



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

Published 2000-08-15

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

**Information technology — Coding of audio-visual objects —**  
**Part 2:**  
**Visual**

**TECHNICAL CORRIGENDUM 1**

*Technologies de l'information — Codage des objets audiovisuels —*  
*Partie 2: Codage visuel*

*RECTIFICATIF TECHNIQUE 1*

Technical Corrigendum 1 to 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*.



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 ) {<br><b>data_element</b><br>...<br>}                                        | 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 {<br><b>data_element</b><br>...<br>} while ( condition )                                     | The data element always occurs at least once.<br><br>The data element is repeated until the condition is not true.                                                                                                                                                      |
| if ( condition ) {<br><b>data_element</b><br>...<br>} else {<br><b>data_element</b><br>...<br>} | If the condition is true, then the first group of data elements occurs next in the data stream.<br><br>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++ ) {<br><b>data_element</b><br>...<br>}                                  | 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 ) {<br><b>data_element</b><br>...<br>}    | 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 {<br><b>data_element</b><br>...<br>} while ( condition ) | The data element always occurs at least once.<br><br>The data element is repeated until the condition is not true.                      |
| do {<br>...<br>continue<br>...                              | The continue continues execution of the next repetition of the nearest while-do loop.                                                   |

|                                                                                                                                    |                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>} while ( condition )</code>                                                                                                 |                                                                                                                                                                                                                                                                         |
| <code>if ( condition ) {<br/>    <b>data_element</b><br/>    ...<br/>} else {<br/>    <b>data_element</b><br/>    ...<br/>}</code> | <p>If the condition is true, then the first group of data elements occurs next in the data stream.</p> <p>If the condition is not true, then the second group of data elements occurs next in the data stream.</p>                                                      |
| <code>for ( i = m; i &lt; n; i++ ) {<br/>    <b>data_element</b><br/>    ...<br/>}</code>                                          | 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. |
| <code>/* comment ... */</code>                                                                                                     | 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 |
|----------------------------------------------|-------------|----------|
| <b>visual_object_sequence_start_code</b>     | 32          | bslbf    |
| <b>profile_and_level_indication</b>          | 8           | uimsbf   |
| while ( next_bits()== user_data_start_code){ |             |          |
| user_data()                                  |             |          |
| }                                            |             |          |
| VisualObject()                               |             |          |
| <b>visual_object_sequence_end_code</b>       | 32          | bslbf    |
| }                                            |             |          |

"

with

"

|                                                           |             |          |
|-----------------------------------------------------------|-------------|----------|
| VisualObjectSequence() {                                  | No. of bits | Mnemonic |
| do {                                                      |             |          |
| <b>visual_object_sequence_start_code</b>                  | 32          | bslbf    |
| <b>profile_and_level_indication</b>                       | 8           | uimsbf   |
| while ( next_bits() == user_data_start_code){             |             |          |
| user_data()                                               |             |          |
| }                                                         |             |          |
| VisualObject()                                            |             |          |
| } while ( next_bits() != visual_object_sequence_end_code) |             |          |
| <b>visual_object_sequence_end_code</b>                    | 32          | bslbf    |
| }                                                         |             |          |

"

In Subclause 6.2.2.1, User data(), replace the user\_data() syntax:

"

|                                                           |             |          |
|-----------------------------------------------------------|-------------|----------|
| user_data() {                                             | No. of bits | Mnemonic |
| <b>user_data_start_code</b>                               | 32          | bslbf    |
| while( next_bits() != '0000 0000 0000 0000 0000 0001' ) { |             |          |
| <b>user_data</b>                                          | 8           | uimsbf   |
| }                                                         |             |          |
| next_start_code()                                         |             |          |
| }                                                         |             |          |

"

with

"

|                                                           |             |          |
|-----------------------------------------------------------|-------------|----------|
| user_data() {                                             | No. of bits | Mnemonic |
| <b>user_data_start_code</b>                               | 32          | bslbf    |
| while( next_bits() != '0000 0000 0000 0000 0000 0001' ) { |             |          |
| <b>user_data</b>                                          | 8           | uimsbf   |
| }                                                         |             |          |
| }                                                         |             |          |

"

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

"

|                              |    |       |
|------------------------------|----|-------|
| <b>group_vop_start_codes</b> | 32 | bslbf |
|------------------------------|----|-------|

"

with

"

|                                |    |       |
|--------------------------------|----|-------|
| <b>group_of_vop_start_code</b> | 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") { |    |        |
| <b>vop_width</b>                               | 13 | uimsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_height</b>                              | 13 | uimsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_horizontal_mc_spatial_ref</b>           | 13 | simsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_vertical_mc_spatial_ref</b>             | 13 | simsbf |
| }                                              |    |        |

with

|                                                |    |        |
|------------------------------------------------|----|--------|
| if(!sprite_enable && vop_coding_type == "I") { |    |        |
| <b>vop_width</b>                               | 13 | uimsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_height</b>                              | 13 | uimsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_horizontal_mc_spatial_ref</b>           | 13 | simsbf |
| <b>marker_bit</b>                              | 1  | bslbf  |
| <b>vop_vertical_mc_spatial_ref</b>             | 13 | simsbf |
| <b>marker_bit</b>                              | 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:

| <code>video_packet_header() {</code>                          | No. of bits | Mnemonic |
|---------------------------------------------------------------|-------------|----------|
| <code>next_resync_marker()</code>                             |             |          |
| <b>resync_marker</b>                                          | 17-23       | uimsbf   |
| <b>macroblock_number</b>                                      | 1-14        | vlcibf   |
| <code>if (video_object_layer_shape != "binary only")</code>   |             |          |
| <b>quant_scale</b>                                            | 5           | uimsbf   |
| <b>header_extension_code</b>                                  | 1           | bslbf    |
| <code>if (header_extension_code) {</code>                     |             |          |
| <code>do {</code>                                             |             |          |
| <b>modulo_time_base</b>                                       | 1           | bslbf    |
| <code>} while (modulo_time_base != '0')</code>                |             |          |
| <b>marker_bit</b>                                             | 1           | bslbf    |
| <b>vop_time_increment</b>                                     | 1-16        | bslbf    |
| <b>marker_bit</b>                                             | 1           | bslbf    |
| <b>vop_coding_type</b>                                        | 2           | uimsbf   |
| <code>if (video_object_layer_shape != "binary only") {</code> |             |          |
| <b>intra_dc_vlc_thr</b>                                       | 3           | uimsbf   |
| <code>if (vop_coding_type != "I")</code>                      |             |          |
| <b>vop_fcode_forward</b>                                      | 3           | uimsbf   |
| <code>if (vop_coding_type == "B")</code>                      |             |          |
| <b>vop_fcode_backward</b>                                     | 3           | uimsbf   |
| <code>}</code>                                                |             |          |
| <code>}</code>                                                |             |          |
| <code>}</code>                                                |             |          |

with

| <code>video_packet_header() {</code>                                                                    | No. of bits | Mnemonic |
|---------------------------------------------------------------------------------------------------------|-------------|----------|
| <code>next_resync_marker()</code>                                                                       |             |          |
| <b>resync_marker</b>                                                                                    | 17-23       | uimsbf   |
| <code>if (video_object_layer_shape != "rectangular") {</code>                                           |             |          |
| <b>header_extension_code</b>                                                                            | 1           | bslbf    |
| <code>if (header_extension_code &amp;&amp; !(sprite_enable &amp;&amp; vop_coding_type == "I")) {</code> |             |          |
| <b>vop_width</b>                                                                                        | 13          | uimsbf   |
| <b>marker_bit</b>                                                                                       | 1           | bslbf    |
| <b>vop_height</b>                                                                                       | 13          | uimsbf   |
| <b>marker_bit</b>                                                                                       | 1           | bslbf    |
| <b>vop_horizontal_mc_spatial_ref</b>                                                                    | 13          | simsbf   |
| <b>marker_bit</b>                                                                                       | 1           | bslbf    |
| <b>vop_vertical_mc_spatial_ref</b>                                                                      | 13          | simsbf   |
| <b>marker_bit</b>                                                                                       | 1           | bslbf    |
| <code>}</code>                                                                                          |             |          |
| <code>}</code>                                                                                          |             |          |
| <b>macroblock_number</b>                                                                                | 1-14        | vlcibf   |
| <code>if (video_object_layer_shape != "binary only")</code>                                             |             |          |
| <b>quant_scale</b>                                                                                      | 5           | uimsbf   |
| <code>if (video_object_layer_shape == "rectangular")</code>                                             |             |          |

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

In Subclause 6.2.5.3, Motion Shape Texture, replace the following rows 11, 12 and 13 of data\_partitioned\_i\_vop() syntax:

|                          |     |        |
|--------------------------|-----|--------|
| if (!transparent_mb()) { |     |        |
| <b>mcbpc</b>             | 1-9 | vlclbf |
| if (mb_type == 4)        |     |        |

with

|                                               |     |        |
|-----------------------------------------------|-----|--------|
| if (!transparent_mb()) {                      |     |        |
| if( video_object_layer_shape != "rectangle"){ |     |        |
| do{                                           |     |        |
| <b>mcbpc</b>                                  | 1-9 | vlclbf |
| } while( derived_mb_type == "stuffing")       |     |        |
| }else{                                        |     |        |
| <b>mcbpc</b>                                  | 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\_partitioned\_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_partitioned_p_vop()` syntax:

“

|                                           |     |        |
|-------------------------------------------|-----|--------|
| if (!transparent_mb()) {                  |     |        |
| <b>not_coded</b>                          | 1   | bslbf  |
| if (!not_coded) {                         |     |        |
| <b>mcbpc</b>                              | 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{                                                      |     |        |
| <b>not_coded</b>                                         | 1   | bslbf  |
| if (!not_coded)                                          |     |        |
| <b>mcbpc</b>                                             | 1-9 | vlclbf |
| } while( !(not_coded    derived_mb_type != “stuffing”) ) |     |        |
| }else{                                                   |     |        |
| <b>not_coded</b>                                         | 1   | bslbf  |
| if (!not_coded){                                         |     |        |
| <b>mcbpc</b>                                             | 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_partitioned_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<br>&& sprite_transmit_mode == "piece")) |     |        |
| <b>not_coded</b>                                                                      | 1   | bslbf  |
| if (!not_coded    vop_coding_type == "I") {                                           |     |        |
| <b>mcbpc</b>                                                                          | 1-9 | vlcblf |
| if (!short_video_header &&<br>(derived_mb_type == 3   <br>derived_mb_type == 4))      |     |        |

"

with

"

|                                                                                                                                                |     |        |
|------------------------------------------------------------------------------------------------------------------------------------------------|-----|--------|
| if (!transparent_mb()) {                                                                                                                       |     |        |
| if (video_object_layer_shape != "rectangular"<br>&& !(sprite_enable && low_latency_sprite_enable<br>&& sprite_transmit_mode == "update")) {    |     |        |
| do{                                                                                                                                            |     |        |
| if (vop_coding_type != "I" && !(sprite_enable<br>&& sprite_transmit_mode == "piece"))                                                          |     |        |
| <b>not_coded</b>                                                                                                                               | 1   | bslbf  |
| if (!not_coded    vop_coding_type == "I"<br>   (vop_coding_type == "S"<br>&& low_latency_sprite_enable<br>&& sprite_transmit_mode == "piece")) |     |        |
| <b>mcbpc</b>                                                                                                                                   | 1-9 | vlcblf |
| } while(!(not_coded    derived_mb_type != "stuffing"))                                                                                         |     |        |
| } else {                                                                                                                                       |     |        |
| if (vop_coding_type != "I" && !(sprite_enable<br>&& sprite_transmit_mode == "piece"))                                                          |     |        |
| <b>not_coded</b>                                                                                                                               | 1   | bslbf  |
| if (!not_coded    vop_coding_type == "I"<br>   (vop_coding_type == "S"<br>&& low_latency_sprite_enable<br>&& sprite_transmit_mode == "piece")) |     |        |
| <b>mcbpc</b>                                                                                                                                   | 1-9 | vlcblf |
| }                                                                                                                                              |     |        |

|                                                                                                                                                  |  |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| if (!not_coded    vop_coding_type == "I"<br>   (vop_coding_type == "S"<br>&& low_latency_sprite_enable<br>&& sprite_transmit_mode == "piece")) { |  |  |
| if (!short_video_header &&<br>(derived_mb_type == 3   <br>derived_mb_type == 4))                                                                 |  |  |

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

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

|                                                             |    |        |
|-------------------------------------------------------------|----|--------|
| <b>marker_bit</b>                                           | 1  | bslbf  |
| <b>object_width</b>                                         | 15 | uimsbf |
| <b>marker_bit</b>                                           | 1  | bslbf  |
| <b>object_height</b>                                        | 15 | uimsbf |
| <b>marker_bit</b>                                           | 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{                                                         |    |        |
| <b>quant_byte</b>                                           |    |        |
| } while( quant_byte >>7)                                    |    |        |
| <b>max_bitplanes</b>                                        |    |        |
| 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() {                 | <b>No. of bits</b> | <b>Mnemonic</b> |
| <b>still_texture_object_start_code</b> | 32                 |                 |
| <b>texture_object_id</b>               | 16                 | uimsbf          |
| <b>marker_bit</b>                      | 1                  | bslbf           |
| <b>wavelet_filter_type</b>             | 1                  | uimsbf          |
| <b>wavelet_download</b>                | 1                  | uimsbf          |
| <b>wavelet_decomposition_levels</b>    | 4                  | uimsbf          |
| <b>scan_direction</b>                  | 1                  | bslbf           |
| <b>start_code_enable</b>               | 1                  | bslbf           |

|                                                                   |    |        |
|-------------------------------------------------------------------|----|--------|
| <b>texture_object_layer_shape</b>                                 | 2  | uimsbf |
| <b>quantization_type</b>                                          | 2  | uimsbf |
| if (quantization_type == 2) {                                     |    |        |
| <b>spatial_scalability_levels</b>                                 | 4  | uimsbf |
| if (spatial_scalability_levels != wavelet_decomposition_levels) { |    |        |
| <b>use_default_spatial_scalability</b>                            | 1  | uimsbf |
| if (use_default_spatial_layer_size == 0)                          |    |        |
| for (i=0; i<spatial_scalability_levels – 1; i++)                  |    |        |
| <b>wavelet_layer_index</b>                                        | 4  |        |
| }                                                                 |    |        |
| }                                                                 |    |        |
| if (wavelet_download == “1”){                                     |    |        |
| <b>uniform_wavelet_filter</b>                                     | 1  | uimsbf |
| if (uniform_wavelet_filter == “1”)                                |    |        |
| download_wavelet_filters()                                        |    |        |
| else                                                              |    |        |
| for (i=0; i<wavelet_decomposition_levels; i++)                    |    |        |
| download_wavelet_filters( )                                       |    |        |
| }                                                                 |    |        |
| <b>wavelet_stuffing</b>                                           | 3  | uimsbf |
| if(texture_object_layer_shape == “00”){                           |    |        |
| <b>texture_object_layer_width</b>                                 | 15 | uimsbf |
| <b>marker_bit</b>                                                 | 1  | bslbf  |
| <b>texture_object_layer_height</b>                                | 15 | uimsbf |
| <b>marker_bit</b>                                                 | 1  | bslbf  |
| }                                                                 |    |        |
| else {                                                            |    |        |
| <b>horizontal_ref</b>                                             | 15 | imsbf  |
| <b>marker_bit</b>                                                 | 1  | bslbf  |
| <b>vertical_ref</b>                                               | 15 | imsbf  |
| <b>marker_bit</b>                                                 | 1  | bslbf  |
| <b>object_width</b>                                               | 15 | uimsbf |
| <b>marker_bit</b>                                                 | 1  | bslbf  |
| <b>object_height</b>                                              | 15 | uimsbf |
| <b>marker_bit</b>                                                 | 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{                                                         |  |  |
| <b>quant_byte</b>                                           |  |  |
| } while( quant_byte >>7)                                    |  |  |
| <b>max_bitplanes</b>                                        |  |  |
| 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 {                                                       |             |          |
| <b>quant_byte</b>                                          | 8           | uimsbf   |
| } while (quant_byte >> 7)                                  |             |          |
| for (i=0; i<wavelet_decomposition_levels; i++)             |             |          |
| if ( i!=0    color!= "u","v" ) {                           |             |          |
| <b>max_bitplane[i]</b>                                     | 5           | uimsbf   |
| if ((i+1)%4==0)                                            |             |          |
| <b>marker_bit</b>                                          | 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 {                                                                                                                                                                                                                                                                                         |             |          |
| <b>quant_byte</b>                                                                                                                                                                                                                                                                            | 8           | uimbsf   |
| } while (quant_byte >> 7)                                                                                                                                                                                                                                                                    |             |          |
| for (i=0; i<wavelet_decomposition_levels; i++)                                                                                                                                                                                                                                               |             |          |
| if ( i!=0    color!= "u","v" ) {                                                                                                                                                                                                                                                             |             |          |
| <b>max_bitplane[i]</b>                                                                                                                                                                                                                                                                       | 5           | uimbsf   |
| if ((i+1)%4==0)                                                                                                                                                                                                                                                                              |             |          |
| <b>marker_bit</b>                                                                                                                                                                                                                                                                            | 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")   <br>(second_wavelet_layer && color=="u","v") ) |             |          |
| do {                                                                                      |             |          |
| <b>quant_byte</b>                                                                         | 8           | uimbsf   |
| } while (quant_byte >> 7)                                                                 |             |          |
| if (color == "y")                                                                         |             |          |
| <b>max_bitplanes</b>                                                                      | 5           | uimbsf   |
| else if (!first_wavelet_layer)                                                            |             |          |
| <b>max_bitplanes</b>                                                                      | 5           | uimbsf   |
| }                                                                                         |             |          |
| arith_decode_highbands_bb()                                                               |             |          |
| }                                                                                         |             |          |

"

with

"

| TextureSpatialLayerSQNSC() {                                                                                                                                                                                                                                                                                          | No. of bits | Mnemonic |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|----------|
| for (color="y","u","v") {                                                                                                                                                                                                                                                                                             |             |          |
| if ( (first_wavelet_layer && color=="y")   <br>(second_wavelet_layer && color=="u","v") )                                                                                                                                                                                                                             |             |          |
| do {                                                                                                                                                                                                                                                                                                                  |             |          |
| <b>quant_byte</b>                                                                                                                                                                                                                                                                                                     | 8           | uimsbf   |
| } while (quant_byte >> 7)                                                                                                                                                                                                                                                                                             |             |          |
| if (color == "y")                                                                                                                                                                                                                                                                                                     |             |          |
| <b>max_bitplanes</b>                                                                                                                                                                                                                                                                                                  | 5           | uimbsf   |
| else if (!first_wavelet_layer)                                                                                                                                                                                                                                                                                        |             |          |
| <b>max_bitplanes</b>                                                                                                                                                                                                                                                                                                  | 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".<br>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<br>&& spatial_layer_id == 0) { |             |          |
| for (color = "y" ) {                                                                          |             |          |
| do {                                                                                          |             |          |
| <b>quant_byte</b>                                                                             | 8           | uimsbf   |
| } while (quant_byte >> 7)                                                                     |             |          |
| for (i=0; i<spatial_layers; i++) {                                                            |             |          |
| <b>max_bitplane[i]</b>                                                                        | 5           | uimsbf   |
| if ((i+1)%4 == 0)                                                                             |             |          |
| <b>marker_bit</b>                                                                             | 1           | bslbf    |
| }                                                                                             |             |          |
| }                                                                                             |             |          |
| else {                                                                                        |             |          |
| for (color="y", "u", "v") {                                                                   |             |          |
| do {                                                                                          |             |          |
| <b>quant_byte</b>                                                                             | 8           | uimsbf   |
| } while (quant_byte >> 7)                                                                     |             |          |
| for (i=0; i<spatial_layers; i++) {                                                            |             |          |
| <b>max_bitplane[i]</b>                                                                        | 5           | uimsbf   |
| if ((i+1)%4 == 0)                                                                             |             |          |
| <b>marker_bit</b>                                                                             | 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<br>&& spatial_layer_id == 0) { |             |          |
| for (color = "y" ) {                                                                          |             |          |
| do {                                                                                          |             |          |
| <b>quant_byte</b>                                                                             | 8           | uimsbf   |
| } while (quant_byte >> 7)                                                                     |             |          |
| for (i=0; i<spatial_layers; i++) {                                                            |             |          |
| <b>max_bitplane[i]</b>                                                                        | 5           | uimsbf   |
| if ((i+1)%4 == 0)                                                                             |             |          |
| <b>marker_bit</b>                                                                             | 1           | bslbf    |
| }                                                                                             |             |          |
| }                                                                                             |             |          |
| }                                                                                             |             |          |
| else {                                                                                        |             |          |
| for (color="y", "u", "v") {                                                                   |             |          |
| do {                                                                                          |             |          |
| <b>quant_byte</b>                                                                             | 8           | uimsbf   |
| } while (quant_byte >> 7)                                                                     |             |          |
| for (i=0; i<spatial_layers; i++) {                                                            |             |          |
| <b>max_bitplane[i]</b>                                                                        | 5           | uimsbf   |
| if ((i+1)%4 == 0)                                                                             |             |          |
| <b>marker_bit</b>                                                                             | 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( ){       | <b>No. of bits</b> | <b>Mnemonic</b> |
| <b>lowpass_filter_length</b>       | 4                  | uimsbf          |
| <b>highpass_filter_length</b>      | 4                  | uimsbf          |
| do{                                |                    |                 |
| if ( wavelet_filter_type == 0) {   |                    |                 |
| <b>filter_tap_integer</b>          | 16                 | imsbf           |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| } else {                           |                    |                 |
| <b>filter_tap_float_high</b>       | 16                 | uimsbf          |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| <b>filter_tap_float_low</b>        | 16                 | uimsbf          |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| }                                  |                    |                 |
| } while (lowpass_filter_length--)  |                    |                 |
| do{                                |                    |                 |
| if ( wavelet_filter_type == 0){    |                    |                 |
| <b>filter_tap_integer</b>          | 16                 | imsbf           |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| } else {                           |                    |                 |
| <b>filter_tap_float_high</b>       | 16                 | uimsbf          |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| <b>filter_tap_float_low</b>        | 16                 | uimsbf          |
| <b>marker_bit</b>                  | 1                  | bslbf           |
| }                                  |                    |                 |
| } while (highpass_filter_length--) |                    |                 |
| if ( wavelet_filter_type == 0) {   |                    |                 |
| <b>integer_scale</b>               | 16                 | uimsbf          |
| <b>marker_bit</b>                  |                    |                 |
| }                                  |                    |                 |
| }                                  |                    |                 |

"

with

"

|                               |                    |                 |
|-------------------------------|--------------------|-----------------|
| download_wavelet_filters( ){  | <b>No. of bits</b> | <b>Mnemonic</b> |
| <b>lowpass_filter_length</b>  | 4                  | uimsbf          |
| <b>highpass_filter_length</b> | 4                  | uimsbf          |