	-1	82	44	-5	19	58
596	18	42	7	55	32	7	10	48	**640**	18	42	7	55	32	7	10	48
597	10	55	-3	78	55	-29	12	45	**641**	10	55	-3	78	55	-29	12	45
598	17	51	15	46	32	1	0	69	**642**	17	51	15	46	32	1	0	69
599	17	46	22	31	0	0	20	33	**643**	17	46	22	31	0	0	20	33
600	0	89	-1	84	27	36	8	63	**644**	0	89	-1	84	27	36	8	63
601	26	-19	25	7	33	-25	35	-18	**645**	26	-19	25	7	33	-25	35	-18
602	22	-17	30	-7	34	-30	33	-25	**646**	22	-17	30	-7	34	-30	33	-25
603	26	-17	28	3	36	-28	28	-3	**647**	26	-17	28	3	36	-28	28	-3
604	30	-25	28	4	38	-28	24	10	**648**	30	-25	28	4	38	-28	24	10
605	28	-20	32	0	38	-27	27	0	**649**	28	-20	32	0	38	-27	27	0
606	33	-23	34	-1	34	-18	34	-14	**650**	33	-23	34	-1	34	-18	34	-14
607	37	-27	30	6	35	-16	52	-44	**651**	37	-27	30	6	35	-16	52	-44
608	33	-23	30	6	34	-14	39	-24	**652**	33	-23	30	6	34	-14	39	-24
609	40	-28	32	9	32	-8	19	17	**653**	40	-28	32	9	32	-8	19	17
610	38	-17	31	19	37	-6	31	25	**654**	38	-17	31	19	37	-6	31	25
611	33	-11	26	27	35	0	36	29	**655**	33	-11	26	27	35	0	36	29
612	40	-15	26	30	30	10	24	33	**656**	40	-15	26	30	30	10	24	33
613	41	-6	37	20	28	18	34	15	**657**	41	-6	37	20	28	18	34	15
614	38	1	28	34	26	25	30	20	**658**	38	1	28	34	26	25	30	20
615	41	17	17	70	29	41	22	73	**659**	41	17	17	70	29	41	22	73

Insert a new Table 9-24d as follows:

Table 9-24d– Values of variables m and n for ctxIdx from 660 to 717

ctxIdx	I slices		Value of cabac_init_idc						ctxIdx	I slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
660	-17	120	-4	79	-5	85	-3	78	**689**	2	59	2	59	2	58	-11	68
661	-20	112	-7	71	-6	81	-8	74	**690**	23	-13	9	-2	17	-10	9	-2
662	-18	114	-5	69	-10	77	-9	72	**691**	26	-13	26	-9	32	-13	30	-10
663	-11	85	-9	70	-7	81	-10	72	**692**	40	-15	33	-9	42	-9	31	-4
664	-15	92	-8	66	-17	80	-18	75	**693**	49	-14	39	-7	49	-5	33	-1
665	-14	89	-10	68	-18	73	-12	71	**694**	44	3	41	-2	53	0	33	7
666	-26	71	-19	73	-4	74	-11	63	**695**	45	6	45	-3	64	3	31	12
667	-15	81	-12	69	-10	83	-5	70	**696**	44	34	49	9	68	10	37	23
668	-14	80	-16	70	-9	71	-17	75	**697**	33	54	45	27	66	27	31	38
669	0	68	-15	67	-9	67	-14	72	**698**	19	82	36	59	47	57	20	64
670	-14	70	-20	62	-1	61	-16	67	**699**	21	-10	21	-13	17	-10	9	-2
671	-24	56	-19	70	-8	66	-8	53	**700**	24	-11	33	-14	32	-13	30	-10
672	-23	68	-16	66	-14	66	-14	59	**701**	28	-8	39	-7	42	-9	31	-4
673	-24	50	-22	65	0	59	-9	52	**702**	28	-1	46	-2	49	-5	33	-1
674	-11	74	-20	63	2	59	-11	68	**703**	29	3	51	2	53	0	33	7
675	-14	106	-5	85	-3	81	-3	78	**704**	29	9	60	6	64	3	31	12
676	-13	97	-6	81	-3	76	-8	74	**705**	35	20	61	17	68	10	37	23
677	-15	90	-10	77	-7	72	-9	72	**706**	29	36	55	34	66	27	31	38
678	-12	90	-7	81	-6	78	-10	72	**707**	14	67	42	62	47	57	20	64
679	-18	88	-17	80	-12	72	-18	75	**708**	-3	75	-6	66	-5	71	-9	71
680	-10	73	-18	73	-14	68	-12	71	**709**	-1	23	-7	35	0	24	-7	37
681	-9	79	-4	74	-3	70	-11	63	**710**	1	34	-7	42	-1	36	-8	44
682	-14	86	-10	83	-6	76	-5	70	**711**	1	43	-8	45	-2	42	-11	49
683	-10	73	-9	71	-5	66	-17	75	**712**	0	54	-5	48	-2	52	-10	56
684	-10	70	-9	67	-5	62	-14	72	**713**	-2	55	-12	56	-9	57	-12	59
685	-10	69	-1	61	0	57	-16	67	**714**	0	61	-6	60	-6	63	-8	63
686	-5	66	-8	66	-4	61	-8	53	**715**	1	64	-5	62	-4	65	-9	67
687	-9	64	-14	66	-9	60	-14	59	**716**	0	68	-8	66	-4	67	-6	68
688	-5	58	0	59	1	54	-9	52	**717**	-9	92	-8	76	-7	82	-10	79

Insert a new Table 9-24e as follows.

Table 9-24e– Values of variables m and n for ctxIdx from 718 to 775

ctxIdx	I slices		Value of cabac_init_idc						ctxIdx	I slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
718	-17	120	-4	79	-5	85	-3	78	**747**	2	59	2	59	2	58	-11	68
719	-20	112	-7	71	-6	81	-8	74	**748**	23	-13	9	-2	17	-10	9	-2
720	-18	114	-5	69	-10	77	-9	72	**749**	26	-13	26	-9	32	-13	30	-10
721	-11	85	-9	70	-7	81	-10	72	**750**	40	-15	33	-9	42	-9	31	-4
722	-15	92	-8	66	-17	80	-18	75	**751**	49	-14	39	-7	49	-5	33	-1
723	-14	89	-10	68	-18	73	-12	71	**752**	44	3	41	-2	53	0	33	7
724	-26	71	-19	73	-4	74	-11	63	**753**	45	6	45	3	64	3	31	12
725	-15	81	-12	69	-10	83	-5	70	**754**	44	34	49	9	68	10	37	23
726	-14	80	-16	70	-9	71	-17	75	**755**	33	54	45	27	66	27	31	38
727	0	68	-15	67	-9	67	-14	72	**756**	19	82	36	59	47	57	20	64
728	-14	70	-20	62	-1	61	-16	67	**757**	21	-10	21	-13	17	-10	9	-2
729	-24	56	-19	70	-8	66	-8	53	**758**	24	-11	33	-14	32	-13	30	-10
730	-23	68	-16	66	-14	66	-14	59	**759**	28	-8	39	-7	42	-9	31	-4
731	-24	50	-22	65	0	59	-9	52	**760**	28	-1	46	-2	49	-5	33	-1
732	-11	74	-20	63	2	59	-11	68	**761**	29	3	51	2	53	0	33	7
733	-14	106	-5	85	-3	81	-3	78	**762**	29	9	60	6	64	3	31	12
734	-13	97	-6	81	-3	76	-8	74	**763**	35	20	61	17	68	10	37	23
735	-15	90	-10	77	-7	72	-9	72	**764**	29	36	55	34	66	27	31	38
736	-12	90	-7	81	-6	78	-10	72	**765**	14	67	42	62	47	57	20	64
737	-18	88	-17	80	-12	72	-18	75	**766**	-3	75	-6	66	-5	71	-9	71
738	-10	73	-18	73	-14	68	-12	71	**767**	-1	23	-7	35	0	24	-7	37
739	-9	79	-4	74	-3	70	-11	63	**768**	1	34	-7	42	-1	36	-8	44
740	-14	86	-10	83	-6	76	-5	70	**769**	1	43	-8	45	-2	42	-11	49
741	-10	73	-9	71	-5	66	-17	75	**770**	0	54	-5	48	-2	52	-10	56
742	-10	70	-9	67	-5	62	-14	72	**771**	-2	55	-12	56	-9	57	-12	59
743	-10	69	-1	61	0	57	-16	67	**772**	0	61	-6	60	-6	63	-8	63
744	-5	66	-8	66	-4	61	-8	53	**773**	1	64	-5	62	-4	65	-9	67
745	-9	64	-14	66	-9	60	-14	59	**774**	0	68	-8	66	-4	67	-6	68
746	-5	58	0	59	1	54	-9	52	**775**	-9	92	-8	76	-7	82	-10	79

Insert a new Table 9-24f as follows:

Table 9-24f – Values of variables m and n for ctxIdx from 776 to 863

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
776	-6	93	-13	106	-21	126	-22	127	**820**	-6	93	-13	106	-21	126	-22	127
777	-6	84	-16	106	-23	124	-25	127	**821**	-6	84	-16	106	-23	124	-25	127
778	-8	79	-10	87	-20	110	-25	120	**822**	-8	79	-10	87	-20	110	-25	120
779	0	66	-21	114	-26	126	-27	127	**823**	0	66	-21	114	-26	126	-27	127
780	-1	71	-18	110	-25	124	-19	114	**824**	-1	71	-18	110	-25	124	-19	114
781	0	62	-14	98	-17	105	-23	117	**825**	0	62	-14	98	-17	105	-23	117
782	-2	60	-22	110	-27	121	-25	118	**826**	-2	60	-22	110	-27	121	-25	118
783	-2	59	-21	106	-27	117	-26	117	**827**	-2	59	-21	106	-27	117	-26	117
784	-5	75	-18	103	-17	102	-24	113	**828**	-5	75	-18	103	-17	102	-24	113
785	-3	62	-21	107	-26	117	-28	118	**829**	-3	62	-21	107	-26	117	-28	118
786	-4	58	-23	108	-27	116	-31	120	**830**	-4	58	-23	108	-27	116	-31	120
787	-9	66	-26	112	-33	122	-37	124	**831**	-9	66	-26	112	-33	122	-37	124
788	-1	79	-10	96	-10	95	-10	94	**832**	-1	79	-10	96	-10	95	-10	94
789	0	71	-12	95	-14	100	-15	102	**833**	0	71	-12	95	-14	100	-15	102
790	3	68	-5	91	-8	95	-10	99	**834**	3	68	-5	91	-8	95	-10	99
791	10	44	-9	93	-17	111	-13	106	**835**	10	44	-9	93	-17	111	-13	106
792	-7	62	-22	94	-28	114	-50	127	**836**	-7	62	-22	94	-28	114	-50	127
793	15	36	-5	86	-6	89	-5	92	**837**	15	36	-5	86	-6	89	-5	92
794	14	40	9	67	-2	80	17	57	**838**	14	40	9	67	-2	80	17	57
795	16	27	-4	80	-4	82	-5	86	**839**	16	27	-4	80	-4	82	-5	86
796	12	29	-10	85	-9	85	-13	94	**840**	12	29	-10	85	-9	85	-13	94
797	1	44	-1	70	-8	81	-12	91	**841**	1	44	-1	70	-8	81	-12	91
798	20	36	7	60	-1	72	-2	77	**842**	20	36	7	60	-1	72	-2	77
799	18	32	9	58	5	64	0	71	**843**	18	32	9	58	5	64	0	71
800	5	42	5	61	1	67	-1	73	**844**	5	42	5	61	1	67	-1	73
801	1	48	12	50	9	56	4	64	**845**	1	48	12	50	9	56	4	64
802	10	62	15	50	0	69	-7	81	**846**	10	62	15	50	0	69	-7	81
803	17	46	18	49	1	69	5	64	**847**	17	46	18	49	1	69	5	64
804	9	64	17	54	7	69	15	57	**848**	9	64	17	54	7	69	15	57

805	-12	104	10	41	-7	69	1	67	**849**	-12	104	10	41	-7	69	1	67
806	-11	97	7	46	-6	67	0	68	**850**	-11	97	7	46	-6	67	0	68
807	-16	96	-1	51	-16	77	-10	67	**851**	-16	96	-1	51	-16	77	-10	67
808	-7	88	7	49	-2	64	1	68	**852**	-7	88	7	49	-2	64	1	68
809	-8	85	8	52	2	61	0	77	**853**	-8	85	8	52	2	61	0	77
810	-7	85	9	41	-6	67	2	64	**854**	-7	85	9	41	-6	67	2	64
811	-9	85	6	47	-3	64	0	68	**855**	-9	85	6	47	-3	64	0	68
812	-13	88	2	55	2	57	-5	78	**856**	-13	88	2	55	2	57	-5	78
813	4	66	13	41	-3	65	7	55	**857**	4	66	13	41	-3	65	7	55
814	-3	77	10	44	-3	66	5	59	**858**	-3	77	10	44	-3	66	5	59
815	-3	76	6	50	0	62	2	65	**859**	-3	76	6	50	0	62	2	65
816	-6	76	5	53	9	51	14	54	**860**	-6	76	5	53	9	51	14	54
817	10	58	13	49	-1	66	15	44	**861**	10	58	13	49	-1	66	15	44
818	-1	76	4	63	-2	71	5	60	**862**	-1	76	4	63	-2	71	5	60
819	-1	83	6	64	-2	75	2	70	**863**	-1	83	6	64	-2	75	2	70

Insert a new Table 9-24g as follows:

Table 9-24g – Values of variables m and n for ctxIdx from 864 to 951

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
864	15	6	14	11	19	-6	17	-13	**908**	15	6	14	11	19	-6	17	-13
865	6	19	11	14	18	-6	16	-9	**909**	6	19	11	14	18	-6	16	-9
866	7	16	9	11	14	0	17	-12	**910**	7	16	9	11	14	0	17	-12
867	12	14	18	11	26	-12	27	-21	**911**	12	14	18	11	26	-12	27	-21
868	18	13	21	9	31	-16	37	-30	**912**	18	13	21	9	31	-16	37	-30
869	13	11	23	-2	33	-25	41	-40	**913**	13	11	23	-2	33	-25	41	-40
870	13	15	32	-15	33	-22	42	-41	**914**	13	15	32	-15	33	-22	42	-41
871	15	16	32	-15	37	-28	48	-47	**915**	15	16	32	-15	37	-28	48	-47
872	12	23	34	-21	39	-30	39	-32	**916**	12	23	34	-21	39	-30	39	-32
873	13	23	39	-23	42	-30	46	-40	**917**	13	23	39	-23	42	-30	46	-40
874	15	20	42	-33	47	-42	52	-51	**918**	15	20	42	-33	47	-42	52	-51
875	14	26	41	-31	45	-36	46	-41	**919**	14	26	41	-31	45	-36	46	-41
876	14	44	46	-28	49	-34	52	-39	**920**	14	44	46	-28	49	-34	52	-39

877	17	40	38	-12	41	-17	43	-19	**921**	17	40	38	-12	41	-17	43	-19
878	17	47	21	29	32	9	32	11	**922**	17	47	21	29	32	9	32	11
879	24	17	45	-24	69	-71	61	-55	**923**	24	17	45	-24	69	-71	61	-55
880	21	21	53	-45	63	-63	56	-46	**924**	21	21	53	-45	63	-63	56	-46
881	25	22	48	-26	66	-64	62	-50	**925**	25	22	48	-26	66	-64	62	-50
882	31	27	65	-43	77	-74	81	-67	**926**	31	27	65	-43	77	-74	81	-67
883	22	29	43	-19	54	-39	45	-20	**927**	22	29	43	-19	54	-39	45	-20
884	19	35	39	-10	52	-35	35	-2	**928**	19	35	39	-10	52	-35	35	-2
885	14	50	30	9	41	-10	28	15	**929**	14	50	30	9	41	-10	28	15
886	10	57	18	26	36	0	34	1	**930**	10	57	18	26	36	0	34	1
887	7	63	20	27	40	-1	39	1	**931**	7	63	20	27	40	-1	39	1
888	-2	77	0	57	30	14	30	17	**932**	-2	77	0	57	30	14	30	17
889	-4	82	-14	82	28	26	20	38	**933**	-4	82	-14	82	28	26	20	38
890	-3	94	-5	75	23	37	18	45	**934**	-3	94	-5	75	23	37	18	45
891	9	69	-19	97	12	55	15	54	**935**	9	69	-19	97	12	55	15	54
892	-12	109	-35	125	11	65	0	79	**936**	-12	109	-35	125	11	65	0	79
893	36	-35	27	0	37	-33	36	-16	**937**	36	-35	27	0	37	-33	36	-16
894	36	-34	28	0	39	-36	37	-14	**938**	36	-34	28	0	39	-36	37	-14
895	32	-26	31	-4	40	-37	37	-17	**939**	32	-26	31	-4	40	-37	37	-17
896	37	-30	27	6	38	-30	32	1	**940**	37	-30	27	6	38	-30	32	1
897	44	-32	34	8	46	-33	34	15	**941**	44	-32	34	8	46	-33	34	15
898	34	-18	30	10	42	-30	29	15	**942**	34	-18	30	10	42	-30	29	15
899	34	-15	24	22	40	-24	24	25	**943**	34	-15	24	22	40	-24	24	25
900	40	-15	33	19	49	-29	34	22	**944**	40	-15	33	19	49	-29	34	22
901	33	-7	22	32	38	-12	31	16	**945**	33	-7	22	32	38	-12	31	16
902	35	-5	26	31	40	-10	35	18	**946**	35	-5	26	31	40	-10	35	18
903	33	0	21	41	38	-3	31	28	**947**	33	0	21	41	38	-3	31	28
904	38	2	26	44	46	-5	33	41	**948**	38	2	26	44	46	-5	33	41
905	33	13	23	47	31	20	36	28	**949**	33	13	23	47	31	20	36	28
906	23	35	16	65	29	30	27	47	**950**	23	35	16	65	29	30	27	47
907	13	58	14	71	25	44	21	62	**951**	13	58	14	71	25	44	21	62

Insert a new Table 9-24h as follows:

Table 9-24h – Values of variables m and n for ctxIdx from 952 to 1011

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
952	-3	71	-6	76	-23	112	-24	115	**982**	-3	71	-6	76	-23	112	-24	115
953	-6	42	-2	44	-15	71	-22	82	**983**	-6	42	-2	44	-15	71	-22	82
954	-5	50	0	45	-7	61	-9	62	**984**	-5	50	0	45	-7	61	-9	62
955	-3	54	0	52	0	53	0	53	**985**	-3	54	0	52	0	53	0	53
956	-2	62	-3	64	-5	66	0	59	**986**	-2	62	-3	64	-5	66	0	59
957	0	58	-2	59	-11	77	-14	85	**987**	0	58	-2	59	-11	77	-14	85
958	1	63	-4	70	-9	80	-13	89	**988**	1	63	-4	70	-9	80	-13	89
959	-2	72	-4	75	-9	84	-13	94	**989**	-2	72	-4	75	-9	84	-13	94
960	-1	74	-8	82	-10	87	-11	92	**990**	-1	74	-8	82	-10	87	-11	92
961	-9	91	-17	102	-34	127	-29	127	**991**	-9	91	-17	102	-34	127	-29	127
962	-5	67	-9	77	-21	101	-21	100	**992**	-5	67	-9	77	-21	101	-21	100
963	-5	27	3	24	-3	39	-14	57	**993**	-5	27	3	24	-3	39	-14	57
964	-3	39	0	42	-5	53	-12	67	**994**	-3	39	0	42	-5	53	-12	67
965	-2	44	0	48	-7	61	-11	71	**995**	-2	44	0	48	-7	61	-11	71
966	0	46	0	55	-11	75	-10	77	**996**	0	46	0	55	-11	75	-10	77
967	-16	64	-6	59	-15	77	-21	85	**997**	-16	64	-6	59	-15	77	-21	85
968	-8	68	-7	71	-17	91	-16	88	**998**	-8	68	-7	71	-17	91	-16	88
969	-10	78	-12	83	-25	107	-23	104	**999**	-10	78	-12	83	-25	107	-23	104
970	-6	77	-11	87	-25	111	-15	98	**1000**	-6	77	-11	87	-25	111	-15	98
971	-10	86	-30	119	-28	122	-37	127	**1001**	-10	86	-30	119	-28	122	-37	127
972	-12	92	1	58	-11	76	-10	82	**1002**	-12	92	1	58	-11	76	-10	82
973	-15	55	-3	29	-10	44	-8	48	**1003**	-15	55	-3	29	-10	44	-8	48
974	-10	60	-1	36	-10	52	-8	61	**1004**	-10	60	-1	36	-10	52	-8	61
975	-6	62	1	38	-10	57	-8	66	**1005**	-6	62	1	38	-10	57	-8	66
976	-4	65	2	43	-9	58	-7	70	**1006**	-4	65	2	43	-9	58	-7	70

977	-12	73	-6	55	-16	72	-14	75	**1007**	-12	73	-6	55	-16	72	-14	75
978	-8	76	0	58	-7	69	-10	79	**1008**	-8	76	0	58	-7	69	-10	79
979	-7	80	0	64	-4	69	-9	83	**1009**	-7	80	0	64	-4	69	-9	83
980	-9	88	-3	74	-5	74	-12	92	**1010**	-9	88	-3	74	-5	74	-12	92
981	-17	110	-10	90	-9	86	-18	108	**1011**	-17	110	-10	90	-9	86	-18	108

Insert a new Table 9-24i as follows:

Table 9-24i – Values of variables m and n for ctxIdx from 1012 to 1023

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	N	m	n
1012	-3	70	-3	74	-2	73	-5	79	**1018**	-10	90	-8	87	-9	91	-11	91
1013	-8	93	-9	92	-12	104	-11	104	**1019**	-30	127	-23	126	-31	127	-30	127
1014	-10	90	-8	87	-9	91	-11	91	**1020**	-3	70	-3	74	-2	73	-5	79
1015	-30	127	-23	126	-31	127	-30	127	**1021**	-8	93	-9	92	-12	104	-11	104
1016	-3	70	-3	74	-2	73	-5	79	**1022**	-10	90	-8	87	-9	91	-11	91
1017	-8	93	-9	92	-12	104	-11	104	**1023**	-30	127	-23	126	-31	127	-30	127

In subclause 9.3.2 "Binarization process", make the following changes.

*Replace the paragraph starting with "*The possible values of the context index ctxIdx*" with the following.*

The possible values of the context index ctxIdx are in the range 0 to 1023, inclusive. The value assigned to ctxIdxOffset specifies the lower value of the range of ctxIdx assigned to the corresponding binarization or binarization part of a syntax element.

Replace Table 9-25 with the following:

Table 9-25 – Syntax elements and associated types of binarization, maxBinIdxCtx, and ctxIdxOffset

Syntax element	Type of binarization	maxBinIdxCtx	ctxIdxOffset
mb_type (SI slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 0 suffix: 6	prefix: 0 suffix: 3
mb_type (I slices only)	as specified in subclause 9.3.2.5	6	3
mb_skip_flag (P, SP slices only)	FL, cMax=1	0	11
mb_type (P, SP slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 2 suffix: 5	prefix: 14 suffix: 17
sub_mb_type (P, SP slices only)	as specified in subclause 9.3.2.5	2	21
mb_skip_flag (B slices only)	FL, cMax=1	0	24
mb_type (B slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 3 suffix: 5	prefix: 27 suffix: 32
sub_mb_type (B slices only)	as specified in subclause 9.3.2.5	3	36
mvd_l0[ ][ ][0], mvd_l1[ ][ ][0]	prefix and suffix as given by UEG3 with signedValFlag=1, uCoff=9	prefix: 4 suffix: na	prefix: 40 suffix: na (uses DecodeBypass)
mvd_l0[ ][ ][1], mvd_l1[ ][ ][1]		prefix: 4 suffix: na	prefix: 47 suffix: na (uses DecodeBypass)
ref_idx_l0, ref_idx_l1	U	2	54
mb_qp_delta	as specified in subclause 9.3.2.7	2	60
intra_chroma_pred_mode	TU, cMax=3	1	64
prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag	FL, cMax=1	0	68
rem_intra4x4_pred_mode, rem_intra8x8_pred_mode	FL, cMax=7	0	69
mb_field_decoding_flag	FL, cMax=1	0	70
coded_block_pattern	prefix and suffix as specified in subclause 9.3.2.6	prefix: 3 suffix: 1	prefix: 73 suffix: 77
coded_block_flag (ctxBlockCat < 5)	FL, cMax=1	0	85
significant_coeff_flag (frame coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	105
last_significant_coeff_flag (frame coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	166
coeff_abs_level_minus1 (blocks with ctxBlockCat < 5)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 227 suffix: na, (uses DecodeBypass)
coeff_sign_flag	FL, cMax=1	0	na, (uses DecodeBypass)
end_of_slice_flag	FL, cMax=1	0	276

significant_coeff_flag (field coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	277
last_significant_coeff_flag (field coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	338
transform_size_8x8_flag	FL, cMax=1	0	399
significant_coeff_flag (frame coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	402
last_significant_coeff_flag (frame coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	417
coeff_abs_level_minus1 (blocks with ctxBlockCat == 5)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 426 suffix: na, (uses DecodeBypass)
significant_coeff_flag (field coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	436
last_significant_coeff_flag (field coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	451
coded_block_flag (5 < ctxBlockCat < 9)	FL, cMax=1	0	460
coded_block_flag (9 < ctxBlockCat < 13)	FL, cMax=1	0	472
coded_block_flag (ctxBlockCat == 5, 9, or 13)	FL, cMax=1	0	1012
significant_coeff_flag (frame coded blocks with 5<ctxBlockCat<9)	FL, cMax=1	0	484
significant_coeff_flag (frame coded blocks with 9<ctxBlockCat<13)	FL, cMax=1	0	528
last_significant_coeff_flag (frame coded blocks with 5<ctxBlockCat<9)	FL, cMax=1	0	572
last_significant_coeff_flag (frame coded blocks with 9<ctxBlockCat<13)	FL, cMax=1	0	616
coeff_abs_level_minus1 (blocks with 5<ctxBlockCat<9)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 952 suffix: na, (uses DecodeBypass)
coeff_abs_level_minus1 (blocks with 9<ctxBlockCat<13)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 982 suffix: na, (uses DecodeBypass)
significant_coeff_flag (field coded blocks with 5<ctxBlockCat< 9)	FL, cMax=1	0	776
significant_coeff_flag (field coded blocks with 9<ctxBlockCat<13)	FL, cMax=1	0	820
last_significant_coeff_flag (field coded blocks with 5<ctxBlockCat<9)	FL, cMax=1	0	864
last_significant_coeff_flag (field coded blocks with 9<ctxBlockCat<13)	FL, cMax=1	0	908
significant_coeff_flag (frame coded blocks with ctxBlockCat == 9)	FL, cMax=1	0	660

significant_coeff_flag (frame coded blocks with ctxBlockCat == 13)	FL, cMax=1	0	718
last_significant_coeff_flag (frame coded blocks with ctxBlockCat == 9)	FL, cMax=1	0	690
last_significant_coeff_flag (frame coded blocks with ctxBlockCat == 13)	FL, cMax=1	0	748
coeff_abs_level_minus1 (blocks with ctxBlockCat == 9)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 708 suffix: na, (uses DecodeBypass)
coeff_abs_level_minus1 (blocks with ctxBlockCat == 13)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 766 suffix: na, (uses DecodeBypass)
significant_coeff_flag (field coded blocks with ctxBlockCat == 9)	FL, cMax=1	0	675
significant_coeff_flag (field coded blocks with ctxBlockCat == 13)	FL, cMax=1	0	733
last_significant_coeff_flag (field coded blocks with ctxBlockCat == 9)	FL, cMax=1	0	699
last_significant_coeff_flag (field coded blocks with ctxBlockCat == 13)	FL, cMax=1	0	757

In subclause 9.3.2.6 "Binarization process for coded block pattern", replace the paragraph starting with "The binarization of coded_block_pattern consists of…" *with the following:*

The binarization of coded_block_pattern consists of a prefix part and (when present) a suffix part. The prefix part of the binarization is given by the FL binarization of CodedBlockPatternLuma with cMax = 15. When ChromaArrayType is not equal to 0 or 3, the suffix part is present and consists of the TU binarization of CodedBlockPatternChroma with cMax = 2. The relationship between the value of the syntax element coded_block_pattern and the values of CodedBlockPatternLuma and CodedBlockPatternChroma is given as specified in subclause 7.4.5.

In subclause 9.3.3.1 "Derivation process for ctxIdx", replace Table 9-31 with the following:

Table 9-31 – Assignment of ctxIdxBlockCatOffset to ctxBlockCat for syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1

Syntax element	ctxBlockCat (as specified in Table 9-33)													
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
coded_block_flag	0	4	8	12	16	0	0	4	8	4	0	4	8	8
significant_coeff_flag	0	15	29	44	47	0	0	15	29	0	0	15	29	0
last_significant_coeff_flag	0	15	29	44	47	0	0	15	29	0	0	15	29	0
coeff_abs_level_minus1	0	10	20	30	39	0	0	10	20	0	0	10	20	0

In subclause 9.3.3.1.1.9 "Derivation process of ctxIdxInc for the syntax element coded_block_flag", make the following changes.

Replace the first paragraph with the following:

Input to this process is ctxBlockCat and additional input is specified as follows.

- If ctxBlockCat is equal to 0, 6, or 10, no additional input
- Otherwise, if ctxBlockCat is equal to 1 or 2, luma4x4BlkIdx
- Otherwise, if ctxBlockCat is equal to 3, the chroma component index iCbCr
- Otherwise, if ctxBlockCat is equal to 4, chroma4x4BlkIdx and the chroma component index iCbCr
- Otherwise, if ctxBlockCat is equal to 5, luma8x8BlkIdx
- Otherwise, if ctxBlockCat is equal to 7 or 8, cb4x4BlkIdx
- Otherwise, if ctxBlockCat is equal to 9, cb8x8BlkIdx
- Otherwise, if ctxBlockCat is equal to 11 or 12, cr4x4BlkIdx
- Otherwise, (ctxBlockCat is equal to 13), cr8x8BlkIdx

*Replace the paragraph starting with "*If ctxBlockCat is equal to 0, the following applies.*" with the following:*

- If ctxBlockCat is equal to 0, 6, or 10, the following applies.
  - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available and the macroblock mbAddrN is coded in Intra_16x16 prediction mode, the following applies.
      - When ctxBlockCat is equal to 0, the luma DC block of macroblock mbAddrN is assigned to transBlockN
      - When ctxBlockCat is equal to 6, the Cb DC block of macroblock mbAddrN is assigned to transBlockN
      - When ctxBlockCat is equal to 10, the Cr DC block of macroblock mbAddrN is assigned to transBlockN
    - Otherwise, transBlockN is marked as not available.

*Replace the sentence "*Otherwise (ctxBlockCat is equal to 4), the following applies*" with the following:*

Otherwise, if ctxBlockCat is equal to 4, the following applies

*Insert the following paragraphs before the paragraph starting with "*Let the variable condTermFlagN (with N being either A or B) be derived as follows*":*

- Otherwise, if ctxBlockCat is equal to 5, the following applies.
  - The derivation process for neighbouring 8x8 luma blocks specified in subclause 6.4.8.2 is invoked with luma8x8BlkIdx as input and the output is assigned to mbAddrN, luma8x8BlkIdxN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, ((CodedBlockPatternLuma >>luma8x8BlkIdx) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 1 for the macroblock mbAddrN, the 8x8 luma block with index luma8x8BlkIdxN of macroblock mbAddrN is assigned to transBlockN.

- Otherwise, transBlockN is marked as not available.

- Otherwise, if ctxBlockCat is equal to7 or 8, the following applies.
  - The derivation process for neighbouring 4x4 Cb blocks specified in subclause 6.4.8.4.1 is invoked with cb4x4BlkIdx as input and the output is assigned to mbAddrN, cb4x4BlkIdxN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, ((CodedBlockPatternLuma >> (cb4x4BlkIdxN >>2)) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 0 for the macroblock mbAddrN, the 4x4 Cb block with index cb4x4BlkIdxN of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, if mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip or B_Skip, ((CodedBlockPatternLuma >> (cb4x4BlkIdxN >>2)) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 1 for the macroblock mbAddrN, the 8x8 Cb block with index (cb4x4BlkIdxN >> 2) of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, transBlockN is marked as not available.
- Otherwise, if ctxBlockCat is equal to 9, the following applies.
  - The derivation process for neighbouring 8x8 Cb blocks specified in subclause 6.4.8.2.1 is invoked with cb8x8BlkIdx as input and the output is assigned to mbAddrN, cb8x8BlkIdxN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, ((CodedBlockPatternLuma >>cb8x8BlkIdx) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 1 for the macroblock mbAddrN, the 8x8 Cb block with index cb8x8BlkIdxN of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, transBlockN is marked as not available.
- Otherwise, if ctxBlockCat is equal to 11 or 12, the following applies.
  - The derivation process for neighbouring 4x4 Cr blocks specified in subclause 6.4.8.4.1 is invoked with cr4x4BlkIdx as input and the output is assigned to mbAddrN, cr4x4BlkIdxN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, ((CodedBlockPatternLuma >> (cr4x4BlkIdxN >>2)) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 0 for the macroblock mbAddrN, the 4x4 Cr block with index cr4x4BlkIdxN of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, if mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip or B_Skip, ((CodedBlockPatternLuma >> (cr4x4BlkIdxN >>2)) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 1 for the macroblock mbAddrN, the 8x8 Cr block with index (cr4x4BlkIdxN >> 2) of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, transBlockN is marked as not available.
- Otherwise (ctxBlockCat is equal to 13), the following applies.
  - The derivation process for neighbouring 8x8 Cr blocks specified in subclause 6.4.8.2.1 is invoked with cr8x8BlkIdx as input and the output is assigned to mbAddrN, cr8x8BlkIdxN (with N being either A or B).
  - The variable transBlockN is derived as follows.
    - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM,, ((CodedBlockPatternLuma >>cr8x8BlkIdx) & 1) is not equal to 0 for the macroblock mbAddrN, and transform_size_8x8_flag is equal to 1 for the macroblock mbAddrN, the 8x8 Cr block with index cr8x8BlkIdxN of macroblock mbAddrN is assigned to transBlockN.
    - Otherwise, transBlockN is marked as not available.