INTERNATIONAL STANDARD ISO/IEC 14496-2:1999
TECHNICAL CORRIGENDUM 1

Published 2000-08-15

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 International Standard ISO/IEC 14496-2:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ICS 35.040 **Ref. No. ISO/IEC 14496-2:1999/Cor.1:2000(E)**

Throughout the whole document, replace "quantization" with "quantisation".

On Page xiii, Overview of the object based nonscalable syntax, replace the following paragraph:

"

The coded representation defined in the non-scalable syntax achieves a high compression ratio while preserving good image quality. Further, when access to individual objects is desired, the shape of objects also needs to be coded, and depending on the bandwidth available, the shape information can be coded lossy or losslessly.

"

with

"

The coded representation defined in the non-scalable syntax achieves a high compression ratio while preserving good image quality. Further, when access to individual objects is desired, the shape of objects also needs to be coded, and depending on the bandwidth available, the shape information can be coded in a lossy or lossless fashion.

"

In Subclause 5.1, Method of describing bitstream syntax, replace the following table:

"

while (condition) {
data_element
. . .
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do { **data_element** . . . } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.

if (condition) {
data_element
. . .
} else {
data_element
. . .
}
	If the condition is true, then the first group of data elements occurs next in the data stream.
	
	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (i = m; i < n; i++) {
data_element
. . .
}
	The group of data elements occurs (n-m) times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to m for the first occurrence, incremented by one for the second occurrence, and so forth.

/* comment . . . */	Explanatory comment that may be deleted entirely without in any way altering the syntax.

"

with

"

while (condition) {
data_element
. . .
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do { **data_element** . . . } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.
do {	
. . .	
continue	The continue continues execution of the next repetition of the nearest while-do loop.
. . .	

} while (condition)	

if (condition) {
    data_element
    . . .
} else {
    data_element
    . . .
}
	If the condition is true, then the first group of data elements occurs next in the data stream.
	
	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (i = m; i < n; i++) {
    data_element
    . . .
}
	The group of data elements occurs (n-m) times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to m for the first occurrence, incremented by one for the second occurrence, and so forth.

/* comment . . . */	Explanatory comment that may be deleted entirely without in any way altering the syntax.

"

In Subclause 5.2.4, Definition of next_start_code() function, delete the following paragraph:

"

This function checks whether the current position is byte aligned. If it is not, a zero stuffing bit followed by a number of one stuffing bits may be present before the start code.

"

In Clause 3, Definitions, add the following definitions with the appropriate numbering in alphabetical order:

"

3.xxx **mesh object planes**, **MOP:** The instance of mesh objects at a given time.

3.xxx **video object planes**, **VOP:** The instance of video objects at a given time.

"

In Clause 3, Definitions, remove the following definition:

"

3.3 **backward compatibility**: A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.

"

In Table 6-3, replace the following row 5 (left column):

"

visual_object_sequence__start_code

"

with

"

visual_object_sequence_start_code

"

In Subclause 6.2.1, Start Codes, replace the following paragraph:

"

When coded visual objects are carried within a Systems bitstream defined by ISO/IEC 14496-1, configuration information and elementary stream data are always carried separately. Configuration information and elementary streams follow the syntax below, subject to the break points between them defined above. The Systems specification ISO/IEC 14496-1 defines containers that are used to carry Visual Object and Visual Object Layer configuration information. A separate container is used for each object. For video objects, a separate container is also used for each layer. VisualObjectSequence headers are not carried explicitly, but the information is contained in other parts of the Systems bitstream.

"

with
"
When coded visual objects are carried within a Systems bitstream defined by ISO/IEC 14496-1, configuration information and elementary stream data are always carried separately. Configuration information and elementary streams follow the syntax below, subject to the break points between them defined above. The Systems specification ISO/IEC 14496-1 defines containers that are used to carry Visual Object Sequence, Visual Object and Video Object Layer configuration information. For video objects one container is used for each layer for each object. This container carries a Visual Object Sequence header, a Visual Object header and a Video Object Layer header. For other types of visual objects, one container per visual object is used. This container carries a Visual Object Sequence header and a Visual Object header. The Visual Object Sequence Header must be identical for all visual streams input simultaneously to a decoder. The Visual Object Headers for each layer of a multilayer object must be identical.
"

In Subclause 6.2.1, Start Codes, replace the following paragraph:
"

The elementary stream data associated with a single layer may be wrapped in configuration information defined in accordance with the syntax below. A visual bitstream may contain at most one instance of each of VisualObjectSequence(), VisualObject() and VideoObjectLayer(). The Visual Object Sequence Header must be identical for all streams input simultaneously to a decoder. The Visual Object Headers for each layer of a multilayer object must be identical.
"
with
"
The elementary stream data associated with a single layer may be wrapped in configuration information defined in accordance with the syntax below. A visual bitstream may contain at most one instance of each of VisualObjectSequence(), VisualObject() and VideoObjectLayer(), with the exception of repetition of the Visual Object Sequence Header, the Visual Object Header and the Video Object Layer Header as described below. The Visual Object Sequence Header must be identical for all visual streams input simultaneously to a decoder. The Visual Object Headers for each layer of a multilayer object must be identical. 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.
"

In Subclause 6.2.2, Visual Object Sequence and Visual Object, replace the VisualObjectSequence() syntax:
"

VisualObjectSequence() {	**No. of bits**	**Mnemonic**
visual_object_sequence_start_code	32	bslbf
profile_and_level_indication	8	uimsbf
while (next_bits()== user_data_start_code){		
user_data()		
}		
VisualObject()		
visual_object_sequence_end_code	32	bslbf
}		

"

with

"

VisualObjectSequence() {	**No. of bits**	**Mnemonic**
do {		
visual_object_sequence_start_code	32	bslbf
profile_and_level_indication	8	uimsbf
while (next_bits()== user_data_start_code){		
user_data()		
}		
VisualObject()		
} while (next_bits() != visual_object_sequence_end_code)		
visual_object_sequence_end_code	32	bslbf
}		

"

In Subclause 6.2.2.1, User data(), replace the user_data() syntax:

"

user_data() {	**No. of bits**	**Mnemonic**
user_data_start_code	32	bslbf
while(next_bits() != ‘0000 0000 0000 0000 0000 0001’) {		
user_data	8	uimsbf
}		
next_start_code()		
}		

"

with

"

user_data() {	**No. of bits**	**Mnemonic**
user_data_start_code	32	bslbf
while(next_bits() != ‘0000 0000 0000 0000 0000 0001’) {		
user_data	8	uimsbf
}		
}		

"

In Subclause 6.2.4, Group of Video Object Plane, in conformance to Table 6-3 replace the following row 2:

“

group_vop_start_codes	32	bslbf

"

with

“

group_of_vop_start_code	32	bslbf

"

In Subclause 6.2.5, Video Object Plane and Video Plane with Short Header, replace the following rows 18 to 26 of the VideoObjectPlane() syntax:

"

if(!(sprite_enable && vop_coding_type == “I”)) {		
vop_width	13	uimsbf
marker_bit	1	bslbf
vop_height	13	uimsbf
marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	13	simsbf
}		

"

with

"

if(!(sprite_enable && vop_coding_type == “I”)) {		
vop_width	13	uimsbf
marker_bit	1	bslbf
vop_height	13	uimsbf
marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
}		

"

In Subclause 6.2.5, replace the following rows 33 to 35 of the VideoObjectPlane() syntax:

“

}		
if (!complexity_estimation_disable)		
read_vop_complexity_estimation_header()		

"

with

“

}		
if (video_object_layer_shape != ‘’binary only‘’)		
if (!complexity_estimation_disable)		
read_vop_complexity_estimation_header()		

"

In Subclause 6.2.5, Video Object Plane and Video Plane with Short Header, replace row 44 of the VideoObjectPlane() syntax:

"

if (no_sprite_points > 0)		

"

with

"

if (no_of_sprite_warping_points > 0)		

"

In Subclause 6.2.5.2, Video Plane with Short Header, replace the video_packet_header() syntax:

"

video_packet_header() {	**No. of bits**	**Mnemonic**
next_resync_marker()		
resync_marker	17-23	uimsbf
macroblock_number	1-14	vlclbf
if (video_object_layer_shape != “binary only”)		
quant_scale	5	uimsbf
header_extension_code	1	bslbf
if (header_extension_code) {		
do {		
modulo_time_base	1	bslbf
} while (modulo_time_base != ‘0’)		
marker_bit	1	bslbf
vop_time_increment	1-16	bslbf
marker_bit	1	bslbf
vop_coding_type	2	uimsbf
if (video_object_layer_shape != “binary only”) {		
intra_dc_vlc_thr	3	uimsbf
if (vop_coding_type != “I”)		
vop_fcode_forward	3	uimsbf
if (vop_coding_type == “B”)		
vop_fcode_backward	3	uimsbf
}		
}		
}		

"

with

"

video_packet_header() {	**No. of bits**	**Mnemonic**
next_resync_marker()		
resync_marker	17-23	uimsbf
if (video_object_layer_shape != “rectangular”) {		
header_extension_code	1	bslbf
if (header_extension_code && !(sprite_enable && vop_coding_type == “I”)) {		
vop_width	13	uimsbf
marker_bit	1	bslbf
vop_height	13	uimsbf
marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	13	simsbf
marker_bit	1	bslbf
}		
}		
macroblock_number	1-14	vlclbf
if (video_object_layer_shape != “binary only”)		
quant_scale	5	uimsbf
if (video_object_layer_shape == “rectangular”)		

header_extension_code	1	bslbf
if (header_extension_code) {		
do {		
modulo_time_base	1	bslbf
} while (modulo_time_base != '0')		
marker_bit	1	bslbf
vop_time_increment	1-16	bslbf
marker_bit	1	bslbf
vop_coding_type	2	uimsbf
if (video_object_layer_shape != "rectangular") {		
change_conv_ratio_disable	1	bslbf
if (vop_coding_type != "I")		
vop_shape_coding_type	1	bslbf
}		
if (video_object_layer_shape != "binary only") {		
intra_dc_vlc_thr	3	uimsbf
if (vop_coding_type != "I")		
vop_fcode_forward	3	uimsbf
if (vop_coding_type == "B")		
vop_fcode_backward	3	uimsbf
}		
}		
}		

"

In Subclause 6.2.5.3, Motion Shape Texture, replace the following rows 11, 12 and 13 of data_patitioned_i_vop() syntax:

"

if (!transparent_mb()) {		
mcbpc	1-9	vlclbf
if (mb_type == 4)		

"

with

"

If (!transparent_mb()) {		
if(video_object_layer_shape != "rectangle"){		
do{		
mcbpc	1-9	vlclbf
} while(derived_mb_type == "stuffing")		
}else{		
mcbpc	1-9	vlclbf
if(derived_mb_type == "stuffing")		
continue		
}		
if (mb_type == 4)		

"

In Subclause 6.2.5.3, Motion Shape Texture, replace the Note at the end of data_patitioned_i_vop() syntax:

"

NOTE The value of block_count is 6 in the 4:2:0 format. The value of alpha_block_count is 4.

"

with

"

NOTE 1 — The value of mb_in_video_packet is the number of macroblocks in a video packet. The count of stuffing macroblocks is not included in this value. NOTE 2 — The value of block_count is 6 in the 4:2:0 format. NOTE 3 — The value of alpha_block_count is 4.

"

In Subclause 6.2.5.3, Motion Shape Texture, replace the following rows 15 to 22 of the data_patitioned_p_vop() syntax:

"

if (!transparent_mb()) {		
not_coded	1	bslbf
if (!not_coded) {		
mcbpc	1-9	vlclbf
if (derived_mb_type < 3)		
motion_coding(“forward”, derived_mb_type)		
}		
}		

"

with

"

if (!transparent_mb()) {		
if(video_object_layer_shape != “rectangle”){		
do{		
not_coded	1	bslbf
if (!not_coded)		
mcbpc	1-9	vlclbf
} while(!(not_coded || derived_mb_type != “stuffing”))		
}else{		
not_coded	1	bslbf
if (!not_coded){		
mcbpc	1-9	vlclbf
if(derived_mb_type == “stuffing”)		
continue		
}		
}		
if (!not_coded) {		
if (derived_mb_type < 3)		
motion_coding(“forward”, derived_mb_type)		
}		
}		

"

In Subclause 6.2.5.3, Motion Shape Texture, replace the Note at the end of data_patitioned_p_vop() syntax:

"

NOTE The value of block_count is 6 in the 4:2:0 format. The value of alpha_block_count is 4.

"

with

"

NOTE 1 — The value of mb_in_video_packet is the number of macroblocks in a video packet. The count of stuffing macroblocks is not included in this value. NOTE 2 — The value of block_count is 6 in the 4:2:0 format. NOTE 3 — The value of alpha_block_count is 4.

"

In Subclause 6.2.6, Macroblock, replace the following rows 6 to 11 of the macroblock() syntax:

"

if (!transparent_mb()) {		
if (vop_coding_type != "I" && !(sprite_enable && sprite_transmit_mode == "piece"))		
not_coded	1	bslbf
if (!not_coded || vop_coding_type == "I") {		
mcbpc	1-9	vlclbf
if (!short_video_header && (derived_mb_type == 3 || derived_mb_type == 4))		

"

with

"

if (!transparent_mb()) {		

if (video_object_layer_shape != "rectangular"
&& !(sprite_enable && low_latency_sprite_enable
&& sprite_transmit_mode == "update")) {		

do{		
if (vop_coding_type != "I" && !(sprite_enable && sprite_transmit_mode == "piece"))		
not_coded	1	bslbf
if (!not_coded || vop_coding_type == "I" || (vop_coding_type == "S" && low_latency_sprite_enable && sprite_transmit_mode == "piece"))		
mcbpc	1-9	vlclbf
} while(!(not_coded || derived_mb_type != "stuffing"))		
} else {		
if (vop_coding_type != "I" && !(sprite_enable && sprite_transmit_mode == "piece"))		
not_coded	1	bslbf
if (!not_coded || vop_coding_type == "I" || (vop_coding_type == "S" && low_latency_sprite_enable && sprite_transmit_mode == "piece"))		
mcbpc	1-9	vlclbf
}		

if (!not_coded || vop_coding_type == "I"
|| (vop_coding_type == "S"
&& low_latency_sprite_enable
&& sprite_transmit_mode == "piece")) {		

if (!short_video_header && (derived_mb_type == 3 || derived_mb_type == 4))		

"

In Subclause 6.2.8, Still Texture Object, replace the StillTextureObject() syntax:

"

StillTextureObject() {	**No. of bits**	**Mnemonic**
still_texture_object_start_code	32	
texture_object_id	16	uimsbf
marker_bit	1	bslbf
wavelet_filter_type	1	uimsbf
wavelet_download	1	uimsbf
wavelet_decomposition_levels	4	uimsbf
scan_direction	1	bslbf
start_code_enable	1	bslbf
texture_object_layer_shape	2	uimsbf
quantization_type	2	uimsbf
if (quantization_type == 2) {		
spatial_scalability_levels	4	uimsbf
if (spatial_scalability_levels != wavelet_decomposition_levels) {		
use_default_spatial_scalability	1	uimsbf
if (use_default_spatial_layer_size == 0)		
for (i=0; i<spatial_scalability_levels – 1; i++)		
wavelet_layer_index	4	
}		
}		
if (wavelet_download == "1"){		
uniform_wavelet_filter	1	uimsbf
if (uniform_wavelet_filter == "1")		
download_wavelet_filters()		
else		
for (I=0; i<wavelet_decomposition_levels; i++)		
download_wavelet_filters( )		
}		
wavelet_stuffing	3	uimsbf
if(texture_object_layer_shape == "00"){		
texture_object_layer_width	15	uimsbf
marker_bit	1	bslbf
texture_object_layer_height	15	uimsbf
marker_bit	1	bslbf
}		
else {		
horizontal_ref	15	imsbf
marker_bit	1	bslbf
vertical_ref	15	imsbf

marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf
shape_object_decoding ( )		
}		
for (color = “y”, “u”, “v”)		
wavelet_dc_decode()		
if(quantization_type == 1)		
TextureLayerSQ ( )		
else if (quantization_type == 2){		
if (start_code_enable == 1) {		
do {		
TextureSpatialLayerMQ ( )		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i =0; i<spatial_scalability_levels; i++)		
TextureSpatialLayerMQNSC ( )		
}		
}		
else if (quantization_type == 3){		
for (color = “y”, “u”, “v”)		
do{		
quant_byte		
} while(quant_byte >>7)		
max_bitplanes		
if (scan_direction == 0) {		
do {		
TextureSNRLayerBQ ( )		
} while (next_bits() == texture_snr_layer_start_code)		
} else {		
do {		
TextureSpatialLayerBQ ( )		
} while (next_bits() == texture_spatial_layer_start_code)		
}		
}		
}		

"

with

"

StillTextureObject() {	**No. of bits**	**Mnemonic**
still_texture_object_start_code	32	
texture_object_id	16	uimsbf
marker_bit	1	bslbf
wavelet_filter_type	1	uimsbf
wavelet_download	1	uimsbf
wavelet_decomposition_levels	4	uimsbf
scan_direction	1	bslbf
start_code_enable	1	bslbf

texture_object_layer_shape	2	uimsbf
quantization_type	2	uimsbf
if (quantization_type == 2) {		
spatial_scalability_levels	4	uimsbf
if (spatial_scalability_levels != wavelet_decomposition_levels) {		
use_default_spatial_scalability	1	uimsbf
if (use_default_spatial_layer_size == 0)		
for (i=0; i<spatial_scalability_levels – 1; i++)		
wavelet_layer_index	4	
}		
}		
if (wavelet_download == “1”){		
uniform_wavelet_filter	1	uimsbf
if (uniform_wavelet_filter == “1”)		
download_wavelet_filters()		
else		
for (i=0; i<wavelet_decomposition_levels; i++)		
download_wavelet_filters( )		
}		
wavelet_stuffing	3	uimsbf
if(texture_object_layer_shape == “00”){		
texture_object_layer_width	15	uimsbf
marker_bit	1	bslbf
texture_object_layer_height	15	uimsbf
marker_bit	1	bslbf
}		
else {		
horizontal_ref	15	imsbf
marker_bit	1	bslbf
vertical_ref	15	imsbf
marker_bit	1	bslbf
object_width	15	uimsbf
marker_bit	1	bslbf
object_height	15	uimsbf
marker_bit	1	bslbf
shape_object_decoding ( )		
}		
for (color = “y”, “u”, “v”)		
wavelet_dc_decode()		
if(quantization_type == 1)		
TextureLayerSQ ( )		
else if (quantization_type == 2){		
if (start_code_enable == 1) {		
do {		
TextureSpatialLayerMQ ( )		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i =0; i<spatial_scalability_levels; i++)		
TextureSpatialLayerMQNSC ( )		
}		

}		
else if (quantization_type == 3){		
for (color = “y”, “u”, “v”)		
do{		
quant_byte		
} while(quant_byte >>7)		
max_bitplanes		
if (scan_direction == 0) {		
do {		
TextureSNRLayerBQ ( )		
} while (next_bits() == texture_snr_layer_start_code)		
} else {		
do {		
TextureSpatialLayerBQ ( )		
} while (next_bits() == texture_spatial_layer_start_code)		
}		
}		
}		

"

In Subclause 6.2.8.1, replace the TextureLayerSQ() syntax:

"

TextureLayerSQ() {	**No. of bits**	**Mnemonic**
if (scan_direction == 0) {		
for (“y”, “u”, “v”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<wavelet_decomposition_levels; i++)		
if (i!=0 || color!= “u“,“v“) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4==0)		
marker_bit	1	bslbf
}		
}		
for (i = 0; i<tree_blocks; i++)		
for (color = “y”, “u”, “v”)		
arith_decode_highbands_td()		
} else {		
if (start_code_enable) {		
do {		
TextureSpatialLayerSQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i = 0; i< wavelet_decomposition_levels; i++)		
TextureSpatialLayerSQNSC()		
}		
}		
}		

"

with

"

TextureLayerSQ() {	**No. of bits**	**Mnemonic**
if (scan_direction == 0) {		
for (“y”, “u”, “v”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<wavelet_decomposition_levels; i++)		
if (i!=0 || color!= “u“,“v“) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4==0)		
marker_bit	1	bslbf
}		
}		
for (i = 0; i<tree_blocks; i++)		
for (color = “y”, “u”, “v”)		
arith_decode_highbands_td()		
} else {		
if (start_code_enable) {		
do {		
TextureSpatialLayerSQ()		
} while (next_bits() == texture_spatial_layer_start_code)		
} else {		
for (i = 0; i< wavelet_decomposition_levels; i++)		
TextureSpatialLayerSQNSC()		
}		
}		
}		
NOTE — The value of tree_block is that wavelet coefficients are organized in a tree structure which is rooted in the low-low band (DC band) of the wavelet decomposition, then extends into the higher frequency bands at the same spatial location. Note the DC band is encoded separately.		

"

In Subclause 6.2.8.3, replace the TextureSpatialLayerSQNSC() syntax:

"

TextureSpatialLayerSQNSC() {	**No. of bits**	**Mnemonic**
for (color=“y“,“u“,“v“) {		
if ((first_wavelet_layer && color==“y“) || (second_wavelet_layer && color==“u“,“v“))		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
if (color ==“y“)		
max_bitplanes	5	uimbsf
else if (!first_wavelet_layer)		
max_bitplanes	5	uimbsf
}		
arith_decode_highbands_bb()		
}		

"

with

"

TextureSpatialLayerSQNSC() {	No. of bits	Mnemonic
for (color=“y“,“u“,“v“) {		
if ((first_wavelet_layer && color==“y“) || (second_wavelet_layer && color==“u“,“v“))		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
if (color ==“y“)		
max_bitplanes	5	uimbsf
else if (!first_wavelet_layer)		
max_bitplanes	5	uimbsf
}		
for (color="y","u","v")		
if (color="y" || !first_wavelet_layer)		
arith_decode_highbands_bb()		
}		
NOTE — The value of first_wavelet_layer becomes “true” when the variable ‘i’ of Subclause 6.2.8.1 TextureLayerSQ() equals to zero. Otherwise, it is “false”. The value of second_wavelet_layer becomes “true” when the variable ‘i’ of Subclause 6.2.8.1 TextureLayerSQ() equals to one. Otherwise, it is “false”.		

"

In Subclause 6.2.8.7, replace the TextureSNRLayerMQNSC() syntax:

"

TextureSNRLayerMQNSC(){	No. of bits	Mnemonic
if (spatial_scalability_levels == wavelet_decomposition_levels && spatial_layer_id == 0) {		
for (color = “y“) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
else {		
for (color=“y”, “u”, “v”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		

}		
if (scan_direction == 0) {		
for (i = 0; i<tree_blocks; i++)		
for (color = “y”, “u”, “v”)		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_td()		
} else {		
for (i = 0; i< spatial_layers; i++) {		
for (color = “y”, “u”, “v”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_bb()		
}		
}		
}		
}		

"

with

"

TextureSNRLayerMQNSC(){	No. of bits	Mnemonic
if (spatial_scalability_levels == wavelet_decomposition_levels && spatial_layer_id == 0) {		
for (color = “y“) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
else {		
for (color=“y”, “u”, “v”) {		
do {		
quant_byte	8	uimsbf
} while (quant_byte >> 7)		
for (i=0; i<spatial_layers; i++) {		
max_bitplane[i]	5	uimsbf
if ((i+1)%4 == 0)		
marker_bit	1	bslbf
}		
}		
}		
if (scan_direction == 0) {		
for (i = 0; i<tree_blocks; i++)		
for (color = “y”, “u”, “v”)		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_td()		
} else {		

for (i = 0; i< spatial_layers; i++) {		
for (color = “y”, “u”, “v”) {		
if (wavelet_decomposition_layer_id != 0 || color != “u”, “v”)		
arith_decode_highbands_bb()		
}		
}		
}		
}		
NOTE — The value of spatial_layers is equivalent to the maximum number of the wavelet decomposition layers in that scalability layer.		

"

In Subclause 6.2.8.11, DownloadWaveletFilters, replace the download_wavelet_filters syntax:

“

download_wavelet_filters( ){	No. of bits	Mnemonic
lowpass_filter_length	4	uimsbf
highpass_filter_length	4	uimsbf
do{		
if (wavelet_filter_type == 0) {		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		
} while (lowpass_filter_length--)		
do{		
if (wavelet_filter_type == 0){		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		
} while (highpass_filter_length--)		
if (wavelet_filter_type == 0) {		
integer_scale	16	uimsbf
marker_bit		
}		
}		

"

with

“

download_wavelet_filters( ){	No. of bits	Mnemonic
lowpass_filter_length	4	uimsbf
highpass_filter_length	4	uimsbf

do{		
if (wavelet_filter_type == 0) {		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		
} while (lowpass_filter_length--)		
do{		
if (wavelet_filter_type == 0){		
filter_tap_integer	16	imsbf
marker_bit	1	bslbf
} else {		
filter_tap_float_high	16	uimsbf
marker_bit	1	bslbf
filter_tap_float_low	16	uimsbf
marker_bit	1	bslbf
}		
} while (highpass_filter_length--)		
if (wavelet_filter_type == 0) {		
integer_scale	16	uimsbf
marker_bit	1	bslbf
}		
}		

"

In Subclause 6.3.3, Semantics of the low_delay, replace the following:

"

low_delay : This is a one-bit flag which when set to '1' indicates the VOL contains no B-VOPs.

"

with

"

low_delay : This is a one-bit flag which when set to '1' indicates the VOL contains no B-VOPs. If this flag is not present in the bitstream, the default value is 0 for visual object types that support B-VOP otherwise it is 1

"

In Subclause 6.3.3, Video Object Layer, replace the following:

"

sprite_left_coordinate – This is a 13-bit signed integer which defines the left-edge of the sprite. The value of sprite_left_coordinate shall be divisible by two.
sprite_top_coordinate: This is a 13-bit signed integer which defines the top edge of the sprite. The value of sprite_left_coordinate shall be divisible by two.

"

with

"

sprite_left_coordinate – This is a 13-bit signed integer which defines the left edge of the sprite. The value of sprite_left_coordinate shall be divisible by two.
sprite_top_coordinate: This is a 13-bit signed integer which defines the top edge of the sprite. The value of sprite_top_coordinate shall be divisible by two.

"

in Subclause 6.3.3, Video Object Layer, replace table 6-16:

"

Table 6-16 Number of point and implied warping function

Number of points	warping function
0	Stationary
1	Translation
2,3	Affine
4	Perspective

"

with

"

Table 6-16 Number of points and implied warping function

Number of points	warping function
0	Stationary
1	Translation
2,3	Affine
4	Perspective
5-63	Reserved

"

In Subclause 6.3.3, Video Object Layer, replace the following paragraph:

"

first_half_vbv_occupancy, latter_half_vbv_occupancy: The vbv_occupancy is a 26-bit unsigned integer. This value is divided to two parts. The most significant bits are in first_half_vbv_occupancy (11 bits) and the least significant bits are in latter_half_vbv_occupancy (15 bits). The marker_bit is inserted between the first_vbv_buffer_size and the latter_half_vbv_buffer_size in order to avoid the resync_marker emulation. The value of this integer is the VBV occupancy in 64-bit units just before the removal of the first VOP following the VOL header. The purpose for the quantity is to provide the initial condition for VBV buffer fullness.

"

with

"

first_half_vbv_occupancy, latter_half_vbv_occupancy: The vbv_occupancy is a 26-bit unsigned integer. This value is divided to two parts. The most significant bits are in first_half_vbv_occupancy (11 bits) and the least significant bits are in latter_half_vbv_occupancy (15 bits). The marker_bit is inserted between the first_half_vbv_occupancy and the latter_half_vbv_occupancy in order to avoid the resync_marker emulation. The value of this integer is the VBV occupancy in 64-bit units just before the removal of the first VOP following the VOL header. The purpose for the quantity is to provide the initial condition for VBV buffer fullness.

"

In Subclause 6.3.3, Video Object Layer, replace the following paragraph:

"

not_8_bit: This one bit flag is set when the video data precision is not 8 bits per pixel.

"

with

"

not_8_bit: This one bit flag is set when the video data precision is not 8 bits per pixel and visual object type is N-bit.

"

In Subclause 6.3.3, Video Object Layer, replace the following paragraphs:

"

video_object_layer_width: The video_object_layer_width is a 13-bit unsigned integer representing the width of the displayable part of the luminance component in pixel units. The width of the encoded luminance component of VOPs in macroblocks is (video_object_layer_width+15)/16. The displayable part is left-aligned in the encoded VOPs.

video_object_layer_height: The video_object_layer_height is a 13-bit unsigned integer representing the height of the displayable part of the luminance component in pixel units. The height of the encoded luminance component of VOPs in macroblocks is (video_object_layer_height+15)/16. The displayable part is top-aligned in the encoded VOPs.
"

with

"

video_object_layer_width: The video_object_layer_width is a 13-bit unsigned integer representing the width of the displayable part of the luminance component in pixel units. The width of the encoded luminance component of VOPs in macroblocks is (video_object_layer_width+15)/16. The displayable part is left-aligned in the encoded VOPs. A zero value is forbidden.

video_object_layer_height: The video_object_layer_height is a 13-bit unsigned integer representing the height of the displayable part of the luminance component in pixel units. The height of the encoded luminance component of VOPs in macroblocks is (video_object_layer_height+15)/16. The displayable part is top-aligned in the encoded VOPs. A zero value is forbidden.
"

In Subclause 6.3.3, Video Object Layer, replace the following:

"

ref_layer_sampling_direc: This is a one-bit flag which when set to '1' indicates that the resolution of the reference layer (specified by reference_layer_id) is higher than the resolution of the layer being coded. If it is set to '0' then the reference layer has the same or lower resolution then the resolution of the layer being coded.
"

with

"

ref_layer_sampling_direc: This is a one-bit flag which when set to '1' indicates that the resolution of the reference layer (specified by reference_layer_id) is higher than the resolution of the layer being coded. If it is set to '0' then the reference layer has the same or lower resolution than the resolution of the layer being coded.
"

In Subclause 6.3.4, Group of Video Object Plane, replace the following:

"

group_vop_start_code: The group_vop_start_code is the bit string '000001B3' in hexadecimal. It identifies the beginning of a GOV header.
"

with

"

group_of_vop_start_code: The group_of_vop_start_code is the bit string '000001B3' in hexadecimal. It identifies the beginning of a GOV header.
"

In Subclause 6.3.5, Video Object Layer, replace the following:

"

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 rule applies: For an arbitrarily shaped VO (i.e. when the shape type of the VO is either 'binary' or 'binary only'), the alpha plane of the reconstructed VOP shall be completely transparent. For a rectangular VO (i.e. when the shape type of the VO is 'rectangular'), the corresponding rectangular alpha plane of the VOP, having the same size as its luminance component, shall be completely transparent. If there is no alpha plane being used in the decoding and composition process of a rectangular VO, the reconstructed VOP is filled with the respective content of the immediately preceding VOP for which vop_coded!=0.
"

with

"

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 clause 7.6.7.
"

In Subclause 6.3.5, Video Object Plane and Video Plane with Short Header, replace the following paragraph:

"

vop_shape_coding_type: This is a 1 bit flag which specifies whether inter shape decoding is to be carried out for the current P VOP. If vop_shape_coding_type is equal to ‘0’, intra shape decoding is carried out, otherwise inter shape decoding is carried out.

"

with

"

vop_shape_coding_type: This is a 1 bit flag which specifies whether inter shape decoding is to be carried out for the current P- or B-VOP. If vop_shape_coding_type is equal to ‘0’, intra shape decoding is carried out, otherwise inter shape decoding is carried out.

"

In Subclause 6.3.5.2, Video Plane with Short Header, replace the following paragraph:

"

num_macroblocks_in_gob: This is the number of macroblocks in each group of blocks (GOB) unit. This parameter is derived from the source_format as shown in Table 6-25.

"

with

"

num_macroblocks_in_gob: This is the number of macroblocks in each group of blocks (GOB) unit. This parameter is derived from the source_format as shown in Table 6-25. The count of stuffing macroblocks is not included in this value.

"

In Subclause 6.3.5.4, Sprite coding, replace the following paragraph:

"

send_mb(): This function returns 1 if the current macroblock has already been sent previously and “not coded”. Otherwise it returns 0.

"

with

"

send_mb(): This function returns 1 if the current macroblock has already been transmitted. Otherwise it returns 0.

"

In Subclause 6.3.6, Macroblock related, replace the following paragraph:

"

not_coded: This is a 1-bit flag which signals if a macroblock is coded or not. When set to’1’ it indicates that a macroblock is not coded and no further data is included in the bitstream for this macroblock; decoder shall treat this macroblock as ‘inter’ with motion vector equal to zero and no DCT coefficient data. When set to ‘0’ it indicates that the macroblock is coded and its data is included in the bitstream.

"

with

"

not_coded: This is a 1-bit flag which signals if a macroblock is coded or not. When set to’1’ it indicates that a macroblock is not coded and no further data is included in the bitstream for this macroblock (with the exception of alpha data that may be present). The decoder shall treat this macroblock as ‘inter’ with motion vector equal to zero and no DCT coefficient data. When set to ‘0’ it indicates that the macroblock is coded and its data is included in the bitstream.

"

In Subclause 6.3.8, Still texture object, replace the following paragraph:

"

texture_object_id: This is given by 16-bits representing one of the values in the range of ‘0000 0000 0000 0000’ to ‘1111 1111 1111 1111’ in binary. The texture_object_layer_id uniquely identifies a texture object layer.

"

with

"

texture_object_id: This is given by 16-bits representing one of the values in the range of ‘0000 0000 0000 0000’ to ‘1111 1111 1111 1111’ in binary. The texture_object_id uniquely identifies a texture object layer.

"

In Subclause 6.3.8, Still texture object, replace the following paragraph:

"

max_bitplanes -- This field indicates the number of maximum bitplanes inbilevel_quant mode.

"

with

"

max_bitplanes -- This field indicates the number of maximum bitplanes in all three quantization modes.

"

In Subclause 6.3.8.1, Texture Layer Decoding, delete the following paragraphs:

"

tree_blocks: The tree block is that wavelet coefficients are organized in a tree structure which is rooted in the low-low band (DC band) of the wavelet decomposition, then extends into the higher frequency bands at the same spatial location. Note the DC band is encoded separately.

spatial_layers: This field is equivalent to the maximum number of the wavelet decomposition layers in that scalability layer.

"

In Subclause 6.3.8.1, Texture Layer Decoding, replace the following paragraph:

"

quant_dc_byte: This field indicates the quantization step size for one color component of the DC subband. A zero value is forbidden. The quantization step size parameter, quant_dc, is decoded using the function get_param( ): quant = get_param(7);

"

with

"

quant_dc_byte: This field indicates the quantization step size for one color component of the DC subband. A zero value is forbidden. The quantization step size parameter, quant_dc, is decoded using the function get_param( ): quant_dc = get_param(7); where get_param() function is defined in the description of band_offset_byte.

"

In Subclause 6.3.8.1, Texture Layer Decoding, replace the following paragraph:

"

band_max_byte -- This field defines one byte of the maximum value of the DC band. The parameter band_max_value is decoded using function get_param

band_max_value = get_param(7);

"

with

"

band_max_byte -- This field defines one byte of the maximum value of the DC band. The parameter band_max_value is decoded using function get_param( ). The number of maximum bitplanes for DC band is derived from CEIL(log2(**band_max_value**+1))

band_max_value = get_param(7);

"

In Subclause 6.3.8.1, Texture Layer Decoding, remove the following paragraphs:

"

root_max_alphabet_byte-- This field defines one byte of the maximum absolute value of the quantized coefficients of the three lowest AC bands. This parameter is decoded using the function get_param( ):

root_max_alphabet = get_param (7);

valz_max_alphabet_byte-- This field defines one byte of the maximum absolute value of the quantized coefficients of the 3 highest AC bands. The parameter valz_max is decoded using the function get_param( ):

valz_max_alphabet = get_param (7);

valnz_max_alphabet_byte-- This field defines one byte of the maximum absolute value of the quantized coefficients which belong to the middle AC bands (the bands between the 3 lowest and the 3 highest AC bands). The parameter valnz_max_alphabet is decoded using the function get_param( ):

valnz_max_alphabet = get_param (7);

"

In Subclause 7.3, VOP reconstruction, replace the following formula:

"

$$0 \le d[y][x] \le 2^{bits\_per\_pixel} - 1, \text{ for all x, y}$$

"

with

"

$$d[y][x] = \begin{cases} 2^{bits\_per\_pixel} - 1; & d[y][x] > 2^{bits\_per\_pixel} - 1 \\ d[y][x]; & 0 \le d[y][x] \le 2^{bits\_per\_pixel} - 1 \\ 0; & d[y][x] < 0 \end{cases}$$

"

In Subclause 7.4.1.3, Escape code, replace the following:

"

Type 3 : ESC is followed by “11”, and the code following ESC + “11” is decoded as fixed length codes. This type of escape codes are represented by 1-bit LAST, 6-bit RUN and 12-bit LEVEL. A marker bit is inserted before and after the 12-bit-LEVEL in order to avoid the resync_marker emulation. Use of this escape sequence for encoding the combinations listed in Table B-16 and Table B-17is prohibited. The codes for RUN and LEVEL are given in Table B-18.

Type 4: The fourth type of escape code is used if and only if short_video_header is 1. In this case, the 15 bits following ESC are decoded as fixed length codes represented by 1-bit LAST, 6-bit RUN and 8-bit LEVEL. The values 0000 0000 and 1000 000 for LEVEL are not used (they are reserved).

"

with

"

Type 3 : ESC is followed by “11”, and the code following ESC + “11” is decoded as fixed length codes. This type of escape codes are represented by 1-bit LAST, 6-bit RUN and 12-bit LEVEL. A marker bit is inserted before and after the 12-bit-LEVEL in order to avoid the resync_marker emulation. Use of this escape sequence for encoding the combinations listed in in Table B-16 and Table B-17is prohibited. The codes for RUN and LEVEL are given in Tables B-18a and b.

Type 4: The fourth type of escape code is used if and only if short_video_header is 1. In this case, the 15 bits following ESC are decoded as fixed length codes represented by 1-bit LAST, 6-bit RUN and 8-bit LEVEL. The values 0000 0000 and 1000 000 for LEVEL are not used (they are reserved). The codes for RUN and LEVEL are given in Table B-18 a and c.

"

In Subclause 7.4.3, replace the following subtitle:

"

Intra dc and ac prediction for intra macroblocks

"

with

"

Dc and ac prediction for intra macroblocks

"

In Subclause 7.4.3.3, Adaptive AC Coefficient prediction, correct the indexes for coefficients in the formula by replacing the following paragraph:

"

If block ‘A’ was selected as the predictor for the block for which coefficient prediction is to be performed, calculate the first column of the quantized AC coefficients as follows.

$QF_x[0][i] = PQF_x[0][i] + (QF_A[0][i] * QP_A) // QP_x$ i = 1 to 7

If block ‘C’ was selected as the predictor for the block for which coefficient prediction is to be performed, calculate the first row of the quantized AC coefficients as follows.

$QF_x[j][0] = PQF_x[j][0] + (QF_C[j][0] * QP_C) // QP_x$ i = 1 to 7

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.

"

with

"

If block ‘A’ was selected as the predictor for the block for which coefficient prediction is to be performed, calculate the first column of the quantized AC coefficients as follows.

$QF_x[v][0] = PQF_x[v][0] + (QF_A[v][0] * QP_A) // QP_x$ v = 1 to 7

If block ‘C’ was selected as the predictor for the block for which coefficient prediction is to be performed, calculate the first row of the quantized AC coefficients as follows.

$QF_x[0][u] = PQF_x[0][u] + (QF_C[0][u] * QP_C) // QP_x$ u = 1 to 7

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.

"

In Subclause 7.4.4.4, Saturation, replace the following:

"

The coefficients resulting from the Inverse Quantisation Arithmetic are saturated to lie in the range [–2bits_per_pixel + 3 , 2bits_per_pixel + 3 – 1]. Thus:

"

with

"

The coefficients resulting from the Inverse Quantisation Arithmetic are saturated to lie in the range $[-2^{bits\_per\_pixel + 3}, 2^{bits\_per\_pixel + 3} - 1]$. Thus:

"

In Subclause 7.4.5, Inverse DCT, replace the following:

"

Once the DCT coefficients, F[u][v] are reconstructed, the inverse DCT transform defined in annex A shall be applied to obtain the inverse transformed values, $f[y][x]$. These values shall be saturated so that: $-2^{N\_bit} \le f[y][x] \le 2^{N\_bit} - 1$, for all x, y.

"

with

"

Once the DCT coefficients, F[u][v] are reconstructed, the inverse DCT transform defined in annex A shall be applied to obtain the inverse transformed values, $f[y][x]$.These values shall be saturated so that: $-2^{bits\_per\_pixel} \le f[y][x] \le 2^{bits\_per\_pixel} - 1$, for all x, y.

"

In Subclause 7.5.1.2, VOP decoding, replace the following paragraphs:

"

- vop_shape_coding_type

This flag is used in error resilient mode and enables the use of intra shape codes in P-VOPs. Finally, in the VOP class, it is necessary to decode

"

with

"

- vop_shape_coding_type

This flag enables the use of intra shape codes in P- or B-VOPs. Finally, in the VOP class, it is necessary to decode

"

In Subclause 7.5.2.2, Binary alpha block motion compensation, replace the following paragraph:

"

Motion Vector of shape (MVs) is used for motion compensation (MC) of shape. The value of MVs is reconstructed as described in subclause 7.5.2.3. Integer pixel motion compensation is carried out on a 16x16 block basis according to subclause 7.5.2.4. Overlapped MC, half sample MC and 8x8 MC are not carried out.

"

with

"

Motion Vector of shape (MVs) is used for motion compensation (MC) of shape. The value of MVs is reconstructed as described in subclause 7.5.2.3. Unrestricted motion compensation is applied, however, Overlapped MC, half sample MC and 8x8 MC are not carried out. Integer pixel unrestricted motion compensation is carried out on a 16x16 block basis according to subclause 7.5.2.4.

"

In Subclause 7.5.4.4, Greyscale Shape Decoding - Intra Macroblock, insert the following paragraphs at the end of the subclause:

"

When the interlaced is equal to “1”, alternate scan should not be applied to coding of gray-level alpha.

When both the interlaced is equal to “1” and the video_object_layer_shape is equal to “11” (grayscale), only the frame DCT should be applied to coding of gray-level alpha.

"

In Subclause 7.5.4.5, Greyscale Shape Decoding – Inter Macroblocks and Motion Compensation, insert the following sentences at the end of the subclause:

"

When both the interlaced is equal to “1” and the video_object_layer_shape is equal to “11” (grayscale), the field padding should be applied to coding of gray-level alpha.

When both the interlaced is equal to “1” and the video_object_layer_shape is equal to “11” (grayscale), both frame and field motion compensation are applicable in coding of texture and gray-level alpha, but only the frame DCT should be applied to coding of gray-level alpha.

"

In Subclause 7.6.3, General motion vector decoding process, replace the following:

"

To decode a motion vector (MVx, MVy), the differential motion vector (MVDx, MVDy) is extracted from the bitstream by using the variable length decoding.

"

with

"

To decode a motion vector (MVx, MVy), the differential motion vector (MVDx, MVDy) is extracted from the bitstream by using the variable length decoding as described in subclause 6.3.6.2.

"

In Subclause 7.6.4, Unrestricted motion compensation, replace the following paragraph:

"

where vhmcsr = vop_horizontal_mc_spatial_reference, vvmcsr = vop_vertical_mc_spatial_reference, (ycurr, xcurr) are the coordinates of a sample in the current VOP, (yref, xref) are the coordinates of a sample in the reference VOP, (dy, dx) is the motion vector, and (ydim, xdim) are the dimensions of the bounding rectangle of the reference VOP. All coordinates are related to the absolute coordinate system shown in Figure 7-19. Note that for rectangular VOP, a reference VOP is defined by video_object_layer_width and video_object_layer_height. For an arbitrary shape VOP, a reference VOP of luminance is defined by vop_width and vop_height extended to multiple of 16, while that of chrominance is defined by (vop_width>>1**)** and (vop_height>>1) extended to multiple of 8.

"

with

"

where vhmcsr = vop_horizontal_mc_spatial_ref, vvmcsr = vop_vertical_mc_spatial_ref, (ycurr, xcurr) are the coordinates of a sample in the current VOP, (yref, xref) are the coordinates of a sample in the reference VOP, (dy, dx) is the motion vector, and (ydim, xdim) are the dimensions of the bounding rectangle of the reference VOP. All coordinates are related to the absolute coordinate system shown in Figure 7-19. Note that for rectangular VOP, a reference VOP is defined by video_object_layer_width and video_object_layer_height. For an arbitrary shape VOP, a reference VOP of luminance is defined by vop_width and vop_height extended to multiple of 16, while that of chrominance is defined by (vop_width>>1**)** and (vop_height>>1) extended to multiple of 8.

"

In Subclause 7.7.2.1, replace the description for field_motion_compensate_one_reference():

"

```
field_motion_compensate_one_reference(
    luma_pred, cb_pred, cr_pred, /* Prediction component pel array */
    luma_ref, cb_ref, cr_ref,    /* Reference VOP pel arrays */
    mv_top_x, mv_top_y,          /* top field motion vector */
    mv_bot_x, mv_bot_y,          /* bottom field motion vector */
    top_field_ref,               /* top field reference */
    bottom_field_ref,            /* bottom field reference */
    x, y,                        /* current luma macroblock coords */
    rounding_type)               /* rounding type */
{
    mc(luma_pred, luma_ref, x, y, 16, 16, mv_top_x, mv_top_y,
        rounding_type, 0, top_field_ref, 2);
    mc(luma_pred, luma_ref, x, y, 16, 16, mv_bot_x, mv_bot_y,
        rounding_type, 1, bottom_field_ref, 2);
    mc(cb_pred, cb_ref, x/2, y/2, 8, 8,
        Div2Round(mv_top_x), Div2Round(mv_top_y),
        rounding_type, 0, top_field_ref, 2);
    mc(cr_pred, cr_ref, x/2, y/2, 8, 8,
        Div2Round(mv_top_x), Div2Round(mv_top_y),
        rounding_type, 0, top_field_ref, 2);
    mc(cb_pred, cb_ref, x/2, y/2, 8, 8,
        Div2Round(mv_bot_x), Div2Round(mv_bot_y),
        rounding_type, 0, top_field_ref, 2);
    mc(cr_pred, cr_ref, x/2, y/2, 8, 8,
        Div2Round(mv_bot_x), Div2Round(mv_bot_y),
        rounding_type, 0, top_field_ref, 2);
}
```

"

with

"

```
field_motion_compensate_one_reference(
    luma_pred, cb_pred, cr_pred, /* Prediction component pel array */
    luma_ref, cb_ref, cr_ref,    /* Reference VOP pel arrays */
    mv_top_x, mv_top_y,          /* top field motion vector */
    mv_bot_x, mv_bot_y,          /* bottom field motion vector */
    top_field_ref,               /* top field reference */
    bottom_field_ref,            /* bottom field reference */
    x, y,                        /* current luma macroblock coords */
    rounding_type)               /* rounding type */
{
    mc(luma_pred, luma_ref, x, y, 16, 16, mv_top_x, mv_top_y,
        rounding_type, 0, top_field_ref, 2);
    mc(luma_pred, luma_ref, x, y, 16, 16, mv_bot_x, mv_bot_y,
        rounding_type, 1, bottom_field_ref, 2);
    mc(cb_pred, cb_ref, x/2, y/2, 8, 8,
        Div2Round(mv_top_x), Div2Round(mv_top_y),
        rounding_type, 0, top_field_ref, 2);
    mc(cr_pred, cr_ref, x/2, y/2, 8, 8,
```



```
        Div2Round(mv_top_x), Div2Round(mv_top_y),
        rounding_type, 0, top_field_ref, 2);
    mc(cb_pred, cb_ref, x/2, y/2, 8, 8,
        Div2Round(mv_bot_x), Div2Round(mv_bot_y),
        rounding_type, 1, bottom_field_ref, 2);
    mc(cr_pred, cr_ref, x/2, y/2, 8, 8,
        Div2Round(mv_bot_x), Div2Round(mv_bot_y),
        rounding_type, 1, bottom_field_ref, 2);
}
```
"

In Subclause 7.8.4, Sprite reference point decoding, replace three times in subclause 7.8.4:

"

no_sprite_point

"

with

"

no_of_sprite_warping_points

"

In Subclause 7.8.5, Warping, correct the typo by changing the upper case I to lower case I, replace the following equation:

"

$$b = D\,(i_2' - i_0')\,W + h\,I_2',$$

"

with

"

$$b = D\,(i_2' - i_0')\,W + h\,i_2',$$

"

In Subclause 7.9.1.3, Decoding process of temporal scalability enhancement layer, replace the following:

"

The VOP of the enhancement layer is decoded as either I-VOP, P-VOP or B-VOP. The shape of the VOP is either rectangular (video_object_layer_id is “00”) or arbitrary (video_object_layer_id is “01”).

"

with

"

The VOP of the enhancement layer is decoded as either I-VOP, P-VOP or B-VOP. The shape of the VOP is either rectangular (video_object_layer_id is “00”) or arbitrary (video_object_layer_id is “01”). B-VOP in base layer shall not be used as a reference for enhancement layer VOP although B-VOP in enhancement layer can be a reference for enhancement layer VOP.

"

In Subclause 7.9.1.3.1, Decoding of I-VOPs, replace the following paragraph

"

The decoding process of I-VOPs in enhancement layer is the same as non-scalable decoding process.

"

with

"

The decoding process of the I-VOPs in the enhancement layer is the same as the non-scalable decoding process. ref_layer_id, ref_layer_sampling_direc, hor_sampling_factor_n, horsampling_factor_m, vertical_sampling_factor_n and vertical_sampling_factor_m are ignored in the temporal scalability I-VOPs.

"

In Subclause 7.9.1.3.4, Decoding of arbitrary shaped VOPs, replace the following paragraph:

"

Prediction for arbitrary shape in P-VOPs or in B-VOPs is same as the one in the base layer (see subclause 7.5.2.1.2).

"

with

"

The vop_shape_coding_type for enhancement layer is '0' when vop_coding_type is '00'. Otherwise, the vop_shape_coding_type for enhancement layer is '1'. Prediction for arbitrary shape in P-VOPs or in B-VOPs is same as the one in the base layer (see subclause 7.5.2.1.2).

"

In Subclause 7.9.2.9, Decoding of I-VOPs, replace the following paragraph:

"

The decoding process of the I-VOP in the enhancement layer is the same as the non_scalable decoding process.

"

with

"

The decoding process of the I-VOPs in the enhancement layer is the same as the non_scalable decoding process. ref_layer_id, ref_layer_sampling_direc, hor_sampling_factor_n, horsampling_factor_m, vertical_sampling_factor_n and vertical_sampling_factor_m are ignored in the spatial scalability I-VOPs.

"

In Subclause 7.10.1, replace the following paragraphs:

"

The wavelet coefficients of DC band are decoded independently from the other bands. First the quantization step size decoded, then the magnitude of the minimum value of the differential quantization indices "band_offset" and the maximum value of the differential quantization indices "band_max_value" are decoded from bitstream. The parameter "band_offset" is negative or zero integer and the parameter "band_max" is a positive integer, so only the magnitude of these parameters are read from the bitstream.

The arithmetic model is initialized with a uniform distribution of band_max_value-band_offset+1. Then, the differential quantization indices are decoded using the arithmetic decoder in a raster scan order, starting from the upper left index and ending with the lowest right one. The model is updated with the decoding of each bits of the predicted wavelet quantization index to adopt the probability model to the statistics of DC band.

"

with

"

The wavelet coefficients of DC subband are decoded independently from the other bands. First the quantization step size decoded, then the offset value of the quantization indices "band_offset" and the maximum value of the differential quantization indices "band_max_value" are decoded from bitstream. The parameter "band_offset" is negative or zero integer and the parameter "band_max_value" is a positive integer, so only the magnitude of these parameters are read from the bitstream.

The arithmetic model is initialized with a uniform distribution. Then, the differential quantization indices are decoded using the arithmetic decoder in a bitplane-by-bitplane fashion, from MSB to LSB. In each bitplane, the indices are decoded in a raster scan order, starting from the upper left index and ending with the lowest right one. Separate probability model for each bitplane and color component is used. The model is updated with the decoding of each bits of the predicted wavelet quantization index to adopt the probability model to the statistics of DC band.

"

In Subclause 7.10.2, replace the following paragraph:

"

The bilevel_quant mode enables fine granular SNR scalability by encoding the wavelet coefficients in a bitplane by bitplane fashion. This mode uses the same zerotree symbols as the multi_quant mode. In this mode, a zero-tree map is decoded for each bitplane, indicating which wavelet coefficients are nonzero relative to that bitplane. The inverse quantization is also performed bitplane by bitplane. After the zero-tree map, additional bits are decoded to refine the accuracy of the previously decoded coefficients.

"

with

"

The bilevel_quant mode enables fine granular SNR scalability by encoding the wavelet coefficients in a bitplane-by-bitplane fashion. This mode uses the same zerotree symbols as the multi_quant mode. In this mode, a zero-tree map is decoded for each bitplane, indicating which wavelet coefficients are nonzero relative to that bitplane. The inverse quantization is also performed bitplane by bitplane. After the zero-tree map, additional bits are decoded to refine the accuracy of the previously decoded coefficients.

"

In Subclause 7.10.2.2, replace the following paragraph:

"

The zero-tree (or type) symbols, quantized coefficient values (magnitude and sign), and residual values (for the multi quant mode) are all decoded using an adaptive arithmetic decoder with a given symbol alphabet. The arithmetic decoder adaptively tracks the statistics of the zerotree symbols and decoded values. For both the single quant and multi quant modes the arithmetic decoder is initialized at the beginning of each color loop for band-by-band scanning and at the beginning of the tree-block loop for tree-depth scanning. In order to avoid start code emulation, the arithmetic encoder always starts with stuffing one bit '1' at the beginning of the entropy encoding. It also stuffs one bit '1' immediately after it encodes every 22 successive '0's. It stuffs one bit '1' to the end of bitstream in the case in which the last output bit of arithmetic encoder is '0'. Thus, the arithmetic decoder reads and discards one bit before starts entropy decoding. During the decoding, it also reads and discards one bit after receiving every 22 successive '0's. The arithmetic decoder reads one bit and discards it if the last input bit to the arithmetic decoder is '0'.

"

with

"

The zero-tree (or type) symbols, quantized coefficient values (magnitude and sign), and residual values (for the multi quant mode) are all decoded using an adaptive arithmetic decoder with a given symbol alphabet. The arithmetic decoder adaptively tracks the statistics of the zerotree symbols and decoded values. For both the single quant and multi quant modes the arithmetic decoder is initialized at the beginning of each color loop for band-by-band scanning and at the beginning of the tree-block loop for tree-depth scanning. Therefore, the decoder is initialized three times in each band for band-by-band scanning and the decoder is initialized once before going to the tree block-loop for tree-depth scanning. In order to avoid start code emulation, the arithmetic encoder always starts with stuffing one bit '1' at the beginning of the entropy encoding. It also stuffs one bit '1' immediately after it encodes every 22 successive '0's. It stuffs one bit '1' to the end of bitstream in the case in which the last output bit of arithmetic encoder is '0'. Thus, the arithmetic decoder reads and discards one bit before starts entropy decoding. During the decoding, it also reads and discards one bit after receiving every 22 successive '0's. The arithmetic decoder reads one bit and discards it if the last input bit to the arithmetic decoder is '0'. To reduce the number of bitplanes in encoding the magnitude of quantized coefficient values, the encoder subtracts the magnitude by one first. Thus the decoder adds one back after decoding the magnitude of quantized coefficient values.

"

In Subclause 7.10.3.1.1, replace the following paragraph:

"

The DC coefficient decoding is the same as that for rectangular image except the following,

1. Only those DC coefficients inside the shape boundary in the DC layer shall be traversed and decoded and DC coefficients outside the shape boundary may be set to zeros.
2. For the inverse DC prediction in the DC layer, if a reference coefficient (A, B, C in Fig.(DC predicition figure)) in the prediction context is outside the shape boundary, zero shall be used to form the prediction syntax.

"

with

"

The DC coefficient decoding is the same as that for rectangular image except the following,

1. Only those DC coefficients inside the shape boundary in the DC layer shall be traversed and decoded and DC coefficients outside the shape boundary may be set to zeros.
2. For the inverse DC prediction in the DC layer, if a reference coefficient (Fig.7-36) in the prediction context is outside the shape boundary, zero shall be used to form the prediction syntax.

"

In Subclause 8.1, replace the following:

"

Two types of shape information, *s*(*x*, *y*, *ta*) and *s*(*x*, *y*, *td*), are necessary for the background composition. *s*(*x*, *y*, *ta*) is called a "forward shape" and *s*(*x*, *y*, *td*) is called a "backward shape". If f(*x, y, td*) is the last VOP in the bitstream of the reference layer, it should be made by copying f(*x, y, ta*). In this case, two shapes s(*x, y, ta*) and s(*x, y, td*) should be identical to the previous backward shape.

"

with

"

Two types of shape information, *s*(*x*, *y*, *ta*) and *s*(*x*, *y*, *td*), are necessary for the background composition. *s*(*x*, *y*, *ta*) is called a "forward shape" and *s*(*x*, *y*, *td*) is called a "backward shape". If f(*x, y, ta*) does not exist, the pixel value of fc(*x, y, t*) for s(*x, y, td*) = 0 is given by f(*x, y, td*) and the pixel value of fc(*x, y, t*) for s(*x, y, td*) = 1 is given by repetitive padding from the boundary pixel

value. If f(*x, y, td*) does not exist, pixel value of fc(*x, y, t*) for s(*x, y, ta*) = 0 is given by f(*x, y, ta*) and the pixel value of fc(*x, y, t*) for s(*x, y, ta*) = 1 is given by repetitive padding from the boundary pixel value.
"

In Subclause 9.1, Visual Object Types, replace NOTE 2 under Table 9-1:
"

NOTE 2 The parameters are restricted as follows for the tool “P-VOP based temporal scalability Arbitrary Shape”:

- ref_select_code shall be either ‘00’ or ‘01’.
- reference layer shall be either I-VOP or P-VOP.
- load_backward_shape shall be ‘0’ and background composition is not performed.

"

with

"

NOTE 2 — The parameters are restricted as follows for the tool “P-VOP based temporal scalability”:

- ref_select_code shall be either ‘00’ or ‘01’.
- vop_coding_type of the enhancement layer VOP shall be either ‘00’ or ‘01’.
- vop_coding_type of the reference layer VOP shall be either ‘00’ or ‘01’.
- load_backward_shape shall be ‘0’ and background composition is not performed.

"

In Subclause 9.1, Visual Object Types, add below Table 9-1, after Note 1 and Note 2, the following Note:
"

NOTE 3 — An ‘X’ for the “Method 1/Method 2 Quantization” indicates that both quantization methods are supported by the Visual Object Type. Where there is no ‘X’ the only quantization method supported is the Second Inverse Quantization Method (subclause 7.4.4.2)
"

In Subclause 9.2, Visual Profiles, delete the following paragraph:
"

Note that the Profiles can be grouped into three categories: Natural Visual (Profile numbers 1-5), Synthetic Visual (Profile numbers 8 and 9), and Synthetic/Natural Hybrid Visual (Profile numbers 6 and 7).
"

In Subclause 9.3.2.1 Scalable Texture Profile, replace Table 9-3 with the following table and add the second footnote:
"

Table 9-3 -- Scalable texture profile levels

Profile	Levels	Default Wavelet Filter	Maximum Download Filter length	Maximum number of Decomposition Levels	Typical Visual Session Size[1]	Maximum Qp value	Maximum number of pixels/ Session	VCV decoder rate (Equivalent MB/s)[2]	Maximum number of bitplanes for DC values	Maximum VCV Buffer size (Equivalent MB)[2]
Scalable Texture	L3	Float, Integer	ON, 15	10	8192x8192	12 bits	67108864	262144	18	262144
Scalable Texture	L2	Integer	ON, 15	8	2048x2048	10 bits	4194304	16384	16	16384
Scalable Texture	L1	Integer	OFF	5	704x576	8 bits	405504	1584	13	1584

(1) This column is for informative use only. It provides an example configuration of the Maximum number of pixels/Session.

(2) This Still texture VCV model is separate from the global video VCV model. An equivalent MB corresponds to 256 pixels.

"

In Subclause 9.3.3.1, Basic Animated Texture Profile, replace the following paragraph:

"

Level 1 = Simple Facial Animation Profile @ Level 1 + Scalable Texture @ Level 1 + the following restrictions on Basic Animated Texture object types:

- Maximum number of Mesh objects (with uniform topology): 4,
  - Maximum total number of nodes (vertices) in Mesh objects: 480,
    (= 4 x nr. Of nodes of a uniform mesh covering a QCIF image with 16x16 pixel elements),
  - Maximum frame-rate of a Mesh object: 30 Hz, and
  - Maximum bitrate of Mesh objects: 128 kbit/sec.

"

with

"

Level 1 = Simple Facial Animation Profile @ Level 1 + Scalable Texture @ Level 1 + the following restrictions on Basic Animated Texture object types:

- Maximum number of Mesh objects (with uniform topology): 4,
  - Maximum total number of nodes (vertices) in Mesh objects: 480,
    (= 4 x nr. Of nodes of a uniform mesh covering a QCIF image with 16x16 pixel elements),
  - Maximum frame-rate of a Mesh object: 30 Hz, and
  - Maximum bitrate of Mesh objects: 64 kbit/sec.

"

Clause A.1, Discrete cosine transform for video texture, replace the following paragraph:

"

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990.

"

with

"

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990, with the following modifications:
1) In item (1) of subclause 3.2 of the IEEE specification, the last sentence is replaced by: <<Data sets of 1 000 000 (one million) blocks each should be generated for (L=256, H=255), (L=H=5) and (L=384, H=383). >>
2) The text of subclause 3.3 of the IEEE specification is replaced by : <<For any pixel location, the peak error shall not exceed 2 in magnitude. There is no other accuracy requirement for this test.>>
3) Let F be the set of 4096 blocks Bi[y][x] (i=0..4095) defined as follows :
   a) Bi[0][0] = i - 2048
   b) Bi[7][7] = 1 if Bi[0][0] is even, Bi[7][7] = 0 if Bi[0][0] is odd
   c) All other coefficients Bi[y][x] other than Bi[0][0] and Bi[7][7] are equal to 0

For each block Bi[y][x] that belongs to set F defined above, an IDCT that claims to be compliant shall output a block f[y][x] that as a peak error of 1 or less compared to the reference saturated mathematical integer-number IDCT fíí(x,y). In other words, | f[y][x] - fíí(x,y)| shall be <= 1 for all x and y.

"

In Subclause A.2.1, Adding the mean, replace the following paragraph:

"

Before applying the inverse wavelet transform, the mean of each color component ("mean_y", "mean_u", and "mean_v") is added to the all wavelet coefficients of dc band.

"

with

"

Before applying the inverse wavelet transform, the mean of each color component ("mean_y", "mean_u", and "mean_v") is added to the all wavelet coefficients of DC subband.

"

In Subclause A.2.2, wavelet filter, replace the title:

"

A.2.2 wavelet filter

"

with

"

A.2.2 Wavelet filter

"

In Subclause A.2.2, wavelet filter, replace the following paragraph:

"

The floating filter coefficients are:

Lowpass	**g[ ] =**	
[0.35355339059327	0.70710678118655	0.35355339059327]

Highpas	**h[ ] =**	
[0.03314563036812	0.06629126073624	-0.17677669529665
-0.41984465132952	0.99436891104360	-0.41984465132952
-0.17677669529665	0.06629126073624	0.03314563036812]

"

with

"

The floating filter coefficients are:

Lowpass	**g[ ] =**	
[0.35355339059327	0.70710678118655	0.35355339059327]

Highpass	**h[ ] =**	
[0.03314563036812	0.06629126073624	-0.17677669529665
-0.41984465132952	0.99436891104360	-0.41984465132952
-0.17677669529665	0.06629126073624	0.03314563036812]

"

In Subclause A.2.2, wavelet filter, replace the following paragraph:

"

In the case of integer wavelet, the outputs at each composition level are scaled down with dividing by 8096 with rounding to the nearest integer.

"

with

"

In the case of integer wavelet, the outputs at each composition level are scaled down with dividing by 8192 with rounding to the nearest integer.

"

In Subclause A.2.3, Symmetric extension, replace the following paragraph:

"

The generated up-sampled and extended wavelet coefficients L[] and H[] are eventually specified as follows:

low-pass band :…0   L[2]   0 | **L[0] 0 L[2] 0 … L[N-4] 0 L[N-2] 0**   **|** L[N-2] 0 L[N-4] 0 …
high-pass band:…H[3] 0 H[1] | **0 H[1] 0 H[3] … 0   H[N-3] 0 H[N-1]** | 0 H[N-1] 0 H[N-3]…

"

with

"

The generated up-sampled and extended wavelet coefficients L[] and H[] are eventually specified as follows:

low-pass band :…0 L[2] 0 | **L[0] 0 L[2] 0 … L[N-4] 0 L[N-2] 0** | L[N-2] 0 L[N-4] 0 …
high-pass band:… H[3] 0 H[1] | **0 H[1] 0 H[3] … 0 H[N-3] 0 H[N-1]** | 0 H[N-1] 0 H[N-3]…
"

Subclause B.1.1, Macroblock type, replace the following caption of Table B-1:
"

Table B-1 -- Macroblock types and included data elements for I- and P-VOPs in combined motion-shape-texture coding

"
with
"

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

"
In Subclause B.1.1, Macroblock type, replace the following caption of Table B-3:
"

Table B-3 -- VLC table for modb in combined motion-shape-texture coding

"
with
"

Table B-3 -- VLC table for modb

"
In Subclause B.1.1, Macroblock type, replace the following caption of Table B-4:
"

Table B-4 -- mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code != '00'||scalability=='0') for combined motion-shape-texture coding

"
with
"

Table B-4 -- mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code != '00'||scalability=='0')

"

In Subclause B.1.1, Macroblock type, replace the following caption of Table B-5:
"

Table B-5 -- mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code == '00'&&scalability!='0') for combined motion-shape-texture coding

"
with
"

Table B-5 -- mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code == '00'&&scalability!='0')

"
In Subclause B.1.2, Macroblock pattern, replace the following caption of Table B-6:
"

Table B-6 -- VLC table for mcbpc for I-VOPs in combined-motion-shape-texture coding and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode=="piece"

"

with
"

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

"

In Subclause B.1.2, Macroblock pattern, replace the following caption of Table B-7:
"

Table B-7 -- VLC table for mcbpc for P-VOPs in combined-motion-shape-texture and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”update”

"
with
"

Table B-7 -- VLC table for mcbpc for P-VOPs and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”update”

"

In Subclause B.1.2, Macroblock pattern, replace the following caption of Table B-8:
"

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

"
with
"

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

"
In Subclause B.1.4, DCT coefficients, insert the following text at the end of 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.

"
In Subclause B.1.4, DCT coefficients, insert the following text at the end of 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.

"
In Subclause B.1.4, DCT coefficients, insert the following text at the end of Table B-15:
"

NOTE — 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.

"
In Subclause B.1.4, DCT coefficients, replace table B-15:
"

Additional code	Differential DC	Size
000000000000 to 011111111111 *	-2048 to -4095	12
00000000000 to 01111111111 *	-1024 to -2047	11
0000000000 to 0111111111 *	-512 to -1023	10
000000000 to 011111111 *	-256 to -511	9
00000000 to 01111111	-255 to -128	8
0000000 to 0111111	-127 to -64	7
000000 to 011111	-63 to -32	6
00000 to 01111	-31 to -16	5
0000 to 0111	-15 to -8	4

000 to 011	-7 to -4	3
00 to 01	-3 to -2	2
0	-1	1
	0	0
1	1	1
10 to 11	2 to 3	2
100 to 111	4 to 7	3
1000 to 1111	8 to 15	4
10000 to 11111	16 to 31	5
100000 to 111111	32 to 63	6
1000000 to 1111111	64 to 127	7
10000000 to 11111111	128 to 255	8
100000000 to 111111111 *	256 to 511	9
1000000000 to 1111111111 *	512 to 1023	10
10000000000 to 11111111111 *	1024 to 2047	11
100000000000 to 111111111111 *	2048 to 4095	12

"

with

"

Additional code	Differential DC	Size
000000000000 to 011111111111 *	-4095 to -2048	12
00000000000 to 01111111111 *	-2047 to -1024	11
0000000000 to 0111111111 *	-1023 to -512	10
000000000 to 011111111 *	-511 to -256	9
00000000 to 01111111	-255 to -128	8
0000000 to 0111111	-127 to -64	7
000000 to 011111	-63 to -32	6
00000 to 01111	-31 to -16	5
0000 to 0111	-15 to -8	4
000 to 011	-7 to -4	3
00 to 01	-3 to -2	2
0	-1	1
	0	0
1	1	1
10 to 11	2 to 3	2
100 to 111	4 to 7	3
1000 to 1111	8 to 15	4
10000 to 11111	16 to 31	5
100000 to 111111	32 to 63	6
1000000 to 1111111	64 to 127	7
10000000 to 11111111	128 to 255	8
100000000 to 111111111 *	256 to 511	9
1000000000 to 1111111111 *	512 to 1023	10
10000000000 to 11111111111 *	1024 to 2047	11
100000000000 to 111111111111 *	2048 to 4095	12

"

In Table B-15, replace the following subscript:

"

In cases where dct_dc_size is greater than 8, marked '*' in , a marker bit is inserted after the dct_dc_additional_code to prevent start code emulations.

"

with

"

In cases where dct_dc_size is greater than 8, marked '*' in Table B-15 , a marker bit is inserted after the dct_dc_additional_code to prevent start code emulations.

"

In Subclause B.1.4, DCT coefficients, replace the following Table B-18:

"

Table B-18 -- FLC table for RUNS and LEVELS

Code	Run
000 000	0
000 001	1
000 010	2
.	.
.	.
111 111	63

Code	Level
Forbidden	-2048
1000 0000 0001	-2047
.	.
1111 1111 1110	-2
1111 1111 1111	-1
Forbidden	0
0000 0000 0001	1
0000 0000 0010	2
.	.
0111 1111 1111	2047

"

with

"

Table B-18 -- FLC table for RUNS and LEVELS

Code	Run
000 000	0
000 001	1
000	**2**
.	**.**
.	**.**
111	63

a) FLC code for RUN

Code	Level
forbidden	-2048
1000 0000 0001	-2047
.	.
1111 1111 1110	-2
1111 1111 1111	-1
forbidden	0
0000 0000 0001	1
0000 0000 0010	2
.	.
0111 1111 1111	2047

b) FLC code for LEVEL

Code	Level
forbidden	-128
1000 0001	-127
.	.
1111 1110	-2
1111 1111	-1
forbidden	0
0000 0001	1
0000 0010	2
.	.
0111 1111	127

c) FLC code for LEVEL when short_video_header is 1

"

In Table B-23, RVLC table for TCOEFF, move description of ESCAPE code behind the table, therefore replace the following:

"

Table B-23 RVLC table for TCOEF

ESCAPE code is added at the beginning and the end of these fixed-length codes for realizing two-way decode as shown below. A marker bit is inserted before and after the 11-bit-LEVEL in order to avoid the resync_marker emulation.

ESCAPE	LAST	RUN	marker bit	LEVEL	marker bit	ESCAPE
00001	x	xxxxxx	1	xxxxxxxxxxx	1	0000s

NOTE — There are two types for ESCAPE added at the end of these fixed-length codes, and codewords are "0000s". Also, S=0 : LEVEL is positive and S=1 : LEVEL is negative.

	Intra			inter				
INDEX	LAST	RUN	LEVEL	LAST	RUN	LEVEL	BITS	VLC_CODE
0	0	0	1	0	0	1	4	110s
1	0	0	2	0	1	1	4	111s
2	0	1	1	0	0	2	5	0001s
3	0	0	3	0	2	1	5	1010s
4	1	0	1	1	0	1	5	1011s
5	0	2	1	0	0	3	6	00100s
6	0	3	1	0	3	1	6	00101s
7	0	1	2	0	4	1	6	01000s
8	0	0	4	0	5	1	6	01001s
9	1	1	1	1	1	1	6	10010s
10	1	2	1	1	2	1	6	10011s
11	0	4	1	0	1	2	7	001100s
12	0	5	1	0	6	1	7	001101s
13	0	0	5	0	7	1	7	010100s
14	0	0	6	0	8	1	7	010101s
15	1	3	1	1	3	1	7	011000s
16	1	4	1	1	4	1	7	011001s
17	1	5	1	1	5	1	7	100010s
18	1	6	1	1	6	1	7	100011s
19	0	6	1	0	0	4	8	0011100s
20	0	7	1	0	2	2	8	0011101s
21	0	2	2	0	9	1	8	0101100s
22	0	1	3	0	10	1	8	0101101s
23	0	0	7	0	11	1	8	0110100s
24	1	7	1	1	7	1	8	0110101s
25	1	8	1	1	8	1	8	0111000s
26	1	9	1	1	9	1	8	0111001s
27	1	10	1	1	10	1	8	1000010s
28	1	11	1	1	11	1	8	1000011s
29	0	8	1	0	0	5	9	00111100s
30	0	9	1	0	0	6	9	00111101s
31	0	3	2	0	1	3	9	01011100s
32	0	4	2	0	3	2	9	01011101s
33	0	1	4	0	4	2	9	01101100s
34	0	1	5	0	12	1	9	01101101s
35	0	0	8	0	13	1	9	01110100s
36	0	0	9	0	14	1	9	01110101s
37	1	0	2	1	0	2	9	01111000s
38	1	12	1	1	12	1	9	01111001s

39	1	13	1	1	13	1	9	10000010s
40	1	14	1	1	14	1	9	10000011s
41	0	10	1	0	0	7	10	001111100s
42	0	5	2	0	1	4	10	001111101s
43	0	2	3	0	2	3	10	010111100s
44	0	3	3	0	5	2	10	010111101s
45	0	1	6	0	15	1	10	011011100s
46	0	0	10	0	16	1	10	011011101s
47	0	0	11	0	17	1	10	011101100s
48	1	1	2	1	1	2	10	011101101s
49	1	15	1	1	15	1	10	011110100s
50	1	16	1	1	16	1	10	011110101s
51	1	17	1	1	17	1	10	011111000s
52	1	18	1	1	18	1	10	011111001s
53	1	19	1	1	19	1	10	100000010s
54	1	20	1	1	20	1	10	100000011s
55	0	11	1	0	0	8	11	0011111100s
56	0	12	1	0	0	9	11	0011111101s
57	0	6	2	0	1	5	11	0101111100s
58	0	7	2	0	3	3	11	0101111101s
59	0	8	2	0	6	2	11	0110111100s
60	0	4	3	0	7	2	11	0110111101s
61	0	2	4	0	8	2	11	0111011100s
62	0	1	7	0	9	2	11	0111011101s
63	0	0	12	0	18	1	11	0111101100s
64	0	0	13	0	19	1	11	0111101101s
65	0	0	14	0	20	1	11	0111110100s
66	1	21	1	1	21	1	11	0111110101s
67	1	22	1	1	22	1	11	0111111000s
68	1	23	1	1	23	1	11	0111111001s
69	1	24	1	1	24	1	11	1000000010s
70	1	25	1	1	25	1	11	1000000011s
71	0	13	1	0	0	10	12	00111111100s
72	0	9	2	0	0	11	12	00111111101s
73	0	5	3	0	1	6	12	01011111100s
74	0	6	3	0	2	4	12	01011111101s
75	0	7	3	0	4	3	12	01101111100s
76	0	3	4	0	5	3	12	01101111101s
77	0	2	5	0	10	2	12	01110111100s
78	0	2	6	0	21	1	12	01110111101s
79	0	1	8	0	22	1	12	01111011100s
80	0	1	9	0	23	1	12	01111011101s
81	0	0	15	0	24	1	12	01111101100s
82	0	0	16	0	25	1	12	01111101101s
83	0	0	17	0	26	1	12	01111110100s
84	1	0	3	1	0	3	12	01111110101s
85	1	2	2	1	2	2	12	01111111000s
86	1	26	1	1	26	1	12	01111111001s
87	1	27	1	1	27	1	12	10000000010s
88	1	28	1	1	28	1	12	10000000011s
89	0	10	2	0	0	12	13	001111111100s
90	0	4	4	0	1	7	13	001111111101s

91	0	5	4	0	2	5	13	010111111100s
92	0	6	4	0	3	4	13	010111111101s
93	0	3	5	0	6	3	13	011011111100s
94	0	4	5	0	7	3	13	011011111101s
95	0	1	10	0	11	2	13	011101111100s
96	0	0	18	0	27	1	13	011101111101s
97	0	0	19	0	28	1	13	011110111100s
98	0	0	22	0	29	1	13	011110111101s
99	1	1	3	1	1	3	13	011111011100s
100	1	3	2	1	3	2	13	011111011101s
101	1	4	2	1	4	2	13	011111101100s
102	1	29	1	1	29	1	13	011111101101s
103	1	30	1	1	30	1	13	011111110100s
104	1	31	1	1	31	1	13	011111110101s
105	1	32	1	1	32	1	13	011111111000s
106	1	33	1	1	33	1	13	011111111001s
107	1	34	1	1	34	1	13	100000000010s
108	1	35	1	1	35	1	13	100000000011s
109	0	14	1	0	0	13	14	0011111111100s
110	0	15	1	0	0	14	14	0011111111101s
111	0	11	2	0	0	15	14	0101111111100s
112	0	8	3	0	0	16	14	0101111111101s
113	0	9	3	0	1	8	14	0110111111100s
114	0	7	4	0	3	5	14	0110111111101s
115	0	3	6	0	4	4	14	0111011111100s
116	0	2	7	0	5	4	14	0111011111101s
117	0	2	8	0	8	3	14	0111101111100s
118	0	2	9	0	12	2	14	0111101111101s
119	0	1	11	0	30	1	14	0111110111100s
120	0	0	20	0	31	1	14	0111110111101s
121	0	0	21	0	32	1	14	0111111011100s
122	0	0	23	0	33	1	14	0111111011101s
123	1	0	4	1	0	4	14	0111111101100s
124	1	5	2	1	5	2	14	0111111101101s
125	1	6	2	1	6	2	14	0111111110100s
126	1	7	2	1	7	2	14	0111111110101s
127	1	8	2	1	8	2	14	0111111111000s
128	1	9	2	1	9	2	14	0111111111001s
129	1	36	1	1	36	1	14	1000000000010s
130	1	37	1	1	37	1	14	1000000000011s
131	0	16	1	0	0	17	15	00111111111100s
132	0	17	1	0	0	18	15	00111111111101s
133	0	18	1	0	1	9	15	01011111111100s
134	0	8	4	0	1	10	15	01011111111101s
135	0	5	5	0	2	6	15	01101111111100s
136	0	4	6	0	2	7	15	01101111111101s
137	0	5	6	0	3	6	15	01110111111100s

138	0	3	7	0	6	4	15	01110111111101s
139	0	3	8	0	9	3	15	01111011111100s
140	0	2	10	0	13	2	15	01111011111101s
141	0	2	11	0	14	2	15	01111101111100s
142	0	1	12	0	15	2	15	01111101111101s
143	0	1	13	0	16	2	15	01111110111100s
144	0	0	24	0	34	1	15	01111110111101s
145	0	0	25	0	35	1	15	01111111011100s
146	0	0	26	0	36	1	15	01111111011101s
147	1	0	5	1	0	5	15	01111111101100s
148	1	1	4	1	1	4	15	01111111101101s
149	1	10	2	1	10	2	15	01111111110100s
150	1	11	2	1	11	2	15	01111111110101s
151	1	12	2	1	12	2	15	01111111111000s
152	1	38	1	1	38	1	15	01111111111001s
153	1	39	1	1	39	1	15	10000000000010s
154	1	40	1	1	40	1	15	10000000000011s
155	0	0	27	0	0	19	16	001111111111100s
156	0	3	9	0	3	7	16	001111111111101s
157	0	6	5	0	4	5	16	010111111111100s
158	0	7	5	0	7	4	16	010111111111101s
159	0	9	4	0	17	2	16	011011111111100s
160	0	12	2	0	37	1	16	011011111111101s
161	0	19	1	0	38	1	16	011101111111100s
162	1	1	5	1	1	5	16	011101111111101s
163	1	2	3	1	2	3	16	011110111111100s
164	1	13	2	1	13	2	16	011110111111101s
165	1	41	1	1	41	1	16	011111011111100s
166	1	42	1	1	42	1	16	011111011111101s
167	1	43	1	1	43	1	16	011111101111100s

168	1	44	1	1	44	1	16	01111110111110 1s
169	ESCAPE						5	0000s

"
with
"

Table B-23 RVLC table for TCOEF

	Intra			inter				
INDEX	LAST	RUN	LEVEL	LAST	RUN	LEVEL	BITS	VLC_CODE
0	0	0	1	0	0	1	4	110s
1	0	0	2	0	1	1	4	111s
2	0	1	1	0	0	2	5	0001s
3	0	0	3	0	2	1	5	1010s
4	1	0	1	1	0	1	5	1011s
5	0	2	1	0	0	3	6	00100s
6	0	3	1	0	3	1	6	00101s
7	0	1	2	0	4	1	6	01000s
8	0	0	4	0	5	1	6	01001s
9	1	1	1	1	1	1	6	10010s
10	1	2	1	1	2	1	6	10011s
11	0	4	1	0	1	2	7	001100s
12	0	5	1	0	6	1	7	001101s
13	0	0	5	0	7	1	7	010100s
14	0	0	6	0	8	1	7	010101s
15	1	3	1	1	3	1	7	011000s
16	1	4	1	1	4	1	7	011001s
17	1	5	1	1	5	1	7	100010s
18	1	6	1	1	6	1	7	100011s
19	0	6	1	0	0	4	8	0011100s
20	0	7	1	0	2	2	8	0011101s
21	0	2	2	0	9	1	8	0101100s
22	0	1	3	0	10	1	8	0101101s
23	0	0	7	0	11	1	8	0110100s
24	1	7	1	1	7	1	8	0110101s
25	1	8	1	1	8	1	8	0111000s
26	1	9	1	1	9	1	8	0111001s
27	1	10	1	1	10	1	8	1000010s
28	1	11	1	1	11	1	8	1000011s
29	0	8	1	0	0	5	9	00111100s
30	0	9	1	0	0	6	9	00111101s
31	0	3	2	0	1	3	9	01011100s
32	0	4	2	0	3	2	9	01011101s
33	0	1	4	0	4	2	9	01101100s
34	0	1	5	0	12	1	9	01101101s
35	0	0	8	0	13	1	9	01110100s
36	0	0	9	0	14	1	9	01110101s
37	1	0	2	1	0	2	9	01111000s
38	1	12	1	1	12	1	9	01111001s
39	1	13	1	1	13	1	9	10000010s
40	1	14	1	1	14	1	9	10000011s
41	0	10	1	0	0	7	10	001111100s

42	0	5	2	0	1	4	10	001111101s
43	0	2	3	0	2	3	10	010111100s
44	0	3	3	0	5	2	10	010111101s
45	0	1	6	0	15	1	10	011011100s
46	0	0	10	0	16	1	10	011011101s
47	0	0	11	0	17	1	10	011101100s
48	1	1	2	1	1	2	10	011101101s
49	1	15	1	1	15	1	10	011110100s
50	1	16	1	1	16	1	10	011110101s
51	1	17	1	1	17	1	10	011111000s
52	1	18	1	1	18	1	10	011111001s
53	1	19	1	1	19	1	10	100000010s
54	1	20	1	1	20	1	10	100000011s
55	0	11	1	0	0	8	11	0011111100s
56	0	12	1	0	0	9	11	0011111101s
57	0	6	2	0	1	5	11	0101111100s
58	0	7	2	0	3	3	11	0101111101s
59	0	8	2	0	6	2	11	0110111100s
60	0	4	3	0	7	2	11	0110111101s
61	0	2	4	0	8	2	11	0111011100s
62	0	1	7	0	9	2	11	0111011101s
63	0	0	12	0	18	1	11	0111101100s
64	0	0	13	0	19	1	11	0111101101s
65	0	0	14	0	20	1	11	0111110100s
66	1	21	1	1	21	1	11	0111110101s
67	1	22	1	1	22	1	11	0111111000s
68	1	23	1	1	23	1	11	0111111001s
69	1	24	1	1	24	1	11	1000000010s
70	1	25	1	1	25	1	11	1000000011s
71	0	13	1	0	0	10	12	00111111100s
72	0	9	2	0	0	11	12	00111111101s
73	0	5	3	0	1	6	12	01011111100s
74	0	6	3	0	2	4	12	01011111101s
75	0	7	3	0	4	3	12	01101111100s
76	0	3	4	0	5	3	12	01101111101s
77	0	2	5	0	10	2	12	01110111100s
78	0	2	6	0	21	1	12	01110111101s
79	0	1	8	0	22	1	12	01111011100s
80	0	1	9	0	23	1	12	01111011101s
81	0	0	15	0	24	1	12	01111101100s
82	0	0	16	0	25	1	12	01111101101s
83	0	0	17	0	26	1	12	01111110100s
84	1	0	3	1	0	3	12	01111110101s
85	1	2	2	1	2	2	12	01111111000s
86	1	26	1	1	26	1	12	01111111001s
87	1	27	1	1	27	1	12	10000000010s
88	1	28	1	1	28	1	12	10000000011s
89	0	10	2	0	0	12	13	001111111100s
90	0	4	4	0	1	7	13	001111111101s
91	0	5	4	0	2	5	13	010111111100s
92	0	6	4	0	3	4	13	010111111101s
93	0	3	5	0	6	3	13	011011111100s

94	0	4	5	0	7	3	13	011011111101s
95	0	1	10	0	11	2	13	011101111100s
96	0	0	18	0	27	1	13	011101111101s
97	0	0	19	0	28	1	13	011110111100s
98	0	0	22	0	29	1	13	011110111101s
99	1	1	3	1	1	3	13	011111011100s
100	1	3	2	1	3	2	13	011111011101s
101	1	4	2	1	4	2	13	011111101100s
102	1	29	1	1	29	1	13	011111101101s
103	1	30	1	1	30	1	13	011111110100s
104	1	31	1	1	31	1	13	011111110101s
105	1	32	1	1	32	1	13	011111111000s
106	1	33	1	1	33	1	13	011111111001s
107	1	34	1	1	34	1	13	100000000010s
108	1	35	1	1	35	1	13	100000000011s
109	0	14	1	0	0	13	14	0011111111100s
110	0	15	1	0	0	14	14	0011111111101s
111	0	11	2	0	0	15	14	0101111111100s
112	0	8	3	0	0	16	14	0101111111101s
113	0	9	3	0	1	8	14	0110111111100s
114	0	7	4	0	3	5	14	0110111111101s
115	0	3	6	0	4	4	14	0111011111100s
116	0	2	7	0	5	4	14	0111011111101s
117	0	2	8	0	8	3	14	0111101111100s
118	0	2	9	0	12	2	14	0111101111101s
119	0	1	11	0	30	1	14	0111110111100s
120	0	0	20	0	31	1	14	0111110111101s
121	0	0	21	0	32	1	14	0111111011100s
122	0	0	23	0	33	1	14	0111111011101s
123	1	0	4	1	0	4	14	0111111101100s
124	1	5	2	1	5	2	14	0111111101101s
125	1	6	2	1	6	2	14	0111111110100s
126	1	7	2	1	7	2	14	0111111110101s
127	1	8	2	1	8	2	14	0111111111000s
128	1	9	2	1	9	2	14	0111111111001s
129	1	36	1	1	36	1	14	1000000000010s
130	1	37	1	1	37	1	14	1000000000011s
131	0	16	1	0	0	17	15	00111111111100s
132	0	17	1	0	0	18	15	00111111111101s
133	0	18	1	0	1	9	15	01011111111100s
134	0	8	4	0	1	10	15	01011111111101s
135	0	5	5	0	2	6	15	01101111111100s
136	0	4	6	0	2	7	15	01101111111101s
137	0	5	6	0	3	6	15	01110111111100s
138	0	3	7	0	6	4	15	01110111111101s
139	0	3	8	0	9	3	15	01111011111100

								s
140	0	2	10	0	13	2	15	01110111111101s
141	0	2	11	0	14	2	15	01111011111100s
142	0	1	12	0	15	2	15	01111011111101s
143	0	1	13	0	16	2	15	01111101111100s
144	0	0	24	0	34	1	15	01111101111101s
145	0	0	25	0	35	1	15	01111110111100s
146	0	0	26	0	36	1	15	01111110111101s
147	1	0	5	1	0	5	15	01111111011100s
148	1	1	4	1	1	4	15	01111111011101s
149	1	10	2	1	10	2	15	01111111110100s
150	1	11	2	1	11	2	15	01111111110101s
151	1	12	2	1	12	2	15	01111111111000s
152	1	38	1	1	38	1	15	01111111111001s
153	1	39	1	1	39	1	15	10000000000010s
154	1	40	1	1	40	1	15	10000000000011s
155	0	0	27	0	0	19	16	001111111111100s
156	0	3	9	0	3	7	16	001111111111101s
157	0	6	5	0	4	5	16	010111111111100s
158	0	7	5	0	7	4	16	010111111111101s
159	0	9	4	0	17	2	16	011011111111100s
160	0	12	2	0	37	1	16	011011111111101s
161	0	19	1	0	38	1	16	011101111111100s
162	1	1	5	1	1	5	16	011101111111101s
163	1	2	3	1	2	3	16	011110111111100s
164	1	13	2	1	13	2	16	011110111111101s
165	1	41	1	1	41	1	16	011111011111100s
166	1	42	1	1	42	1	16	011111011111101s
167	1	43	1	1	43	1	16	011111101111100s
168	1	44	1	1	44	1	16	011111101111101s
169	ESCAPE						5	0000s

ESCAPE code is added at the beginning and the end of these fixed-length codes for realizing two-way decode as shown below. A marker bit is inserted before and after the 11-bit-LEVEL in order to avoid the resync_marker emulation.

ESCAPE	LAST	RUN	marker bit	LEVEL	marker bit	ESCAPE
00001	x	xxxxxx	1	xxxxxxxxxxx	1	0000s

NOTE — There are two types for ESCAPE added at the end of these fixed-length codes, and codewords are “0000s”. Also, S=0 : LEVEL is positive and S=1 : LEVEL is negative.
"

In Subclause B.2.1, Arithmetic decoding for still texture object, replace the following:
"
```
static void update_model (ac_model *acm, int sym)
{
    int i;

    if (acm->cfreq[0]==Max_frequency) {
        int cum = 0;
        acm->cfreq[acm->nsym] = 0;
        for (i = acm->nsym-1; i>=0; i--) {
            acm->freq[i] = ((int)acm->freq[i] + 1) / 2;
            cum += acm->freq[i];
            acm->cfreq[i] = cum;
        }
    }

    acm->freq[sym] += 1;
    for (i=sym; i>=0; i--)
        acm->cfreq[i] += 1;

    return;
}
```
"
with
"
```
static void update_model (ac_model *acm, int sym)
{
    int i, Max_frequency =127;

    if (acm->cfreq[0]==Max_frequency) {
        int cum = 0;
        acm->cfreq[acm->nsym] = 0;
        for (i = acm->nsym-1; i>=0; i--) {
            acm->freq[i] = ((int)acm->freq[i] + 1) / 2;
            cum += acm->freq[i];
            acm->cfreq[i] = cum;
        }
    }

    acm->freq[sym] += 1;
    for (i=sym; i>=0; i--)
        acm->cfreq[i] += 1;
```

```
    return;
}
"
```

In Subclause B.2.1, Arithmetic decoding for still texture object, remove the following paragraphs:

"

The bits_plus_follow function mentioned above calls another function, output_bit. They are:

```
static void output_bit (ac_encoder *ace, int bit) {
    ace->buffer <<= 1;
    if (bit)
        ace->buffer |= 0x01;

    ace->bits_to_go -= 1;
    ace->total_bits += 1;
    if (ace->bits_to_go==0)  {

        if (ace->bitstream) {
            if (ace->bitstream_len >= MAX_BUFFER)
                if ((ace->bitstream = (uChar *)realloc(ace->bitstream, sizeof(uChar)*
                    (ace->bitstream_len/MAX_BUFFER+1)*MAX_BUFFER))==NULL) {
                    fprintf(stderr, "Couldn't reallocate memory for ace->bitstream in output_bit.\n");
                    exit(-1);
                }

            ace->bitstream[ace->bitstream_len++] = ace->buffer;
        }
        ace->bits_to_go = 8;
    }

    return;
}

static void bit_plus_follow (ac_encoder *ace, int bit) {
    output_bit (ace, bit);
    while (ace->fbits > 0)  {
        output_bit (ace, !bit);
        ace->fbits -= 1;
    }

    return;
}
"
```

In Clause D.2, Video Rate Buffer Model Definition, replace the following:

"

3. The instantaneous video object layer channel bit rate seen by the encoder is denoted by $R_{vol}(t)$ in bits per second. If the bit_rate field in the VOL header is present, it defines a peak rate (in units of 400 bits per second; a value of 0 is forbidden) such that $R_{vol}(t) <= 400 \times$ bit_rate The bits related to the initial I-VOP in the elementary stream for basic sprite sequences are ignored for the calculation of the peak rate. Note that $R_{vol}(t)$ counts only visual syntax for the current VOL (refer to the definition of d_i below). If the channel is a serial time mutiplex containing other VOLs or as defined by ISO/IEC 14496-1 with a total instantaneous channel rate seen by the encoder of R(t), then

$$R_{vol}(t) = \begin{cases} R(t) & \text{if } t \in \{\text{channel bit duration of a bit from VOL } vol\} \\ 0 & \text{otherwise} \end{cases}$$

