INTERNATIONAL STANDARD ISO/IEC 14496-2:2004

TECHNICAL CORRIGENDUM 2

Published 2007-12-01

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 2

Technologies de l'information — Codage des objets audiovisuels —

Partie 2 : Codage visuel

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 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 Clause 2, "Normative references", remove the following:

IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990

ICS 35.040 **Ref. No. ISO/IEC 14496-2:2004/Cor.2:2007(E)**

Published in Switzerland

In Clause 2, "Normative references", add the following:

ISO/IEC 23002-1:2006 *Information technology – MPEG video technologies – Part 1: Accuracy requirements for implementation of integer-output 8x8 inverse discrete cosine transform*

In 5.4, "Arithmetic precision", replace the following:

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

with:

(a) Where an arithmetically-precise result is not fully specified, such as in the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values.

In 7.4.4.5, "Mismatch control", replace the following:

NOTE 2 Warning. Small non-zero inputs to the IDCT may result in zero output for compliant IDCTs. If this occurs in an encoder, mismatch may occur in some pictures in a decoder that uses a different compliant IDCT. An encoder should avoid this problem and may do so by checking the output of its own IDCT. It should ensure that it never inserts any non-zero coefficients into the bitstream when the block in question reconstructs to zero through its own IDCT function. If this action is not taken by the encoder, situations can arise where large and very visible mismatches between the state of the encoder and decoder occur.

with:

NOTE 2 Warning: Small non-zero inputs to the IDCT may result in all-zero output for some IDCT implementations that conform to the requirements specified in Annex A. If this occurs in an encoder, a mismatch may occur in decoders that use a different conforming IDCT implementation than the one used in modelling the decoding process within the encoder. An encoder should avoid this problem and may do so by checking the output of its own IDCT implementation. It should ensure that it never inserts any non-zero coefficients into the bitstream when the block in question reconstructs to zero through the encoder's own IDCT function implementation. If this action is not taken by the encoder, situations can arise where large and very visible mismatches between the state of the encoder and decoder occur.

In 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: $-2^{bits\_per\_pixel} \leq f[y][x] \leq 2^{bits\_per\_pixel} - 1$, for all *x*, *y*.

with:

Once the DCT coefficients, F[u][v] are reconstructed, an approximate inverse DCT function that conforms to the accuracy requirements specified in Annex A shall be applied to obtain the integer inverse transformed values *f*[*y*][*x*].

At the end of 7.6, "Motion compensation decoding", just prior to 7.6.1, insert the following:

For control of accumulation of IDCT mismatch error, it is a requirement of bitstream conformance that each macroblock shall be intra-coded at least once within each series of 132 times that the macroblock in its

position is coded in a P-VOP without an intervening I-VOP. For purposes of counting the number of times a macroblock is coded in P-VOPs, a skipped macroblock is not considered to be a coded macroblock.

NOTE When ¼ pel motion compensation is used (i.e. operation in "quarter sample mode" as specified when quarter_sample == 1), IDCT mismatch may be scaled and propagated by the associated 8-tap position interpolation filtering. Even when an IDCT that conforms to the requirements specified in Annex A is used, this can result in obviously visible distortion after performing repeated filtering. These artifacts typically become most serious when the QP value is small and the motion vector components that are applied are fractional valued in both the vertical and horizontal dimensions. Application of a shortened period of intra refresh during encoding is therefore advised under such conditions to mitigate this phenomenon.

In 7.16.4.3.4, "Mismatch control", replace the following:

NOTE 2 Warning. Small non-zero inputs to the IDCT may result in zero output for compliant IDCTs. If this occurs in an encoder, mismatch may occur in some pictures in a decoder that uses a different compliant IDCT. An encoder should avoid this problem and may do so by checking the output of its own IDCT. It should ensure that it never inserts any non-zero coefficients into the bitstream when the block in question reconstructs to zero through its own IDCT function. If this action is not taken by the encoder, situations can arise where large and very visible mismatches between the state of the encoder and decoder occur.

with:

NOTE 2 Warning: Small non-zero inputs to the IDCT may result in all-zero output for some IDCT implementations that conform to the requirements specified in Annex A. If this occurs in an encoder, a mismatch may occur in decoders that use a different conforming IDCT implementation than the one used in modelling the decoding process within the encoder. An encoder should avoid this problem and may do so by checking the output of its own IDCT implementation. It should ensure that it never inserts any non-zero coefficients into the bitstream when the block in question reconstructs to zero through the encoder's own IDCT function implementation. If this action is not taken by the encoder, situations can arise where large and very visible mismatches between the state of the encoder and decoder occur.

Replace 7.16.4.4, "Inverse DCT", which states as follows:

7.16.4.4 Inverse DCT

Once the DCT coefficients, F[u][v] are reconstructed, an IDCT transform that conforms to the specifications of Annex A shall be applied to obtain the inverse transformed values, $f[y][x]$. In the case of mpeg2_stream==0, the decimal point of F[u][v] is shifted 3bits to the left in the binary scale in order to adjust the decimal point of the IDCT input. In the case of mpeg2_stream==1, the reconstructed coefficients are directly input to an IDCT function without the shift process. The inverse transformed values shall be saturated so that: $-2^{bits\_per\_pixel} \leq f[y][x] \leq 2^{bits\_per\_pixel} - 1$, for all x, y.

with:

7.16.4.4 Inverse DCT

Once the DCT coefficients, F[u][v] are reconstructed, an approximate inverse DCT function that conforms to the accuracy requirements specified in Annex A shall be applied to obtain the integer inverse transformed values $f[y][x]$. In the case of mpeg2_stream==0, the binary point of F[u][v] is shifted 3 bits to the left in the binary scale in order to adjust the binary point of the IDCT input. In the case of mpeg2_stream==1, the reconstructed coefficients are directly input to an IDCT function without the shift process.

Replace A.1, "Discrete cosine transform for video texture", which states as follows:

A.1 Discrete cosine transform for video texture

The NxN two dimensional DCT is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

with u, v, x, y = 0, 1, 2, … N-1

where x, y are spatial coordinates in the sample domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

If each pixel is represented by *n* bits per pixel, the input to the forward transform and output from the inverse transform is represented with (*n*+1) bits. The coefficients are represented in (*n*+4) bits. The dynamic range of the DCT coefficients is [-2^{n+3}:$+2^{n+3}$-1].

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.

NOTE 1 Clause 2.3 Std 1180-1990 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. Every macroblock is required to be refreshed before it is coded 132 times as predictive macroblocks. Macroblocks in B-pictures (and skipped macroblocks in P-pictures) are excluded from the counting because they do not lead to the accumulation of mismatch errors. This requirement is the same as indicated in 1180-1990 for visual telephony according to ITU-T Recommendation H.261.

NOTE 2 Whilst the IEEE IDCT standard mentioned above is a necessary condition for the satisfactory implementation of the IDCT function it should be understood that this is not sufficient. In particular attention is drawn to the following sentence from subclause 5.4: "Where arithmetic precision is not specified, such as the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values."

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.

with:

A.1 Discrete cosine transform for video texture

The N × N two-dimensional mathematical real-number IDCT is defined as:

$$f(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

with x, y, u, v = 0, 1, 2, … N – 1

where x, y are spatial coordinates in the sample domain

u, v are coordinates in the transform domain

f(x, y) and F(u, v) are real numbers for each pair of values (x, y) and (u, v)

π is Archimedes' constant 3,141 592 653 589 793 238 462 643 …

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u = 0 \text{ or } v = 0, \text{ respectively} \\ 1 & \text{otherwise} \end{cases}$$

The N × N two-dimensional mathematical real-number DCT is defined as:

$$F(u,v) = \frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

where x, y, u, v, f (x, y), F(u, v), π, C(u), and C(v) are defined as given above for the IDCT definition.

The definition of the DCT (also called forward DCT) is purely informative. The forward DCT is not used by the decoding process specified in this International Standard.

For purposes of this Specification, the value of N shall be considered equal to 8.

The mathematical integer-number IDCT is defined as:

f '(x, y) = round(f(x, y))

with f(x, y) produced by the mathematical real-number IDCT as specified above for each value of x and y, where round() denotes rounding to the nearest integer, with half-integer values rounded away from zero. No clamping or saturation is performed.

The IDCT function used in the decoding process for computation of the integer values f[y][x] may use any method of integer approximation of the mathematical integer-number IDCT results f '(x, y), provided that it shall conform to all requirements specified in the main body of ISO/IEC 23002-1:2006 and to the additional requirements specified in ISO/IEC 23002-1:2006, Annexes A and B. The parameter *B* of ISO/IEC 23002-1:2006 shall be considered equal to bits_per_pixel.

NOTE 1 In addition to the above requirement, it is desirable that the integer output of the IDCT function f[y][x] used in the decoding process additionally produces output that is as close as feasible to the result of the mathematical integer-number IDCT f '(x, y) for input values causing one or more elements f '(x, y) of the output of the mathematical integer-number IDCT to somewhat exceed the range of $[-384*2^{(B-8)}, (384*2^{(B-8)})-1]$ for *B* = bits_per_pixel.

NOTE 2 Whilst conformance to ISO/IEC 23002-1:2006 and its Annexes A and B as mentioned above are a necessary condition for the satisfactory implementation of the IDCT function it should be understood that this is not sufficient. In particular attention is drawn to the requirement specified in subclause 5.4: "Where an arithmetically-precise result is not fully specified, such as in the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values."

Remove A.1.1, "Discrete cosine transform for the Studio Profile".

Add a new informative Annex T, "Deprecated profiles and tools", with the following sentence at its beginning:

This annex contains all information necessary for non-normative implementation of the previously defined MPEG-4 Visual Profiles N-bit and FGS, and the related tools and object types, and for non-normative implementation of the overlapped motion compensation tool that was not included in profile definitions.

Move the following sections into Annex T in the sequence in which they are listed.

Move the following section from the end of the Introduction into Annex T as a subclause T.1:

T.1 Introduction to Fine Granularity Scalability

Two profiles are developed in response to the growing need for a video coding method for Streaming Video on Internet applications. It provides the definition and description of Advanced Simple (AS) Profile and Fine Granularity Scalable (FGS) Profile. AS Profile provides the capability to distribute single-layer frame based video at a wide range of bit rates available for the distribution of video on Internet. FGS Profile uses AS Video Object in the base layer and provides the description of two enhancement layer types - Fine Granularity Scalability (FGS) and FGS Temporal Scalability (FGST). FGS Profile allows the coverage of a wide range of bit rates for the distribution of video on Internet with the flexibility of using multiple layers, where there is a wide range of bandwidth variation.

Fine Granularity Scalability (FGS) provides quality scalability for each VOP. Figure T.1 shows a basic FGS decoder structure.

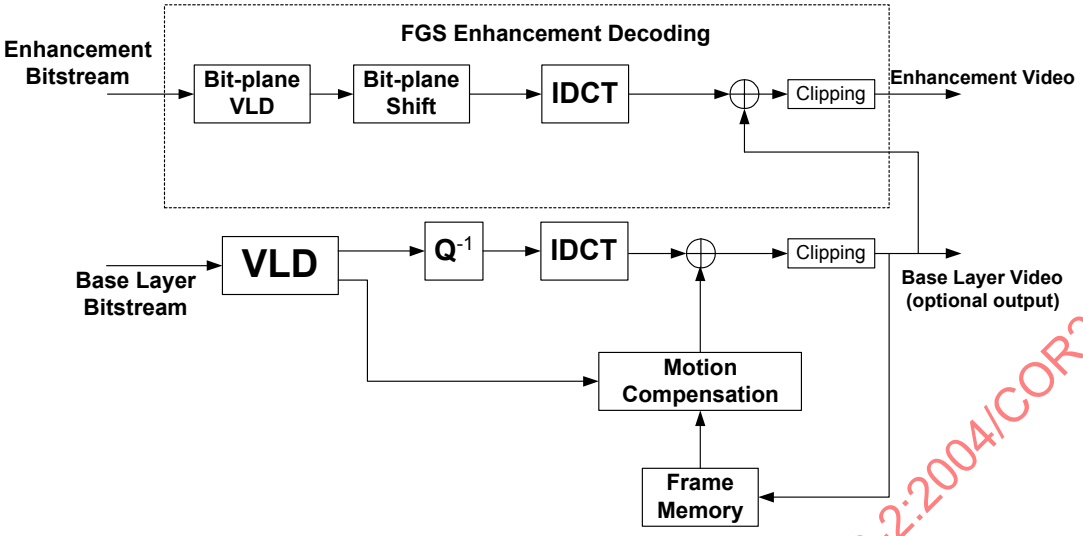

Figure T.1 — A Basic FGS Decoder Structure

To reconstruct the enhanced VOP, the enhancement bitstream is first decoded using bit-plane VLD. The decoded block-bps are used to reconstruct the DCT coefficients in the DCT domain which are then right-shifted based on the frequency weighting and selective enhancement shifting factors. The output of bit-plane shift is the DCT coefficients of the image domain residues. After the IDCT, the image domain residues are reconstructed. They are added to the reconstructed clipped base-layer pixels to reconstruct the enhanced VOP. The reconstructed enhanced VOP pixels are limited into the value range between 0 and 255 by the clipping unit in the enhancement layer to generate the final enhanced video. The reconstructed base layer video is available as an optional output since each base layer reconstructed VOP needs to be stored in the frame buffer for motion compensation.

The basic FGS enhancement layer consists of FGS VOPs that enhance the quality of the base-layer VOPs as shown in Figure T.2.

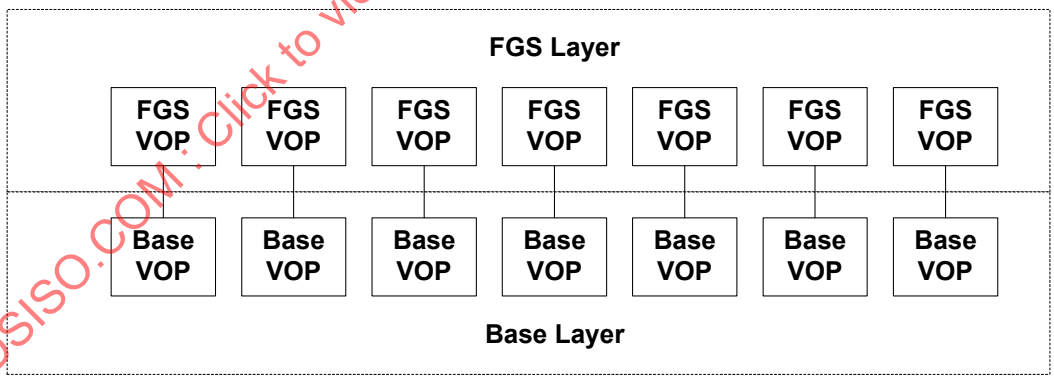

Figure T.2 — Basic FGS Enhancement Structure

When FGS temporal scalability (FGST) is used, there are two possible enhancement structures. One structure is to have two separate enhancement layers for FGS and FGST as shown in Figure T.3 and the other structure is to have one combined enhancement layer for FGS and FGST as shown in Figure T.4.

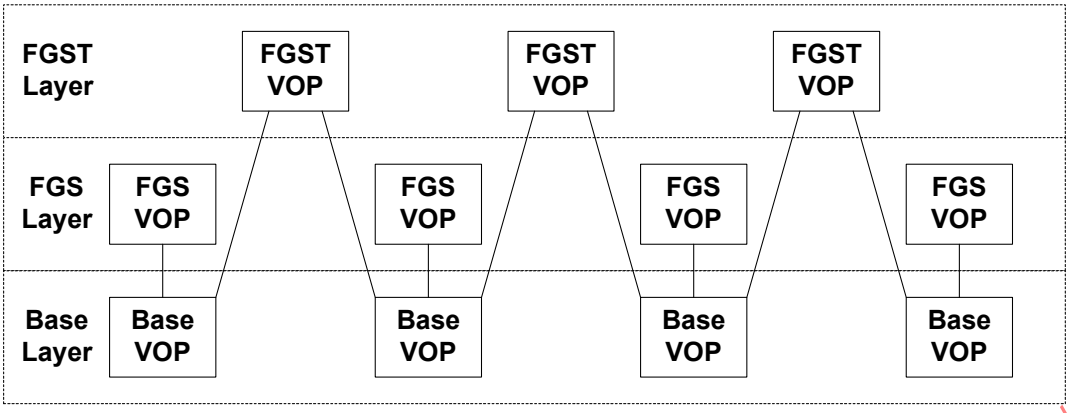

Figure T.3 — Two Separate Enhancement Layers for FGS and FGST

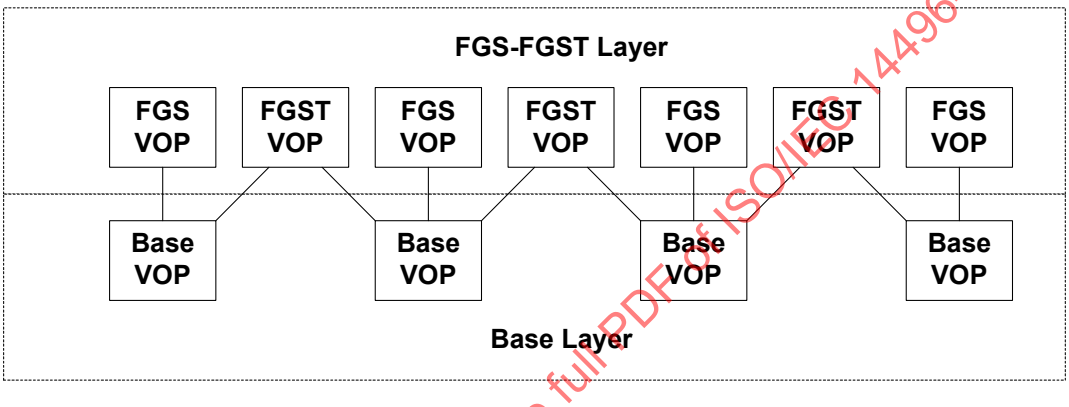

Figure T.4 — One Combined Enhancement Layer for FGS and FGST

In either one of these two structures that include FGS temporal scalability, the prediction for the FGS temporal scalable VOPs can only be from the base layer. Each FGS temporal scalable VOP has two separate parts. The first part contains motion vector data and the second part contains the DCT texture data. The syntax for the first part is similar to that in the temporal scalability described in subclause 6.2. The DCT texture data in the second part are coded using bit-plane coding in the same way as that in FGS. To distinguish the temporal scalability in subclause 6.2 and FGS temporal scalability, the FGS temporal scalability layer in Figure T.3 is called “FGST layer”. The combined FGS and FGST layer in Figure T.4 is called “FGS-FGST layer”. The “FGS VOP” shown in Figure T.3 and Figure T.4 is an fgs vop with **fgs_vop_coding_type** being ‘I’. The “FGST VOP” shown in Figure T.3 and Figure T.4 is an fgs vop with **fgs_vop_coding_type** being ‘P’ or ‘B’.

The code value of **profile_and_level_indication** in VisualObjectSequence() has been extended to include the profile and level indications for AS Profile and FGS Profile. The identifier for an enhancement layer is the syntax **video_object_type_indication** in VideoObjectLayer(). A unique code is defined for FGS Object Type to indicate that this VOL contains fgs vops. Another unique code is defined for AS Object Type to indicate that this VOL is the base-layer. There is a syntax **fgs_layer_type** in VideoObjectLayer() to indicate whether this VOL is an FGS layer as shown in Figure T.2 and Figure T.3, or an FGST layer as shown in Figure T.3, or an FGS-FGST layer as shown in Figure T.4. Similar to the syntax structure in subclause 6.2, under each VOL for FGS, there is a hierarchy of fgs vop, fgs macroblock, and fgs block. An fgs vop starts with a unique **fgs_vop_start_code**. Within each fgs vop, there are multiple vop-bps. Each vop-bp in an fgs vop starts with an **fgs_bp_start_code** whose last 5 bits indicate the ID of the vop-bp. In each fgs macroblock, there are 4 block-bps for the luminance component (Y), 2 block-bps for the two chrominance components (U and V) for the 4:2:0 chrominance format. Each block-bp is coded by VLC.

Move the following from Clause 3 into Annex T as a subclause T.2:

T.2 Terms and Definitions for Fine Granularity Scalability

T.2.1 **fgs block**: An 8-row by 8-column matrix of bits, each from one DCT coefficient at the same significant position of accuracy, or its coded representation. The usage is clear from the context.

T.2.2 **fgs macroblock**: The four block-bps of luminance component (Y) and the two (for 4:2:0 chrominance format) corresponding block-bps of chrominance components (U and V) with the same accuracy significance coming from the DCT coefficients of a macroblock. It may also be used to refer to the coded representation of the six block-bps. The usage is clear from the context.

T.2.3 **fgs macroblock number**: A number for an fgs macroblock within a vop-bp. The fgs macroblock number of the top-left fgs macroblock in each vop-bp shall be zero. The fgs macroblock number increments from left to right and from top to bottom.

T.2.4 **fgs run**: The number of '0' bits preceding a '1' bit within a block-bp.

T.2.5 **fgs temporal scalability; FGST**: A type of scalability where an enhancement layer uses predictions from sample data derived from the base layer using motion vectors. The VOP size in the enhancement layer is the same as that in the base layer. FGST is a specific type of temporal scalability where all DCT coefficients are coded using bit-plane coding as in FGS.

T.2.6 **fgs vop**: The pixel differences between the original VOP and the reconstructed VOP in the base layer. It may be used to refer to the DCT coefficients of the pixel differences or the original VOP. It may also be used to refer to the coded representation of the DCT coefficients. In the context of FGST, fgs vop refers to the original temporal scalable VOP. The usage is clear from the context.

T.2.7 **fine granularity scalability; FGS**: A type of scalability where an enhancement layer uses prediction from sample data of reconstructed VOP in the base layer. The encoded bitstream for each fgs vop can be truncated into any number of bits. The truncated bitstream for each fgs vop can be decoded to provide quality enhancement proportional to the amount of bits in the truncated bitstream of the fgs vop. The fgs vop has the same size and VOP rate as those of the base layer.

T.2.8 **vop-bp**: An array of block-bps with the same accuracy significance in an fgs vop. There are three color components (Y, U, and V) in a vop-bp. Each color component in a vop-bp consists of all the block-bps of that color.

Move the following from 6.1 into Annex T as a subclause T.3:

T.3 Structure of Coded Visual Data with Fine Granularity Scalability

In a typical application of FGS, the bitstream at the input of an FGS decoder is a truncated version of the bitstream at the output of an FGS encoder. It is likely that, at the end of each fgs vop before the next fgs_vop_start_code, only partial bits of the fgs vop are at the input of the decoder due to truncation of the fgs vop bitstream. Decoding of the truncated bitstream is not normative. An example of dealing with the truncated bitstream is described in Annex S. The FGS syntax description in this clause is for a complete bitstream without truncation.

*Replace Table 6-3 in subclause 6.2.1 with the following, and move the previous Table 6-3 into Annex T as Table T.1 in a subclause T.4 entitled "*Start Codes and Bitstream Structure with Fine Granularity Scalability*":*

Table 6-3 — Start code values

name	start code value (hexadecimal)
video_object_start_code	00 through 1F
video_object_layer_start_code	20 through 2F
reserved	30 through 3F
Forbidden	40 through 5F
reserved	60 through AF
visual_object_sequence_start_code	B0
visual_object_sequence_end_code	B1
user_data_start_code	B2
group_of_vop_start_code	B3
video_session_error_code	B4
visual_object_start_code	B5
vop_start_code	B6
slice_start_code	B7
extension_start_code	B8
Forbidden	B9
fba_object_start_code	BA
fba_object_plane_start_code	BB
mesh_object_start_code	BC
mesh_object_plane_start_code	BD
still_texture_object_start_code	BE
texture_spatial_layer_start_code	BF
texture_snr_layer_start_code	C0
texture_tile_start_code	C1
texture_shape_layer_start_code	C2
stuffing_start_code	C3
reserved	C4-C5
System start codes (see note)	C6 through FF
NOTE System start codes are defined in ISO/IEC 14496-1:2001.	

Replace 6.2.3 with the following, and move the previous 6.2.3 into Annex T as a subclause T.5 entitled "Syntax of Video Object Layer with Fine Granularity Scalability"*:*

6.2.3 Video Object Layer

VideoObjectLayer() {	No. of bits	Mnemonic
if(next_bits() == video_object_layer_start_code) {		
short_video_header = 0		
video_object_layer_start_code	32	bslbf
random_accessible_vol	1	bslbf
video_object_type_indication	8	uimsbf
is_object_layer_identifier	1	uimsbf
if (is_object_layer_identifier) {		

video_object_layer_verid	4	uimsbf
video_object_layer_priority	3	uimsbf
}		
aspect_ratio_info	4	uimsbf
if (aspect_ratio_info == “extended_PAR”) {		
par_width	8	uimsbf
par_height	8	uimsbf
}		
vol_control_parameters	1	bslbf
if (vol_control_parameters) {		
chroma_format	2	uimsbf
low_delay	1	uimsbf
vbv_parameters	1	blsbf
if (vbv_parameters) {		
first_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
latter_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
first_half_vbv_buffer_size	15	uimsbf
marker_bit	1	bslbf
latter_half_vbv_buffer_size	3	uimsbf
first_half_vbv_occupancy	11	uimsbf
marker_bit	1	blsbf
latter_half_vbv_occupancy	15	uimsbf
marker_bit	1	blsbf
}		
}		
video_object_layer_shape	2	uimsbf
if (video_object_layer_shape == "grayscale" && video_object_layer_verid != ‘0001’)		
video_object_layer_shape_extension	4	uimsbf
marker_bit	1	bslbf
vop_time_increment_resolution	16	uimsbf
marker_bit	1	bslbf
fixed_vop_rate	1	bslbf
if (fixed_vop_rate)		
fixed_vop_time_increment	1-16	uimsbf
if (video_object_layer_shape != “binary only”) {		
if (video_object_layer_shape == “rectangular”) {		
marker_bit	1	bslbf
video_object_layer_width	13	uimsbf
marker_bit	1	bslbf
video_object_layer_height	13	uimsbf
marker_bit	1	bslbf
}		
interlaced	1	bslbf

obmc_disable	1	bslbf
if (video_object_layer_verid == '0001')		
sprite_enable	1	bslbf
else		
sprite_enable	2	uimsbf
if (sprite_enable== "static" ll sprite_enable == "GMC") {		
if (sprite_enable != "GMC") {		
sprite_width	13	uimsbf
marker_bit	1	bslbf
sprite_height	13	uimsbf
marker_bit	1	bslbf
sprite_left_coordinate	13	simsbf
marker_bit	1	bslbf
sprite_top_coordinate	13	simsbf
marker_bit	1	bslbf
}		
no_of_sprite_warping_points	6	uimsbf
sprite_warping_accuracy	2	uimsbf
sprite_brightness_change	1	bslbf
if (sprite_enable != "GMC")		
low_latency_sprite_enable	1	bslbf
}		
if (video_object_layer_verid != '0001' && video_object_layer_shape != "rectangular")		
sadct_disable	1	bslbf
not_8_bit	1	bslbf
if (not_8_ bit) {		
quant_precision	4	uimsbf
bits_per_pixel	4	uimsbf
}		
if (video_object_layer_shape=="grayscale") {		
no_gray_quant_update	1	bslbf
composition_method	1	bslbf
linear_composition	1	bslbf
}		
quant_type	1	bslbf
if (quant_type) {		
load_intra_quant_mat	1	bslbf
if (load_intra_quant_mat)		
intra_quant_mat	8*[2-64]	uimsbf
load_nonintra_quant_mat	1	bslbf
if (load_nonintra_quant_mat)		
nonintra_quant_mat	8*[2-64]	uimsbf
if(video_object_layer_shape=="grayscale") {		
for(i=0; i<aux_comp_count; i++) {		
load_intra_quant_mat_grayscale	1	bslbf

if(load_intra_quant_mat_grayscale)		
intra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
load_nonintra_quant_mat_grayscale	1	bslbf
if(load_nonintra_quant_mat_grayscale)		
nonintra_quant_mat_grayscale[i]	8*[2-64]	uimsbf
}		
}		
}		
if (video_object_layer_verid != '0001')		
quarter_sample	1	bslbf
complexity_estimation_disable	1	bslbf
if (!complexity_estimation_disable)		
define_vop_complexity_estimation_header()		
resync_marker_disable	1	bslbf
data_partitioned	1	bslbf
if(data_partitioned)		
reversible_vlc	1	bslbf
if(video_object_layer_verid != '0001') {		
newpred_enable	1	bslbf
if (newpred_enable) {		
requested_upstream_message_type	2	uimsbf
newpred_segment_type	1	bslbf
}		
reduced_resolution_vop_enable	1	bslbf
}		
scalability	1	bslbf
if (scalability) {		
hierarchy_type	1	bslbf
ref_layer_id	4	uimsbf
ref_layer_sampling_direc	1	bslbf
hor_sampling_factor_n	5	uimsbf
hor_sampling_factor_m	5	uimsbf
vert_sampling_factor_n	5	uimsbf
vert_sampling_factor_m	5	uimsbf
enhancement_type	1	bslbf
if(video_object_layer == "binary" && hierarchy_type== '0') {		
use_ref_shape	1	bslbf
use_ref_texture	1	bslbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
}		

else {		
if(video_object_layer_verid !=”0001”) {		
scalability	1	bslbf
if(scalability) {		
ref_layer_id	4	uimsbf
shape_hor_sampling_factor_n	5	uimsbf
shape_hor_sampling_factor_m	5	uimsbf
shape_vert_sampling_factor_n	5	uimsbf
shape_vert_sampling_factor_m	5	uimsbf
}		
}		
resync_marker_disable	1	bslbf
}		
next_start_code()		
while (next_bits()== user_data_start_code){		
user_data()		
}		
if (sprite_enable == “static” && !low_latency_sprite_enable)		
VideoObjectPlane()		
do {		
if (next_bits() == group_of_vop_start_code)		
Group_of_VideoObjectPlane()		
VideoObjectPlane()		
if ((preceding_vop_coding_type == "B" || preceding_vop_coding_type == "S" || video_object_layer_shape != "rectangular") && next_bits() == stuffing_start_code) {		
stuffing_start_code	32	bslbf
while (next_bits() != ‘0000 0000 0000 0000 0000 0001’)		
stuffing_byte	8	bslbf
}		
} while ((next_bits() == group_of_vop_start_code) || (next_bits() == vop_start_code))		
} else {		
short_video_header = 1		
do {		
video_plane_with_short_header()		
} while(next_bits() == short_video_start_marker)		
}		
}		
NOTE Preceding_vop_coding_type has the same value as vop_coding_type in the immediately preceding VideoObjectPlane() in the decoding order.		

Move subclause 6.2.14 (and its subordinate subclauses) into Annex T as a subclause T.6 (and subordinate subclauses renumbered correspondingly, retaining the same subclause titles).

*Replace the corresponding table in subclause 6.3.3 with the following, and move the previous table into Annex T as Table T.2 in a subclause T.7 entitled "*Semantics of Video Object Layer with Fine Granularity Scalability, N-Bit Objects, and Overlapped Motion Compensation"*:*

Table 6-11 — FLC table for video_object_type indication

Video Object Type	Code
Reserved	00000000
Simple Object Type	00000001
Simple Scalable Object Type	00000010
Core Object Type	00000011
Main Object Type	00000100
Forbidden	00000101
Basic Anim. 2D Texture	00000110
Anim. 2D Mesh	00000111
Simple Face	00001000
Still Scalable Texture	00001001
Advanced Real Time Simple	00001010
Core Scalable	00001011
Advanced Coding Efficiency	00001100
Advanced Scalable Texture	00001101
Simple FBA	00001110
Simple Studio	00001111
Core Studio	00010000
Advanced Simple	00010001
Forbidden	00010010
Reserved	00010011 - 11111111

Move the following paragraphs from subclause 6.3.3 into Annex T subclause T.7 and renumber appropriately:

fgs_layer_type – This is a 2-bit code indicating whether this layer is FGS only, FGST only, or a combination of FGS and FGST. Table 6-12 shows the codes and the meanings.

Table T.3 — Code for fgs_layer_type

Code	Meaning
00	reserved
01	FGS
10	FGST
11	FGS-FGST

video_object_layer_priority – This is a 3-bit code which specifies the priority of the video object layer. It takes values between 1 and 7, with 1 representing the highest priority and 7 the lowest priority. The value of zero is reserved. For the transmission of FGS and FGST in two VOLs, the relative transmission priority of an FGS VOL vs. that of an FGST VOL can be specified by setting this parameter in the FGS VOL relative to the same parameter in the FGST VOL.

fgs_ref_layer_id – This is a 4-bit unsigned integer with value between 0 and 15. It indicates the layer to be used as reference for prediction in the case of fgs_layer_type being FGST or FGS_FGST.

fgs_frequency_weighting_enable – This is a one-bit flag to indicate that frequency weighting is used in this VOL, when set to '1'. Otherwise, when this flag is set to '0', frequency weighting is not used. The default frequency weighting matrix is an all zero matrix when fgs_frequency_weighting_enable is '1'.

load_fgs_frequency_weighting_matrix – This is a one-bit flag which is set to '1' when fgs_frequency_weighting_matrix follows. If it is set to '0' then the default frequency weighting matrix is used.

fgs_frequency_weighting_matrix – This is a list of 2 to 64 three-bit unsigned integers. The integers are in zigzag scan order representing the fgs_frequency_weighting_matrix. A value of 0 indicates that no more values are transmitted and the remaining, non-transmitted values are set to zero.

fgst_frequency_weighting_enable – This is a one-bit flag to indicate that frequency weighting is used in this VOL, when set to '1'. Otherwise, when this flag is set to '0', frequency weighting is not used. The default frequency weighting matrix is an all zero matrix when fgst_frequency_weighting_enable is '1'.

load_fgst_frequency_weighting_matrix – This is a one-bit flag which is set to '1' when fgst_frequency_weighting_matrix follows. If it is set to '0' then the default matrix is used.

fgst_frequency_weighting_matrix – This is a list of 2 to 64 three-bit unsigned integers. The integers are in zigzag scan order representing the fgst_frequency_weighting_matrix. A value of 0 indicates that no more values are transmitted and the remaining, non-transmitted values are set to zero.

fgs_resync_marker_disable – This is a one-bit flag which when set to '1' indicates that there is no fgs_resync_marker in coded fgs vops of this VOL. When this flag is set to '0', it indicates that fgs_resync_marker may be used in coded fgs vops of this VOL.

*Replace the paragraph in subclause 6.3.3 that starts with "***obmc_disable***" with the following, and move the previous text into Annex T subclause T.7:*

obmc_disable: This is a one-bit flag which shall be set to '1'.

*Replace the paragraph in subclause 6.3.3 that starts with "***not_8_bit***" with the following, and move the previous text into Annex T subclause T.7:*

not_8_bit: This one bit flag shall be set to '0'.

*Replace the paragraph in subclause 6.3.3 that starts with "***quarter_sample***" with the following, and move the previous text into Annex T subclause T.7:*

quarter_sample: This is a one-bit flag which when set to '0' indicates that half sample mode and when set to '1' indicates that quarter sample mode shall be used for motion compensation of the luminance component.

*Replace the corresponding table in subclause 6.3.5.2 with the following, and move the previous table into Annex T as Table T.4 in a subclause T.8 entitled "*Video Plane with Short Header Interpretation with Overlapped Motion Compensation*":*

Table 6-28 — Fixed Settings for video_plane_with_short_header()

Parameter	Value
video_object_layer_shape	“rectangular”
quant_type	0
resync_marker_disable	1
data_partitioned	0
block_count	6
reversible_vlc	0
vop_rounding_type	0
vop_fcode_forward	1
vop_coded	1
interlaced	0
complexity_estimation_disable	1
use_intra_dc_vlc	0
scalability	0
not_8_bit	0
bits_per_pixel	8
colour_primaries	1
transfer_characteristics	1
matrix_coefficients	6

*Replace the corresponding table in subclause 6.3.13.3 with the following, and move the previous table into Annex T as Table T.5 in a subclause T.9 entitled "*Studio Video Object Layer Interpretation with Fine Granularity Scalability and N-bit Objects".

Table 6-81 — FLC table for video_object_type indication

Video Object Type	Code
Reserved	00000000
Simple Object Type	00000001
Simple Scalable Object Type	00000010
Core Object Type	00000011
Main Object Type	00000100
Forbidden	00000101
Basic Anim. 2D Texture	00000110
Anim. 2D Mesh	00000111
Simple Face	00001000
Still Scalable Texture	00001001
Advanced Real Time Simple	00001010
Core Scalable	00001011
Advanced Coding Efficiency	00001100
Advanced Scalable Texture	00001101
Simple Studio Object Type	00001110
Core Studio Object Type	00001111
Reserved	00010000 - 11111111

Move subclause 6.3.14 into Annex T as a subclause T.10 (without changing the title of the subclause).

*Replace the first paragraph in subclause 7.5.5.5 with the following, and move the previous paragraph into Annex T as a subclause T.11 entitled "*Inter Macroblocks and Motion Compensation with Overlapped Motion Compensation"*:*

Motion compensation is carried out for P-, B-, and S(GMC)-VOPs, using the 8x8 or 16x16 luminance motion vectors, or the warping of the previous decoded VOP, in the same way as for luminance data. Forward, backward, bidirectional and direct mode motion compensation are used for B-VOPs. Where the luminance motion vectors are not present because the texture macroblock is skipped, the exact same style of non-coded motion compensation used for luminance is applied to the alpha data. Note that this does not imply that the alpha macroblock is skipped, because an error signal to update the resulting motion compensated alpha macroblock may still be present if indicated by coda_pb. When the co-located P-VOP texture macroblock is skipped for B-VOPs, then the alpha macroblock is assumed to be skipped with no syntax transmitted.

*Replace the last paragraph in subclause 7.6.5 with the following, and move the previous paragraph into Annex T as a subclause T.12 entitled "*Vector decoding processing and motion-compensation in progressive P- and S(GMC)-VOP with Overlapped Motion Compensation*":*

Half sample values are found using bilinear interpolation as described in subclause 7.6.2. The prediction for chrominance is obtained by applying the motion vector MVD_{CHR} to all pixels in the two chrominance blocks.

Move subclause 7.6.6 into Annex T as a subclause T.13 (without changing its title).

Move the following sentence from subclause 7.6.9 into Annex T as a subclause T.14 entitled "Motion compensation in non-scalable progressive B-VOPs for Overlapped Motion Compensation"*:*

In B-VOPs the overlapped motion compensation (OBMC) is not employed.

*Move the following sentence from subclause 7.7.2.1 into Annex T as a subclause T.15 entitled "*Motion vector decoding in P- and S(GMC)-VOP for Overlapped Motion Compensation*":*

In the case that obmc_disable is “0”, the OBMC is not applied if the current MB is field-predicted.

Move subclause 7.17 (and its subordinate subclauses) into Annex T as a subclause T.16 (without changing the title of the subclauses).

*Replace the corresponding tables in subclause 9.1 with the following, and move the previous tables into Annex T as Tables T.6, T.7, and T.8 in a subclause T.17 entitled "*Visual Object Types with N-bit Objects and Overlapped Motion Compensation*":*

Table 9-1 — Tools for Version 1 Visual Object Types

Visual Tools	Visual Object Types								
	Simple	**Core**	**Main**	**Simple Scalable**		**Animated 2D Mesh**	**Basic Animated Texture**	**Still Scalable Texture**	**Simple Face**
Basic • I-VOP • P-VOP • AC/DC Prediction • 4-MV, Unrestricted MV	X	X	X	X		X			
Error resilience • GOB Resynchronization • Data Partitioning • Reversible VLC	X	X	X	X		X			
Short Header	X	X	X			X			
B-VOP		X	X	X		X			
Method 1/Method 2 Quantisation		X	X			X			
P-VOP based temporal scalability • Rectangular • Arbitrary Shape		X	X			X			
Binary Shape		X	X			X	X		
Grey Shape			X						
Interlace			X						
Sprite			X						
Temporal Scalability (Rectangular)				X					
Spatial Scalability (Rectangular)				X					
N-Bit									
Scalable Still Texture						X	X	X	
2D Dynamic Mesh with uniform topology						X	X		
2D Dynamic Mesh with Delaunay topology						X			
Facial Animation Parameters									X

Table 9-2 — Tools for Version 2 Visual Object Types

Visual Tools	Visual Object Types				
	Advanced Real Time Simple	**Advanced Coding Efficiency**	**Advanced Scalable Texture**	**Core Scalable**	**Simple FBA**
Basic • I-VOP • P-VOP • AC/DC Prediction • 4-MV, Unrestricted MV	X	X		X	
Error resilience • GOB Resynchronization • Data Partitioning • Reversible VLC	X	X		X	
Short Header	X	X		X	
B-VOP		X		X	
Method 1/Method 2 Quantisation		X		X	
P-VOP based temporal scalability • Rectangular • Arbitrary Shape		X		X	
Binary Shape		X		X	
Grey Shape		X			
Interlace		X			
Sprite					
Temporal Scalability (Rectangular)				X	
Spatial Scalability (Rectangular)				X	
Scalable Still Texture			X		
2D Dynamic Mesh with uniform topology					
2D Dynamic Mesh with Delaunay topology					
Facial Animation Parameters					X
Body_Animation_Parameters					X
Reduced Resolution VOP	X				
NEWPRED	X				
Global Motion Compensation		X			

Quarter-pel Motion Compensation		X			
SA-DCT		X			
Error Resilience for Visual Texture Coding			X		
Wavelet Tiling			X		
Scalable Shape Coding for Still Texture			X		
Object Based Temporal Scalability				X	
Object Based Spatial Scalability				X	

Table 9-4 — Tools for FGS Visual Object Types

Visual Tools	**Visual Object Types**	
	Advanced Simple	
I-VOP	X	
P-VOP	X	
B-VOP	X	
DC Prediction	X	
AC Prediction	X	
4-MV, Unrestricted MV	X	
GOB Resynchronization	X	
Data Partitioning	X	
Reversible VLC	X	
Short Header	X	
Method 1/Method 2 Quantisation	X	
Interlace	X	
Global Motion Compensation	X	
Quarter-pel Motion Compensation	X	

NOTE 1 The interlace tools are not used for levels L0, L1, L2, and L3 of AS.

*Replace the corresponding table in subclause 9.2 with the following, and move the previous table into Annex T as Table T.9 in a subclause T.18 entitled "*Visual Profiles with Fine Granularity Scalability and N-bit Objects*":*

Table 9-5 — Version 1 Visual Profiles

	Object Types / Profiles	Simple	Core	Main	Simple Scalable		Animated 2D Mesh	Basic Animated Texture	Scalable Texture	Simple Face
1.	Simple	X								
2.	Simple Scalable	X			X					
3.	Core	X	X							
4.	Main	X	X	X					X	
5.	Hybrid	X	X				X	X	X	X
6.	Basic Animated Texture							X	X	X
7.	Scalable Texture								X	
8.	Simple FA									X

Replace the corresponding sentence and table in subclause 9.2 with the following, and move the previous sentence and table into Annex T subclause T.18, renumbering the prior table as Table T.10:

Table 9-8 shows the definition of Advanced Simple Profile.

Table 9-8 — Definition of Advanced Simple Profile

ID	Object Types / Profiles	Simple	Advanced Simple	
AS	Advanced Simple	X	X	

Move subclause B.4 of Annex B into Annex T as a subclause T.19 (without changing the title of the subclause).

*Replace the corresponding tables in Annex G, and move the previous tables into Annex T as Tables T.11 and T.12 in a subclause T.20 entitled "*Profile and Level Indications with Fine Granularity Scalability and N-bit Objects, and Overlapped Motion Compensation*":*

Table G.1 — FLC table for profile_and_level_indication

Profile/Level	Code
Reserved	00000000
Simple Profile/Level 1	00000001
Simple Profile/Level 2	00000010
Simple Profile/Level 3	00000011
Reserved	00000100 – 00000111
Simple Profile/Level 0	00001000
Reserved	00001001 - 00001111