INTERNATIONAL STANDARD ISO/IEC 14496-2:2004
TECHNICAL CORRIGENDUM 1

Published 2004-06-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Coding of audio-visual objects —

Part 2:
Visual

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 14496-2:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

In 3.43 clarify coded order, replace:

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

by:

3.43 **coded (coding) order**: The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order. In the text, “coded order”, “coding order” and “decoding order” are interchangeable.

ICS 35.040 **Ref. No. ISO/IEC 14496-2:2004/Cor.1:2004(E)**

Published in Switzerland

In 3.60 decoding order, replace:

3.60 **decoding order**: The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order.

by:

3.60 **decoding order**: The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order. In the text, “coded order”, “coding order” and “decoding order” are interchangeable.

In 3.189, clarify relationship of S-VOP and S(GMC)-VOP, replace:

3.189 **S(GMC)-VOP:** A picture that is coded using prediction based on global motion compensation from the past VOP.

by:

3.189 **S(GMC)-VOP:** A picture that is coded using prediction based on global motion compensation from the past VOP. This is syntactically defined as a sub-class of S-VOP, but is semantically different as no static sprite is used.

In clause 3, insert a subclause, time base:

3.203 **time base**: The time stamp of a VOP in a visual bitstream is coded according to some system reference clock. This reference clock is referred to as the time base of the visual bitstream. If there is no system involved, the time base of a visual bitstream typically starts at zero for the first VOP. Otherwise, the time base is provided via the interface between the system and the visual bitstream defined in other system specifications.

In subclause 6.1.1, replace:

Visual object sequence is the highest syntactic structure of the coded visual bitstream.

A visual object sequence commences with a visual_object_sequence_start_code which is followed by one or more visual objects coded concurrently. The visual object sequence is terminated by a visual_object_sequence_end_code.

At various points in the visual object sequence, a repeat visual_object_sequence_start_code can be inserted for coded video data. In that case, the repeat visual_object_sequence_start_code shall follow a particular VOP. When profile_and_level_indication indicates a Studio Profile, StudioVisualObject() shall follow it.

by:

Visual object sequence is the highest syntactic structure of the visual bitstream.

A visual object sequence commences with a visual_object_sequence_start_code which is followed by one visual object (when used in a visual bitstream), or one or more visual objects (when used in a systems bitstream with separate configuration information). The visual object sequence is terminated by a visual_object_sequence_end_code. To enable random access into the visual bitstream and recovery from errors, the visual object sequence header may be repeated before termination, as described in clause 6.2.1.

When profile_and_level_indication indicates a Studio Profile, a repeat visual_object_sequence_start_code can be inserted for coded video data at various points in the visual object sequence. In that case, the repeat visual_object_sequence_start_code shall follow a particular VOP. StudioVisualObject() shall follow it.

In subclause 6.1.3.4, clarify definition of progressive and interlaced VOP, replace:

A reconstructed VOP is obtained by decoding a coded VOP. A coded VOP may have been derived from either a progressive or interlaced frame.

by:

A reconstructed VOP is obtained by decoding a coded VOP. Depending on the value of the interlaced flag in the VOL header of the VO to which it belongs, a coded VOP is called a progressive VOP (interlaced flag set to "0") or interlaced VOP (interlaced flag set to "1").

In subclause 6.2.1 and Annex K.3, clarify the descriptions of configuration information.

1) Replace the following text in subclause 6.2.1:

The configuration information contains all data that is not part of an elementary stream, including that defined by VisualObjectSequence(), VisualObject() and VideoObjectLayer().

by:

The configuration information contains visual_object_sequence_start_code, profile_and_level_indication, user_data() (if necessary), the Visual Object Header and the Video Object Layer Header. visual_object_sequence_end_code is not included in the configuration information.

2) Add the following text to the end of Annex K.3:

When ISO/IEC 14496-1, ISO/IEC 14496-12 or ISO/IEC 14496-14 is used to carry visual streams, it is not necessary to use visual_object_sequence_end_code to signal the end of a VisualObjectSequence(). The end of a VisualObjectSequence() is indicated through either visual_object_sequence_end_code in the visual stream or the system layer defined by ISO/IEC 14496-1, ISO/IEC 14496-12 or ISO/IEC 14496-14. Usage in the context of non-MPEG systems depends on the specification of these systems.

In subclause 6.2.1, clarify usage of repeated headers. Replace:

The Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header may be repeated in a single visual bitstream. Repeating these headers enables random access into the visual bitstream and recovery of these headers when the original headers are corrupted by errors. This header repetition is used only when visual_object_type in the Visual Object Header indicates that visual object type is video. (i.e. visual_object_type==”video ID”) All of the data elements in the Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header repeated in a visual bitstream shall have the same value as in the original headers, except that first_half_vbv_occupancy and latter_half_vbv_occupancy may be changed to specify the VBV occupancy just before the removal of the first VOP following the repeated Video Object Layer Header.

by:

The Visual Object Sequence Header, followed by the Visual Object Header and the Video Object Layer Header may be repeated in a single visual bitstream. Repeating these headers enables random access into the visual bitstream and recovery of these headers when the original headers are corrupted by errors. This header repetition must only be used when visual_object_type in the Visual Object Header indicates that visual object type is video. (i.e. visual_object_type==”video ID”) All of the data elements in the Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header repeated in a visual bitstream shall have the same value as in the original headers, except that first_half_vbv_occupancy and latter_half_vbv_occupancy may be changed to specify the VBV occupancy just before the removal of the first VOP following the repeated Video Object Layer Header.

In subclause 6.2.6, fix conditions to code “dbquant” and “forward motion vector” in the spatial scalability. Replace:

else {		
If (video_object_layer_shape != “rectangular”)		
mb_binary_shape_coding()		

if ((co_located_not_coded != 1 || (scalability && (ref_select_code != '11'
|| enhancement_type == 1)) || (sprite_enable == “GMC”
&& backward_reference_vop_coding_type == “S”))
&& video_object_layer_shape != “binary only”) {		

if (!transparent_mb()) {		
modb	1-2	Vlclbf
if (modb != ‘1’) {		
mb_type	1-4	Vlclbf
if (modb == ‘00’)		
cbpb	3-6	vlclbf
if (ref_select_code != ‘00’ || !scalability) {		
If (mb_type != “1” && cbpb!=0)		
dbquant	1-2	vlclbf
If (interlaced)		
interlaced_information()		
If (mb_type == ‘01’ || mb_type == ‘0001’) {		
motion_vector(“forward”)		
if (interlaced && field_prediction)		
motion_vector(“forward”)		
}		
if (mb_type == ‘01’ || mb_type == ‘001’) {		

motion_vector(“backward”)		
if (interlaced && field_prediction)		
motion_vector(“backward”)		
}		
if (mb_type == “1”)		
motion_vector(“direct”)		
}		
if (ref_select_code == ‘00’ && scalability && cbpb !=0) {		
dbquant	1-2	vlclbf
if (mb_type == ‘01’ || mb_type == ‘1’)		
motion_vector(“forward”)		
}		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
block(i)		
}		
}		
}		
}		

by:

else {		
if (video_object_layer_shape != “rectangular”)		
mb_binary_shape_coding()		
if ((co_located_not_coded != 1 || (scalability && (ref_select_code != '11' || enhancement_type == 1)) || (sprite_enable == “GMC” && backward_reference_vop_coding_type == “S”)) && video_object_layer_shape != “binary only”) {		
if (!transparent_mb()) {		
modb	1-2	vlclbf
if (modb != ‘1’) {		
mb_type	1-4	vlclbf
if (modb == ‘00’)		
cbpb	3-6	vlclbf
if (ref_select_code != ‘00’ || !scalability) {		
if (mb_type != “1” && cbpb!=0)		
dbquant	1-2	vlclbf
if (interlaced)		
interlaced_information()		
if (mb_type == ‘01’ || mb_type == ‘0001’) {		
motion_vector(“forward”)		
if (interlaced && field_prediction)		
motion_vector(“forward”)		
}		
if (mb_type == ‘01’ || mb_type == ‘001’) {		
motion_vector(“backward”)		

if (interlaced && field_prediction)		
motion_vector(“backward”)		
}		
if (mb_type == “1”)		
motion_vector(“direct”)		
}		
if (ref_select_code == ‘00’ && scalability) {		
if (cbpb !=0)		
dbquant	1-2	vlclbf
if (mb_type == ‘01’ || mb_type == ‘1’)		
motion_vector(“forward”)		
}		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
block(i)		
}		
}		
}		
}		

In subclauses 6.2.14.3 and 6.3.14.3, add conditions for coding “fgs_field_prediction” and correct the corresponding descriptions. Replace:

fgs_motion_interlaced_information( ) {	**No. of bits**	**Mnemonic**
fgs_field_prediction	1	bslbf
if (fgs_field_prediction) {		

if (fgs_vop_coding_type == “P” ||
(fgs_vop_coding_type==“B” &&
fgs_b_mb_type!=“001”)){		

fgs_forward_top_field_reference	1	bslbf
fgs_forward_bottom_field_reference	1	bslbf
}		
if ((fgs_vop_coding_type == “B”) && (fgs_b_mb_type != “1”)) {		
fgs_backward_top_field_reference	1	bslbf
fgs_backward_bottom_field_reference	1	bslbf
}		
}		
}		

by:

fgs_motion_interlaced_information( ) {	**No. of bits**	**Mnemonic**
if (((fgs_vop_coding_type == "P") && (fgs_p_mb_type == 0)) || ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1"))) {		
fgs_field_prediction	1	bslbf
if (fgs_field_prediction) {		

if (fgs_vop_coding_type == "P" ||
(fgs_vop_coding_type=="B" &&
fgs_b_mb_type!="001")){		

fgs_forward_top_field_reference	1	bslbf
fgs_forward_bottom_field_reference	1	bslbf
}		
if ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1")) {		
fgs_backward_top_field_reference	1	bslbf
fgs_backward_bottom_field_reference	1	bslbf
}		
}		
}		
}		

Following the previous item, in subclause 6.3.14.3 replace:

fgs_field_prediction − This is a 1-bit flag indicating whether the FGS motion macroblock is field predicted or frame predicted. This flag is set to '1' when the FGS motion macroblock is predicted using field motion vectors. If it is set to '0' then frame prediction (16x16 or 8x8) will be used. This flag is only present in the bitstream if the interlaced flag is set to "1", and either the fgs_p_mb_type is "0" in a "P" fgs vop or the FGS motion macroblock is in a "B" fgs vop.

by:

fgs_field_prediction − This is a 1-bit flag indicating whether the FGS motion macroblock is field predicted or frame predicted. This flag is set to '1' when the FGS motion macroblock is predicted using field motion vectors. If it is set to '0' then frame prediction (16x16 or 8x8) will be used. This flag is only present in the bitstream if the interlaced flag is set to "1", and either the fgs_p_mb_type is "0" in a "P" fgs vop or the non-direct mode FGS motion macroblock is in a "B" fgs vop.

In subclause 6.3.2, clarify definition of video range. Replace:

video_range: This one-bit flag indicates the black level and range of the luminance and chrominance signals.

by:

video_range: This one-bit flag indicates the black level and range of the luminance and chrominance signals as specified in the explanation above Table 6-10.

In subclauses 6.3.3 and 6.3.13.3, clarify definition of condition for random access. In both subclauses, replace

random_accessible_vol: This flag may be set to "1" to indicate that every VOP in this VOL is individually decodable. If all of the VOPs in this VOL are intra-coded VOPs and some more conditions are satisfied then random_accessible_vol may be set to "1". The flag random_accessible_vol is not used by the decoding process. random_accessible_vol is intended to aid random access or editing capability. This shall be set to "0" if any of the VOPs in the VOL are non-intra coded or certain other conditions are not fulfilled.

by:

random_accessible_vol: This flag may be set to "1" to indicate that every VOP in this VOL is individually decodable. If all of the VOPs in this VOL are intra-coded VOPs, then random_accessible_vol may be set to "1". The flag random_accessible_vol is not used by the normative decoding process, random_accessible_vol is intended to aid random access or editing capability. This shall be set to "0" if any of the VOPs in the VOL are non-intra coded.

In subclause 6.3.3, clarify semantics of fixed_vop_rate. Replace:

fixed_vop_rate: This is a one-bit flag which indicates that all VOPs are coded with a fixed VOP rate. It shall only be '1' if and only if all the distances between the display time of any two successive VOPs in the display order in the video object layer are constant. In this case, the VOP rate can be derived from the fixed_vop_time_increment. If it is '0' the display time between any two successive VOPs in the display order can be variable thus indicated by the time stamps provided in the VOP header.

by:

fixed_vop_rate: This is a one-bit flag which indicates that all VOPs are coded with a fixed VOP rate. It shall only be '1' if and only if all the distances between the display times of any two successive VOPs in the display order in the video object layer are constant and less than one second. In this case, the VOP rate can be derived from the fixed_vop_time_increment. If it is '0' the display time between any two successive VOPs in the display order can be variable as indicated by the time stamps provided in the VOP header.

In subclause 6.3.3, clarify the naming for representing the fixed VOP period. Replace:

fixed_vop_time_increment: This value represents the number of ticks between two successive VOPs in the display order. The length of a tick is given by vop_time_increment_resolution. It can take a value in the range of [0,vop_time_increment_resolution). The number of bits representing the value is calculated as the minimum number of unsigned integer bits required to represent the above range. fixed_vop_time_increment shall only be present if fixed_vop_rate is '1' and its value must be identical to the constant given by the distance between the display time of any two successive VOPs in the display order. In this case, the fixed VOP rate is given as (vop_time_increment_resolution / fixed_vop_time_increment). A zero value is forbidden.

EXAMPLE
VOP time = tick × vop_time_increment
= vop_time_increment / vop_time_increment_resolution

Table 6-18 — Examples of vop_time_increment_resolution, fix_vop_time_increment, and vop_time_increment

Fixed VOP rate = 1/VOP time	vop_time_increment_ resolution	fixed_vop_time_ increment	vop_time_increment
15Hz	15	1	0, 1, 2, 3, 4,…
7.5Hz	15	2	0, 2, 4, 6, 8,…
29.97…Hz	30000	1001	0, 1001, 2002, 3003,…
59.94…Hz	60000	1001	0, 1001, 2002, 3003,…

by:

fixed_vop_time_increment: This value represents the number of ticks between two successive VOPs in the display order. The length of a tick is given by vop_time_increment_resolution. It can take a value in the range of [0,vop_time_increment_resolution). The number of bits representing the value is calculated as the minimum number of unsigned integer bits required to represent the above range. fixed_vop_time_increment shall only be present if fixed_vop_rate is '1' and its value must be identical to the constant given by the distance between the display time of any two successive VOPs in the display order. In this case, the fixed VOP rate is given as (vop_time_increment_resolution ÷ fixed_vop_time_increment). A zero value is forbidden.

EXAMPLE
Fixed VOP Period = tick × vop_time_increment
= vop_time_increment ÷ vop_time_increment_resolution

Table 6-18 — Examples of vop_time_increment_resolution, fix_vop_time_increment, and vop_time_increment

Fixed VOP rate = 1÷Fixed VOP Period	vop_time_increment_ resolution	fixed_vop_time_ increment	vop_time_increment
15Hz	15	1	0, 1, 2, 3, 4,…
7.5Hz	15	2	0, 2, 4, 6, 8,…
29.97…Hz	30000	1001	0, 1001, 2002, 3003,…
59.94…Hz	60000	1001	0, 1001, 2002, 3003,…

In subclause 6.3.3, clarify usage of default matrices. Replace:

load_intra_quant_mat: This is a one-bit flag which is set to ‘1’ when intra_quant_mat follows. If it is set to ‘0’ then there is no change in the values that shall be used.

by:

load_intra_quant_mat: This is a one-bit flag which is set to ‘1’ when intra_quant_mat follows. If it is set to ‘0’, the values of the default intra matrix shall be used.

Following the previous item, replace:

load_nonintra_quant_mat: This is a one-bit flag which is set to ‘1’ when nonintra_quant_mat follows. If it is set to ‘0’ then there is no change in the values that shall be used.

by:

load_nonintra_quant_mat: This is a one-bit flag which is set to ‘1’ when nonintra_quant_mat follows. If it is set to ‘0’, the values of the default nonintra matrix shall be used.

In subclause 6.3.4, remove the redundant condition. Replace:

closed_gov: This is a one-bit flag which indicates the nature of the predictions used in the first consecutive B-VOPs (if any) immediately following the first coded I-VOP after the GOV header .The closed_gov is set to '1' to indicate that these B-VOPs have been encoded using only backward prediction or intra coding. This bit is provided for use during any editing which occurs after encoding. If the previous pictures have been removed by editing, broken_link may be set to '1' so that a decoder may avoid displaying these B-VOPs following the first I-VOP following the group of plane header. However if the closed_gov bit is set to '1', then the editor may choose not to set the broken_link bit as these B-VOPs can be correctly decoded.

by:

closed_gov: This is a one-bit flag which indicates the nature of the predictions used in the first consecutive B-VOPs (if any) immediately following the first coded I-VOP after the GOV header .The closed_gov is set to '1' to indicate that these B-VOPs have been encoded using only backward prediction. This bit is provided for use during any editing which occurs after encoding. If the previous pictures have been removed by editing, broken_link may be set to '1' so that a decoder may avoid displaying these B-VOPs following the first I-VOP following the group of plane header. However if the closed_gov bit is set to '1', then the editor may choose not to set the broken_link bit as these B-VOPs can be correctly decoded.

In subclause 6.3.5, clarify the case of not_coded I-VOP. Replace:

vop_coded: This is a 1-bit flag which when set to '0' indicates that no subsequent data exists for the VOP. In this case, the following decoding rules apply: If binary shape or alpha plane does exist for the VOP (i.e. video_object_layer_shape != "rectangular"), it shall be completely transparent. If binary shape or alpha plane does not exist for the VOP (i.e. video_object_layer_shape == "rectangular"), the luminance and chrominance planes of the reconstructed VOP shall be filled with the forward reference VOP as defined in subclause 7.6.7.

by:

vop_coded: This is a 1-bit flag which when set to '0' indicates that no subsequent data exists for the VOP. In this case, the following decoding rules apply: If binary shape or alpha plane does exist for the VOP (i.e. video_object_layer_shape != "rectangular"), it shall be completely transparent. If binary shape or alpha plane does not exist for the VOP (i.e. video_object_layer_shape == "rectangular"), the luminance and chrominance planes of the reconstructed VOP shall be filled with the forward reference VOP as defined in subclause 7.6.7.

NOTE: These decoding rules apply even when vop_coding_type indicates an intra-coded VOP.

In subclause 6.3.5, clarify meaning of alternate scan flag. Replace:

alternate_vertical_scan_flag: This is a 1-bit flag which when set to "1" indicates the use of alternate vertical scan for interlaced VOPs.

by:

alternate_vertical_scan_flag: This is a 1-bit flag which when set to "1" indicates the use of alternate vertical scan for decoding of interlaced VOPs. When set to '0', it indicates that inverse scan pattern is selected in the same way as for decoding progressive VOPs, described in subclause 7.4.2.

In subclause 6.3.6, define cbpc. Replace:

mcbpc: This is a variable length code that is used to derive the macroblock type and the coded block pattern for chrominance . It is always included for coded macroblocks. Table B-6 and Table B-7 list all allowed codes for mcbpc in I-, P-, and S(GMC)- VOPs respectively. The values of the column "MB type" in these tables are

used as the variable “derived_mb_type” which is used in the respective syntax part for motion and texture decoding. In P-VOPs using the short video header format (i.e., when short_video_header is 1), mcbpc codes indicating macroblock type 2 shall not be used.

by:

mcbpc: This is a variable length code that is used to derive the macroblock type and the coded block pattern for chrominance (cbpc). It is always included for coded macroblocks. Table B-6 and Table B-7 list all allowed codes for mcbpc in I-, P-, and S(GMC)- VOPs respectively. The values of the column “MB type” in these tables are used as the variable “derived_mb_type” which is used in the respective syntax part for motion and texture decoding. In P-VOPs using the short video header format (i.e., when short_video_header is 1), mcbpc codes indicating macroblock type 2 shall not be used.

In subclause 6.3.14.2, clarify the default macroblock type in the “B” fgs vops. Replace:

fgs_modb – This is a 1-bit flag that signals if an FGS motion macroblock is coded or not in a “B” fgs vop. When set to’1’ it indicates that an FGS motion macroblock is not coded and no further data is included in the bitstream for this FGS motion macroblock; decoder shall treat this FGS motion macroblock as motion vector equal to zero. When set to ‘0’ it indicates that the FGS motion macroblock is coded and its data is included in the bitstream.

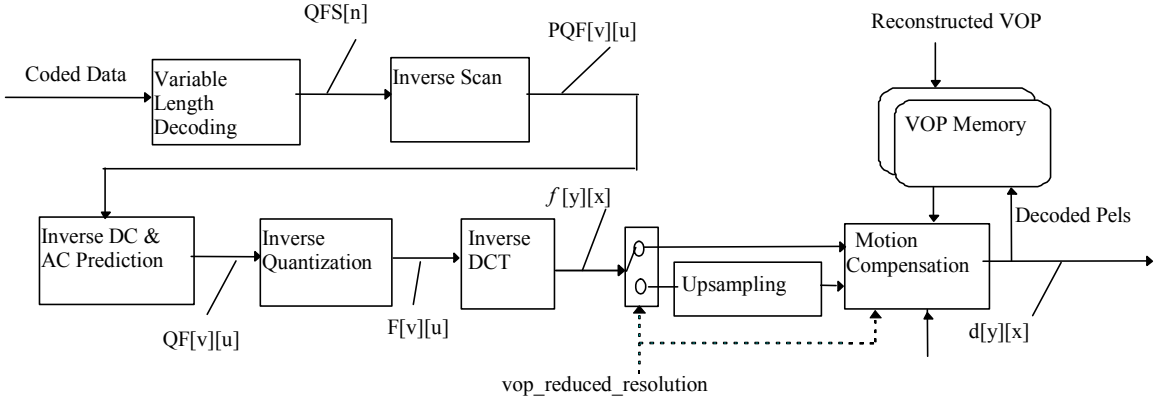
fgs_b_mb_type – This variable length code is present only in coded FGS motion macroblocks of “B” fgs vops. The codes for fgs_b_mb_type are shown in Table 6-111.

by:

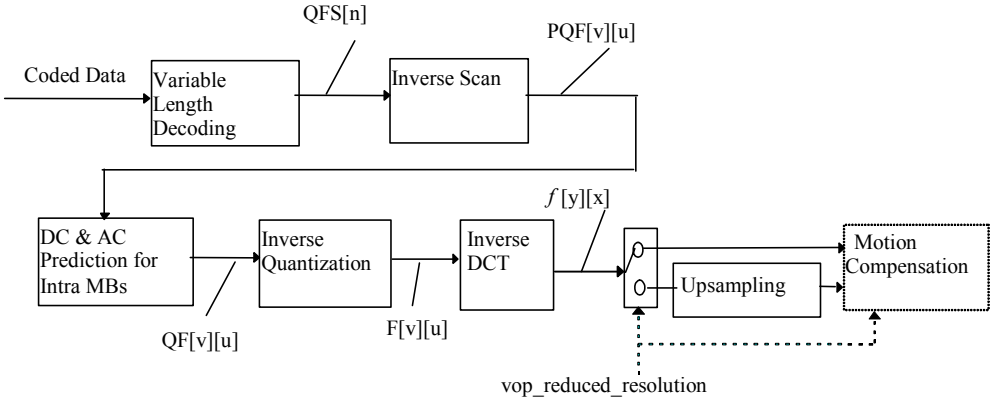
fgs_modb – This is a 1-bit flag that signals if an FGS motion macroblock is coded or not in a "B" fgs vop. When set to'1' it indicates that an FGS motion macroblock is not coded and no further data is included in the bitstream for this FGS motion macroblock; decoder shall treat this FGS motion macroblock as default macroblock type and its delta motion vector equal to zero. When set to '0' it indicates that the FGS motion macroblock is coded and its data is included in the bitstream.

fgs_b_mb_type – This variable length code is present only in coded FGS motion macroblocks of "B" fgs vops. The codes for fgs_b_mb_type are shown in Table 6-111. When mb_type is not present (i.e. fgs_modb=='1') for a FGS motion macroblock of "B" fgs vops, the macroblock type is set to the default type. The default macroblock type is "direct".

In Figure 7-3, clarify that MC is not part of texture decoding. Replace:



by:



In subclause 7.4.1.2, improve wording. Replace:

When short_video_header is 1, the most commonly occurring EVENTS are coded with the variable length codes given in Table B-17 (for all coefficients other than intra DC whether in intra or inter blocks). The last bit “s” denotes the sign of level, “0” for positive and “1” for negative.

When short_video_header is 0, the variable length code table is different for intra blocks and inter blocks. The most commonly occurring EVENTs for the luminance and chrominance components of intra blocks in this case are decoded by referring to the intra columns of Table B-23 when reversible_vlc is set to ‘1’ in I-, P-, or S(GMC)-VOPs, and by referring to Table B-16, otherwise. The most commonly occurring EVENTs for the luminance and chrominance components of inter blocks in this case are decoded by referring to the inter columns of Table B-23 when reversible_vlc is set to ‘1’ in I-, P-, or S(GMC)-VOPs, and by referring to Table B-17, otherwise. The last bit “s” denotes the sign of level, “0” for positive and “1” for negative. The combinations of (LAST, RUN, LEVEL) not represented in these tables are decoded as described in subclause 7.4.1.3.

by:

"When short_video_header is 1, the (LAST, RUN, LEVEL) events are coded with the variable length codes given in Table B-17 (for all coefficients other than intra DC whether in intra or inter blocks). When the VLC is not equal to the "escape" code ("0000 011") the last bit “s” denotes the sign of level, “0” for positive and “1” for negative. When the VLC is equal to the "escape" code, the (LAST, RUN, LEVEL) events are further coded as described in subclause 7.4.1.3.

When short_video_header is 0, the variable length code table is different for intra blocks and inter blocks. The (LAST, RUN, LEVEL) events for the luminance and chrominance components of intra blocks in this case are decoded by referring to the intra columns of Table B-23 when reversible_vlc is set to ‘1’ in I-, P-, or S(GMC)-VOPs, and by referring to Table B-16, otherwise. The (LAST, RUN, LEVEL) events for the luminance and chrominance components of inter blocks in this case are typically coded by referring to the inter columns of Table B-23 when reversible_vlc is set to ‘1’ in I-, P-, or S(GMC)-VOPs, and by referring to Table B-17, otherwise. When the VLC is not equal to the "escape" code, the last bit “s” denotes the sign of level, “0” for positive and “1” for negative. When the VLC is equal to the "escape" code, the (LAST, RUN, LEVEL) events are further coded as described in subclause 7.4.1.3.

In subclause 7.4.3.3, add one extra condition for resetting AC coefficients to zero. Replace:

If the prediction block (block 'A' or block 'C') is outside of the boundary of the VOP or video packet, then all the prediction coefficients of that block are assumed to be zero.

by:

If the prediction block (block 'A' or block 'C') is outside of the boundary of the VOP or video packet, or if the block is not part of an intra-coded macroblock, then all the prediction coefficients of that block are assumed to be zero.

In subclause 7.6.2.2, clarify the explanation of Figure 7-30. Replace:

In quarter sample mode interpolation, for each block of size MxN in the reference VOP which position is defined by the decoded motion vector for the block to be predicted, a reference block of size (M+1)x(N+1) biased in the direction of the half or quarter sample position is read from the reconstructed and padded reference VOP. Then, this reference block is symmetrically extended at the block boundaries by three samples using block boundary mirroring according to Figure 7-30.

by:

In quarter sample mode interpolation, for each block of size MxN in the reference VOP which position is defined by the decoded motion vector for the block to be predicted, a reference block of size (M+1)x(N+1) biased in the direction of the half or quarter sample position is read from the reconstructed and padded reference VOP. Then, this reference block is symmetrically extended at the block boundaries using block boundary mirroring as illustrated in Figure 7-30. An extension by three samples is made.

Following the previous item, replace:

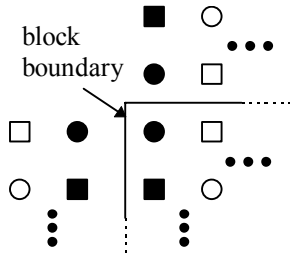


Figure 7-30 — block boundary mirroring

by:

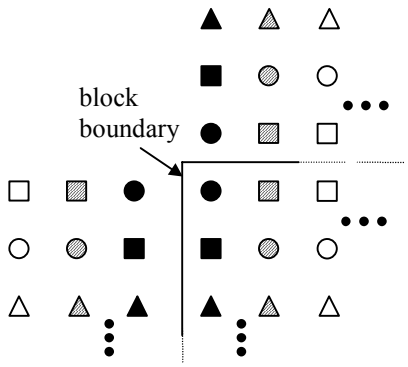


Figure 7-30 — block boundary mirroring

To clarify the order of steps to generate the reference, add a subclause after subclause 7.6.2.2.2:

7.6.2.2.3 Generation of two-dimensional reference block

The whole process to generate the reference block is to do the horizontal processes (mirroring, 8-tap FIR filtering, bilinear interpolation) and get the corresponding samples of horizontal position (in quarter-pel unit), first. Then, do vertical processing (mirroring, 8-tap FIR filtering, bilinear interpolation) and get the corresponding samples of vertical position. The following steps generate the reference block of size MxN in the quarter-pel MC.

1. If the x-component of a motion vector is fractional, horizontal mirroring and filtering are applied to (M+1)x(N+1) reference block in the previous frame to get the horizontal half-pel interpolated reference of size Mx(N+1). Otherwise, all horizontal processes (Steps 1 and 2) are skipped, and the left-top region of size Mx(N+1) is used in Step 3.
2. In the 1/4 and 3/4 cases, the horizontal bilinear filter is applied to the previous result to get a new horizontal quarter-sampled reference, of size Mx(N+1).
3. If the y-component of a motion vector is fractional, vertical mirroring and filtering are applied to Mx(N+1) reference block in the previous step to get the vertical half-pel interpolated reference of size MxN. Otherwise, all vertical processes (Steps 3 and 4) are skipped, and the left-top region of size MxN is used in Step 5.
4. In the 1/4 and 3/4 cases, the vertical bilinear filter is applied to the previous result to get a new vertical quarter-sampled reference, of size MxN.
5. The so-obtained MxN block is the final result used for quarter-pel motion compensation prediction.

In clause 7.6.3, clarify the range of motion vectors of interlaced macroblocks. Replace:

The parameters in the bitstream shall be such that the components of the reconstructed differential motion vector, MVDx and MVDy, shall lie in the range [low:high], at the time of their use in calculating the values of MVx and MVy (i.e., intermediate values of MVDx and MVDy may occur that are outside the range [low:high]). In addition the components of the reconstructed motion vector, *MVx* and *MVy,* shall also lie in the range [*low* : *high*]. The allowed range [low : high] for the motion vectors depends on the parameter vop_fcode; it is shown in Table 7-9.

by:

The parameters in the bitstream shall be such that the components of the reconstructed differential motion vector, MVDx and MVDy, shall lie in the range [low:high], at the time of their use in calculating the values of MVx and MVy (i.e., intermediate values may occur after adding MVDx and MVDy to the predictor that are outside the range [low:high]). In addition the components of the reconstructed motion vector, *MVx* and *MVy,* shall also lie in the range [*low* : *high*]. The allowed range [low : high] for the motion vectors depends on the parameter vop_fcode; it is shown in Table 7-9. In the case of interlaced macroblocks, the allowed range for reconstructed vertical motion vector component is halved compared to what is specified in Table 7-9. For example, when vop_fcode is 1, motion vector range in unitless integers is [-16, 15].

To finalize the previous item, add the following text to the end of subclause 7.6.3:

NOTE: In the case of interlaced macroblocks, different values MVDy may result in the same reconstructed motion vector.

In subclause 7.6.4, clarify the padding process in case of interlaced sequences. Replace:

Motion vectors are allowed to point outside the decoded area of a reference VOP when (and only when) the short video header format is not in use (i.e., when short_video_header is 0). For an arbitrary shape VOP, the decoded area refers to the area within the bounding rectangle, padded as described in subclause 7.6.1. A bounding rectangle is defined by vop_width and vop_height extended to multiple of 16. In case of half sample mode, when a sample referenced by a motion vector stays outside the decoded VOP area, an edge sample is used. This edge sample is retrieved by limiting the motion vector to the last full pel position inside the decoded VOP area. Limitation of a motion vector is performed on a sample basis and separately for each component of the motion vector, as depicted in Figure 7-33. In case of quarter sample mode, when a sample needed for interpolation (see subclause 7.6.2.2) stays outside the decoded VOP area, an edge sample is used prior to block boundary mirroring.

by:

Motion vectors are allowed to point outside the decoded area of a reference VOP when (and only when) the short video header format is not in use (i.e., when short_video_header is 0). For an arbitrary shape VOP, the decoded area refers to the area within the bounding rectangle, padded as described in subclause 7.6.1. A bounding rectangle is defined by vop_width and vop_height extended to multiple of 16. In case of half sample mode, when a sample referenced by a motion vector stays outside the decoded VOP area, an edge sample is used. This edge sample is retrieved by limiting the motion vector to the last full pel position inside the decoded VOP area. Limitation of a motion vector is performed on a sample basis and separately for each component of the motion vector, as depicted in Figure 7-33. In case of quarter sample mode, when a sample needed for interpolation (see subclause 7.6.2.2) stays outside the decoded VOP area, an edge sample is used prior to block boundary mirroring. Selection of the edge samples is performed identically for interlaced and progressive VOPs.

NOTE: In case of interlaced VOPs, frame organization of lines is used to determine the edge samples. This may mean that an edge sample is selected from a different field than indicated as prediction reference.

In subclauses 7.6.9.2 to 7.6.9.4, correct the pseudo code for motion compensation. In subclause 7.6.9.2, replace:

```
mc(Pf_Y, ref_Y_for, x, y, 16, 16, MVFx, MVFy, 0, 0, 0, 1);
mc(Pf_U, ref_U_for, x/2, y/2, 8, 8,  MVFx_chro, MVFy_chro, 0, 0, 0,1);
mc(Pf_V, ref_V_for, x/2, y/2, 8, 8,  MVFx_chro, MVFy_chro, 0, 0, 0,1);
```

by:

```
mc(Pf_Y, ref_Y_for, x, y, 16, 16, MVFx, MVFy, rounding_type, 0, 0, 1);
mc(Pf_U, ref_U_for, x/2, y/2, 8, 8,  MVFx_chro, MVFy_chro, rounding_type, 0, 0,1);
mc(Pf_V, ref_V_for, x/2, y/2, 8, 8,  MVFx_chro, MVFy_chro, rounding_type, 0, 0,1);
```

Following the previous item, in subclause 7.6.9.3, replace:

```
mc(Pb_Y, ref_Y_back, x, y, 16, 16, MVBx, MVBy, 0, 0, 0, 1);
mc(Pb_U, ref_U_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, 0, 0, 0,1);
mc(Pb_V, ref_V_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, 0, 0, 0,1);
```

by:

```
mc(Pb_Y, ref_Y_back, x, y, 16, 16, MVBx, MVBy, rounding_type, 0, 0, 1);
mc(Pb_U, ref_U_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, rounding_type, 0, 0,1);
mc(Pb_V, ref_V_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, rounding_type, 0, 0,1);
```

Following the previous item, in subclause 7.6.9.4, replace:

```
mc(Pf_Y, ref_Y_for, x, y, 16, 16, MVFx, MVFy, 0, 0, 0, 1);
mc(Pf_U, ref_U_for, x/2, y/2, 8, 8, MVFx_chro, MVFy_chro, 0, 0, 0,1);
mc(Pf_V, ref_V_for, x/2, y/2, 8, 8, MVFx_chro, MVFy_chro, 0, 0, 0,1);
mc(Pb_Y, ref_Y_back, x, y, 16, 16, MVBx, MVBy, 0, 0, 0, 1);
mc(Pb_U, ref_U_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, 0, 0, 0,1);
mc(Pb_V, ref_V_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, 0, 0, 0,1);
Pi_Y[i][j] = (Pf_Y[i][j] + Pb_Y[i][j] + 1)>>1;          i,j=0,1,2…15;
Pi_U[i][j] = (Pf_U[i][j] + Pb_U[i][j] + 1)>>1;      i,j=0,1,2…7;
Pi_V[i][j] = (Pf_V[i][j] + Pb_V[i][j] + 1)>>1;          i,j=0,1,2…7;
```

by:

```
mc(Pf_Y, ref_Y_for, x, y, 16, 16, MVFx, MVFy, rounding_type, 0, 0, 1);
mc(Pf_U, ref_U_for, x/2, y/2, 8, 8, MVFx_chro, MVFy_chro, rounding_type, 0, 0,1);
mc(Pf_V, ref_V_for, x/2, y/2, 8, 8, MVFx_chro, MVFy_chro, rounding_type, 0, 0,1);
mc(Pb_Y, ref_Y_back, x, y, 16, 16, MVBx, MVBy, rounding_type, 0, 0, 1);
mc(Pb_U, ref_U_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, rounding_type, 0, 0,1);
mc(Pb_V, ref_V_back, x/2, y/2, 8, 8, MVBx_chro, MVBy_chro, rounding_type, 0, 0,1);
Pi_Y[i][j] = (Pf_Y[i][j] + Pb_Y[i][j] + 1)>>1;          i,j=0,1,2…15;
Pi_U[i][j] = (Pf_U[i][j] + Pb_U[i][j] + 1)>>1;          i,j=0,1,2…7;
Pi_V[i][j] = (Pf_V[i][j] + Pb_V[i][j] + 1)>>1;          i,j=0,1,2…7;
```

In subclause 7.6.9.5.2, correct and clarify the direct mode definition. Replace:

where {(MVx[i],MVy[i]), i = 0,1,2,3} are the MV vectors of the co-located macroblock, TRD is the difference in temporal reference of the B-VOP and the previous reference VOP. TRD is the difference in temporal reference of the temporally next reference VOP with temporally previous reference VOP, assuming B-VOPs or skipped VOPs in between. In case that the MV components of the co-located macroblock are given in quarter sample units and the components MVDx and MVDy of the delta vector are given in half sample units, the components of the co-located macroblock {(MVx[i],MVy[i]), i = 0,1,2,3} are converted to half sample units before the calculation of the forward and the backward motion vectors {(MVFx[i], MVFy[i]), (MVBx[i], MVBy[i]), i = 0,1,2,3}. For this conversion, each component {(MVx[i],MVy[i]), i = 0,1,2,3} is first divided by 2 and then rounded on the basis of Table 7-13.

by:

where {(MVx[i],MVy[i]), i = 0,1,2,3} are the MV vectors of the co-located macroblock, TRB is the difference in temporal position of the B-VOP and the previous reference VOP. TRD is the difference in temporal position of the temporally next reference VOP with the temporally previous reference VOP.

In subclause 7.6.9.6, clarify that macroblocks in I-VOPs cannot be skipped. Replace:

If the co-located macroblock in the most recently decoded I- or P-VOP is skipped, the current B-macroblock is treated as the forward mode with the zero motion vector (MVFx,MVFy). If the modb equals to '1' the current B-macroblock is reconstructed by using the direct mode with zero delta vector. If the co-located macroblock in the most recently decoded S(GMC)-VOP is skipped, this co-located macroblock is treated as a non-skipped macroblock with the averaged motion vector defined in subclause 7.8.7.3 for the current B-macroblock.

by:

If the co-located macroblock in the most recently decoded P-VOP is skipped, the current B-macroblock is treated as the forward mode with the zero motion vector (MVFx,MVFy). If modb is equal to '1' the current B-macroblock is reconstructed by using the direct mode with zero delta vector. If the co-located macroblock in the most recently decoded S(GMC)-VOP is skipped, this co-located macroblock is treated as a non-skipped macroblock with the averaged motion vector defined in subclause 7.8.7.3 for the current B-macroblock.

In subclause 7.7.1, clarify that reference blocks in Figure 7-45 are field coded. Replace:

When dct_type flag is set to '1' (field DCT coding), DCT coefficients of luminance data are formed such that each 8x8 block consists of data from one field as being shown in Figure 6-12. DC and optional AC (see "ac_pred_flag") prediction will be performed for a intra-coded macroblock. For the intra macroblocks which have dct_type flag being set to "1", DC/AC prediction are performed to field blocks shown in Figure 7-45. After taking inverse DCT, all luminance blocks will be inverse permuted back to (frame) macroblock. Chrominance (block) data are not effected by dct_type flag.

by:

When dct_type flag is set to '1' (field DCT coding), DCT coefficients of luminance data are formed such that each 8x8 block consists of data from one field as being shown in Figure 6-12. DC and optional AC (see "ac_pred_flag") prediction will be performed for an intra-coded macroblock. For the intra macroblocks which have dct_type flag being set to "1", DC/AC prediction are performed to field blocks shown in Figure 7-45, using field-coded blocks A to E as references. After taking inverse DCT, all luminance blocks will be inverse permuted back to (frame) macroblock. Chrominance (block) data are not effected by dct_type flag.

Following the previous item, replace:

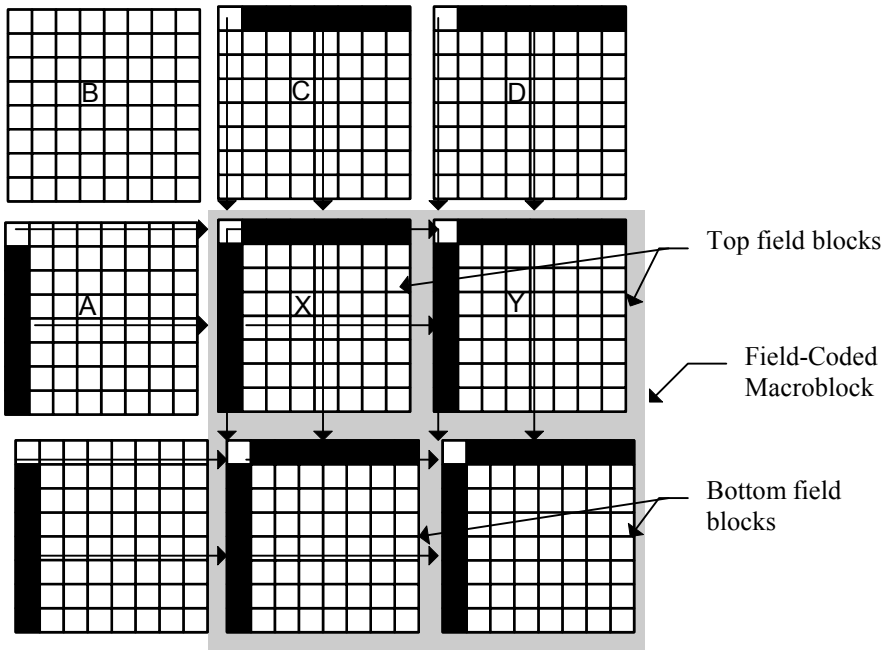


Figure 7-45 — Previous neighboring blocks used in DC/AC prediction for interlaced intra blocks.

by:

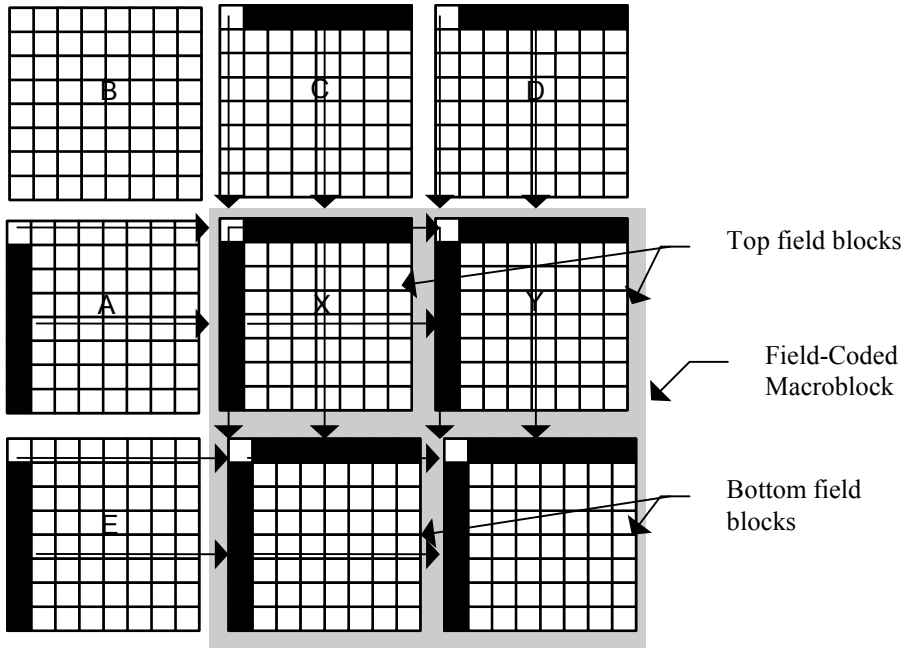


Figure 7-45 — Previous neighboring blocks used in DC/AC prediction for interlaced intra blocks.

In subclause 7.7.2.2, clarify the definition of TRB[i] and TRD[i]. Replace:

The calculation of TRD[i] and TRB[i] depends not only on the current field, reference field, and frame temporal references, but also on whether the current video is top field first or bottom field first.

$$TRD[i] = 2*(T(future)//Tframe - T(past)//Tframe) + \delta[i]$$

$$TRB[i] = 2*(T(current)//Tframe - T(past)//Tframe) + \delta[i]$$

where T(future), T(current) and T(past) are the cumulative VOP times calculated from modulo_time_base and vop_time_increment of the future, current and past VOPs in display order. Tframe is the frame period determined by

$$Tframe = T(first\_B\_VOP) - T(past\_reference\_of\_first\ B\_VOP)$$

where first_B_VOP denotes the first B-VOP following the Video Object Layer syntax. The important thing about Tframe is that the period of time between consecutive fields which constitute an interlaced frame is assuemed to be 0.5 * Tframe for purposes of scaling the motion vectors.

by:

The calculation of TRD[i] and TRB[i] depends not only on the current field, reference field, and frame temporal references, but also on whether the current video is top field first or bottom field first.

$$TRD[i] = 2*(T(future)//Tframe - T(past)//Tframe) + \delta[i]$$

$$TRB[i] = 2*(T(current)//Tframe - T(past)//Tframe) + \delta[i]$$

where T(future), T(current) and T(past) are the cumulative VOP times calculated from modulo_time_base and vop_time_increment of the future, current and past VOPs in display order. Tframe is the frame period determined by (provided that no VOPs is skipped between the first_B_VOP and the past reference)

$$Tframe = T(first\_B\_VOP) - T(past\_reference\_of\_first\ B\_VOP)$$

where first_B_VOP denotes the first B-VOP following the Video Object Layer syntax. The important thing about Tframe is that the period of time between consecutive fields which constitute an interlaced frame is assumed to be 0.5 * Tframe for purposes of scaling the motion vectors.

NOTE: VOP time is the duration of a single VOP. The cumulative VOP time is the sum of VOP times starting from the last synchronization point. For the case of fixed VOP rate, an example for computation of VOP time is given in subclause 6.3.2.1.

In subclause 7.7.2.2, add references to tables 7-14 and 7-15. Replace:

For interlaced B-VOP motion vector predictors, four prediction motion vectors (PMVs) are used:

Table 7-14 — Prediction motion vector allocation for interlaced P- and S(GMC)-VOPs

Function	**1.1.1 MV**
Top field forward	0
Bottom field forward	1
Top field backward	2
Bottom field backward	3

These PMVs are used as follows for the different macroblock prediction modes:

Table 7-15 — Prediction motion vectors for interlaced B-VOP decoding

Macroblock mode	**PMVs used**	**PMVs updated**
Direct	none	none
Frame forward	0	0,1
Frame backward	2	2,3
Frame bidirectional	0,2	0,1,2,3
Field forward	0,1	0,1
Field backward	2,3	2,3
Field bidirectional	0,1,2,3	0,1,2,3

The PMVs used by a macroblock are set to the value of current macroblock motion vectors after being used.

When a frame macroblock is decoded, the two field PMVs (top and bottom field) for each prediction direction are set to the same frame value. The PMVs are reset to zero at the beginning of each row of macroblocks. The predictors are not zeroed by skipped macroblocks or direct mode macroblocks.

by:

For interlaced B-VOP motion vector predictors, four prediction motion vectors (PMVs) are supported as defined in Table 7-14.

Table 7-14 — Prediction motion vector definition for interlaced P- and S(GMC)-VOPs

Function	PMV
Top field forward	0
Bottom field forward	1
Top field backward	2
Bottom field backward	3

These PMVs are used for the different macroblock prediction modes as specified in Table 7-15.

Table 7-15 — Prediction motion vectors for interlaced B-VOP decoding

Macroblock mode	PMVs used	PMVs updated
Direct	none	none
Frame forward	0	0,1
Frame backward	2	2,3
Frame bidirectional	0,2	0,1,2,3
Field forward	0,1	0,1
Field backward	2,3	2,3
Field bidirectional	0,1,2,3	0,1,2,3

The PMVs used by a macroblock are set to the value of current macroblock motion vectors after being used.

When a macroblock is decoded in frame mode, the two field PMVs (top and bottom field) for each prediction direction are set to the same frame value. The PMVs are reset to zero at the beginning of each row of macroblocks. The predictors are not zeroed by skipped macroblocks or direct mode macroblocks.

In subclause 7.7.2.2, correct and clarify the direct mode definition with interlace. Replace:

The temporal references (TRB[i] and TRD[i]) are distances in time expressed in field periods. Figure 7-48 shows how they are defined for the case where i is 0 (top field of the B-VOP). The bottom field is analogously.

by:

The values TRB[i] and TRD[i] are distances in time expressed in field periods. Figure 7-48 shows how they are defined for the case where i is 0 (top field of the B-VOP). The bottom field is analogously.

In subclause 7.7.2.2, correct the constraint of vop_time_increment_resolution for interlaced sequences. Replace:

The top field prediction is based on the top field motion vector of the P-VOP macroblock of the future reference VOP. The past reference field is the reference field selected by the co-located macroblock of the future reference VOP for the top field. Analogously, the bottom field predictor is the average of pixels obtained from the future reference VOP's bottom field and the past reference field referenced by the bottom field motion vector of the corresponding macroblock of the future reference VOP. When interlaced direct mode is used, vop_time_increment_resolution must be the smallest integer greater than or equal to the number of frames per second. In each VOP, vop_time_increment counts individual frames within a second.

by:

The top field prediction is based on the top field motion vector of the P-VOP macroblock of the future reference VOP. The past reference field is the reference field selected by the co-located macroblock of the future reference VOP for the top field. Analogously, the bottom field predictor is the average of pixels obtained from the future reference VOP's bottom field and the past reference field referenced by the bottom field motion vector of the corresponding macroblock of the future reference VOP. When interlaced direct mode is used, vop_time_increment_resolution must be the smallest integer greater than or equal to the number of frames per second. In each VOP, vop_time_increment counts individual frames within a second.

NOTE: The restriction in case of usage of interlaced direct mode is necessary, because the definitions of δ[i] in Table 7-16 and the equations above Table 7-16 do not allow normalization by Tframe.

Add a note to subclause A.1 to remind the IDCT mismatch propagation and scaling problem when quarter_sample flag is signalled. Add following text after NOTE 2:

NOTE 3: When ¼ pel motion compensation is used ("quarter sample mode", quarter_sample==1), IDCT mismatch may be scaled and propagated by the 8-tap FIR filtering. Even when a compliant IDCT is used, this possibly results in an obviously visible distortion after performing repeated filtering. The artifacts become serious when the QP value is small and fractional motion vectors are applied to both directions. Application of a shorter period of intra refresh may be necessary when these conditions are met.

In subclause B.1.1, clarify the meaning of MB type. Replace:

Table B-1 — Macroblock types and included data elements for I- and P-, and S-VOPs

VOP type	mb type	Name	not_coded	mcbpc	mcsel	cbpy	dquant	mvd	mvd_{2-4}
P	not coded	-	1						
P	0	inter	1	1		1		1	
P	1	inter+q	1	1		1	1	1	
P	2	inter4v	1	1		1		1	1
P	3	intra	1	1		1			
P	4	intra+q	1	1		1	1		
P	stuffing	-	1	1					
I	3	intra		1		1			
I	4	intra+q		1		1	1		
I	stuffing	-		1					
S (update)	not_coded	-	1						
S (update)	0	inter	1	1		1			
S (update)	1	inter+q	1	1		1	1		
S (update)	3	intra	1	1		1			
S (update)	4	intra+q	1	1		1	1		
S (update)	stuffing	-	1	1					
S (piece)	3	intra		1		1			
S (piece)	4	intra+q		1		1	1		
S (piece)	stuffing	-		1					
S (GMC)	not coded	-	1						
S (GMC)	0	inter	1	1	1	1		1	
S (GMC)	1	inter+q	1	1	1	1	1	1	
S(GMC)	2	inter4v	1	1		1		1	1
S (GMC)	3	intra	1	1		1			
S (GMC)	4	intra+q	1	1		1	1		
S (GMC)	stuffing	-	1	1					

NOTE "1" means that the item is present in the macroblock
S (piece) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "piece"
S (update) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "update"
S (GMC) indicates S-VOPs with sprite_enable == "GMC"

by:

Table B-1 — Macroblock types and included data elements for I- and P-, and S-VOPs

VOP type	MB type	Name	not_coded	mcbpc	mcsel	cbpy	dquant	mvd	mvd_{2-4}
P	not coded	-	1						
P	0	inter	1	1		1		1	
P	1	inter+q	1	1		1	1	1	
P	2	inter4v	1	1		1		1	1
P	3	intra	1	1		1			
P	4	intra+q	1	1		1	1		
P	stuffing	-	1	1					
I	3	intra		1		1			
I	4	intra+q		1		1	1		
I	stuffing	-		1					
S (update)	not_coded	-	1						
S (update)	0	inter	1	1		1			
S (update)	1	inter+q	1	1		1	1		
S (update)	3	intra	1	1		1			
S (update)	4	intra+q	1	1		1	1		
S (update)	stuffing	-	1	1					
S (piece)	3	intra		1		1			
S (piece)	4	intra+q		1		1	1		
S (piece)	stuffing	-		1					
S (GMC)	not coded	-	1						
S (GMC)	0	inter	1	1	1	1		1	
S (GMC)	1	inter+q	1	1	1	1	1	1	
S(GMC)	2	inter4v	1	1		1		1	1
S (GMC)	3	intra	1	1		1			
S (GMC)	4	intra+q	1	1		1	1		
S (GMC)	stuffing	-	1	1					

NOTE "1" means that the item is present in the macroblock
S (piece) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "piece"
S (update) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "update"
S (GMC) indicates S-VOPs with sprite_enable == "GMC"
The values of the column "MB type" in this table are used as the variable "derived_mb_type" which is used in the respective syntax part for motion and texture decoding.

Following the previous item, replace:

Table B-3 — VLC table for modb

Code	cbpb	mb_type
1		
01		1
00	1	1

by:

Table B-3 — VLC table for modb and included data elements

Code	cbpb	mb_type
1		
01		1
00	1	1

In subclause B.1.1, separate block numbers by comma in tables B-6 through B-8 for better readability. Replace Tables B-6 through B-8 by:

Table B-6 — VLC table for mcbpc for I-VOPs and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”piece”

Code	MB type	cbpc (4,5)
1	3	0,0
001	3	0,1
010	3	1,0
011	3	1,1
0001	4	0,0
0000 01	4	0,1
0000 10	4	1,0
0000 11	4	1,1
0000 0000 1	stuffing	--

Note: Numbers (4,5) refer to the macroblock structure in Figure 6-8.

Table B-7 — VLC table for mcbpc for P-VOPs and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”update”, and S(GMC)-VOPs

Code	MB type	cbpc (4,5)
1	0	0,0
0011	0	0,1
0010	0	1,0
0001 01	0	1,1
011	1	0,0
0000 111	1	0,1
0000 110	1	1,0
0000 0010 1	1	1,1
010	2	0,0
0000 101	2	0,1
0000 100	2	1,0
0000 0101	2	1,1
0001 1	3	0,0
0000 0100	3	0,1
0000 0011	3	1,0
0000 011	3	1,1
0001 00	4	0,0
0000 0010 0	4	0,1
0000 0001 1	4	1,0
0000 0001 0	4	1,1
0000 0000 1	stuffing	--

Note: Numbers (4,5) refer to the macroblock structure in Figure 6-8.

Table B-8 — VLC table for cbpy in the case of four non-transparent blocks

Code	cbpy(intra-MB) (0,1 2,3)	cbpy(inter-MB) (0,1 2,3)
0011	0,0 0,0	1,1 1,1
0010 1	0,0 0,1	1,1 1,0
0010 0	0,0 1,0	1,1 0,1
1001	0,0 1,1	1,1 0,0
0001 1	0,1 0,0	1,0 1,1
0111	0,1 0,1	1,0 1,0
0000 10	0,1 1,0	1,0 0,1
1011	0,1 1,1	1,0 0,0
0001 0	1,0 0,0	0,1 1,1
0000 11	1,0 0,1	0,1 1,0
0101	1,0 1,0	0,1 0,1
1010	1,0 1,1	0,1 0,0
0100	1,1 0,0	0,0 1,1
1000	1,1 0,1	0,0 1,0
0110	1,1 1,0	0,0 0,1
11	1,1 1,1	0,0 0,0

Note: Numbers (0,1,2,3) refer to the macroblock structure in Figure 6-8.

In Tables B-13 through B-15, correct allowed codes in 8-bit mode. Replace below Table B-13:

NOTE: The variable length code for dct_dc_size_luminance of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

by:

NOTE: The variable length code for dct_dc_size_luminance of 9, 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

Following the previous item, replace below Table B-14:

NOTE: The variable length code for dct_dc_size_chrominance of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

by:

NOTE: The variable length code for dct_dc_size_chrominance of 9, 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

Following the previous item, replace below Table B-15:

NOTE1: The variable length code for “Size” of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

by:

NOTE1: The variable length code for “Size” of 9, 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

In subclause B.1.6, clarify the meaning of SSS and VLC in Table B-34. Replace:

dmv value	SSS	VLC	dmv_code

by:

dmv value	dmv length	dmv_length code	dmv_code

In Annex D.2, correct typos. Replace:

7. τ_i is the composition time (or presentation time in a no-compositor decoder) of VOP *i*. For a video object plane, τ_i defined by vop_time_increment (in units of 1⁄vop_time_increment_resolution seconds) plus the cumulative number of whole seconds specified by module_time_base. In the case of interlaced video, a VOP consists of lines from two fields and τ_i is the composition time of the first field. The relationship between the composition time and the decoding time for a VOP is given by:

by:

7. τ_i is the composition time (or presentation time in a no-compositor decoder) of VOP *i*. For a video object plane, τ_i defined by vop_time_increment (in units of 1⁄vop_time_increment_resolution seconds) plus the cumulative number of whole seconds specified by modulo_time_base. In the case of interlaced video, a VOP consists of lines from two fields and τ_i is the composition time of the first field. The relationship between the composition time and the decoding time for a VOP is given by:

In subclause E.1.2, clarify the data partitioning for I-VOP. Replace:

In recognizing the need to provide enhanced concealment capabilities, the Video Group has developed an additional error resilient mode that further improves the ability of the decoder to localize an error. Specifically, this approach utilizes data partitioning. This data partitioning is achieved by separating the motion and macroblock header information away from the texture information. This approach requires that a second resynchronization marker be inserted between motion and texture information. Data partitioning, like the use of RVLCs, is signaled to the decoder in the VOL. Figure E-2 illustrates the syntactic structure of the data partitioning mode. If the texture information is lost, this approach utilizes the motion information to conceal these errors. That is, due to the errors the texture information is discarded, while the motion is used to motion compensate the previously decoded VOP.

by:

In recognizing the need to provide enhanced concealment capabilities, the Video Group has developed an additional error resilient mode that further improves the ability of the decoder to localize an error. Specifically, this approach utilizes data partitioning. This data partitioning is achieved by separating DC (which is coded by DC VLC) and header information away from the other texture information for I-VOP and separating the motion and macroblock header information away from the texture information for P-VOP. This approach requires that a second resynchronization marker be inserted between DC or motion and texture information. Data partitioning, like the use of RVLCs, is signaled to the decoder in the VOL. Figure E-2 illustrates the syntactic structure of the data partitioning mode. If the texture information is lost, this approach utilizes the DC or motion information to conceal these errors. That is, due to the errors the texture information is discarded, while the DC or motion is used to provide the block approximation or motion compensate the previously decoded VOP. To make data partitioning perform well, it is suggested that intra_dc_vlc_thr is set to 0 for I-VOP coded in the data partitioning mode.

Following the previous item, replace:

Resync Marker	**macroblock_number**	**quant_scale**	**HEC**	**Motion &Header Information**	**Motion Marker**	**Texture Information**	**Resync Marker**

Figure E-2 — Data Partitioning

by:

Resync Marker	**macroblock_number**	**quant_scale**	**HEC**	**DC (in DC VLC) &Header Information**	**DC Marker**	**Texture Information**	**Resync Marker**

(a)

Resync Marker	**macroblock_number**	**quant_scale**	**HEC**	**Motion &Header Information**	**Motion Marker**	**Texture Information**	**Resync Marker**

(b)

Figure E-2 — Data Partitioning (a) I-VOP (b) P-VOP

In Table G-1, change ESCAPE code reservations in profile_and_level_indication. Replace:

Reserved	01110011 – 10000000

by:

Reserved	01110011 – 01111110
Reserved for Systems use	01111111
Reserved for Escape	10000000

Following the previous item, replace:

Reserved	11111110
Reserved for Escape	11111111

by:

Reserved for Systems use	11111110
Reserved for Systems use	11111111