



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

Published 2004-06-15

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

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

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

RECTIFICATIF TECHNIQUE 1

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

In 3.43 clarify coded order, replace:

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

by:

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

In 3.60 decoding order, replace:

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

by:

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

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

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

by:

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

In clause 3, insert a subclause, time base:

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

In subclause 6.1.1, replace:

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

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

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

by:

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

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

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

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

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

by:

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

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

1) Replace the following text in subclause 6.2.1:

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

by:

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

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

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

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

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

by:

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

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

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

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

by:

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

if (interlaced && field_prediction)		
motion_vector("backward")		
}		
if (mb_type == "1")		
motion_vector("direct")		
}		
if (ref_select_code == '00' && scalability) {		
if (cbpb !=0)		
dbquant	1-2	vlcblbf
if (mb_type == '01' mb_type == '1')		
motion_vector("forward")		
}		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
block(i)		
}		
}		
}		
}		
}		

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

fgs_motion_interlaced_information() {	No. of bits	Mnemonic
fgs_field_prediction	1	bslbf
if (fgs_field_prediction) {		
if (fgs_vop_coding_type == "P" (fgs_vop_coding_type=="B" && fgs_b_mb_type!="001")){		
fgs_forward_top_field_reference	1	bslbf
fgs_forward_bottom_field_reference	1	bslbf
}		
if ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1")) {		
fgs_backward_top_field_reference	1	bslbf
fgs_backward_bottom_field_reference	1	bslbf
}		
}		
}		
}		

by:

fgs_motion_interlaced_information() {	No. of bits	Mnemonic
if (((fgs_vop_coding_type == "P") && (fgs_p_mb_type == 0)) ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1"))) {		
fgs_field_prediction	1	bslbf
if (fgs_field_prediction) {		
if (fgs_vop_coding_type == "P" (fgs_vop_coding_type == "B" && fgs_b_mb_type != "001")){		
fgs_forward_top_field_reference	1	bslbf
fgs_forward_bottom_field_reference	1	bslbf
}		
if ((fgs_vop_coding_type == "B") && (fgs_b_mb_type != "1")) {		
fgs_backward_top_field_reference	1	bslbf
fgs_backward_bottom_field_reference	1	bslbf
}		
}		
}		
}		

Following the previous item, in subclause 6.3.14.3 replace:

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

by:

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

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

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

by:

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

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

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

by:

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

In subclause 6.3.3, clarify semantics of fixed_vop_rate. Replace:

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

by:

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

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

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

EXAMPLE

VOP time = tick × vop_time_increment
= vop_time_increment / vop_time_increment_resolution

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

Fixed VOP rate = 1/VOP time	<code>vop_time_increment_resolution</code>	<code>fixed_vop_time_increment</code>	<code>vop_time_increment</code>
15Hz	15	1	0, 1, 2, 3, 4,...
7.5Hz	15	2	0, 2, 4, 6, 8,...
29.97...Hz	30000	1001	0, 1001, 2002, 3003,...
59.94...Hz	60000	1001	0, 1001, 2002, 3003,...

by:

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

EXAMPLE

Fixed VOP Period = tick $\times$ `vop_time_increment`
= `vop_time_increment` $\div$ `vop_time_increment_resolution`

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

Fixed VOP rate = 1÷Fixed VOP Period	<code>vop_time_increment_resolution</code>	<code>fixed_vop_time_increment</code>	<code>vop_time_increment</code>
15Hz	15	1	0, 1, 2, 3, 4,...
7.5Hz	15	2	0, 2, 4, 6, 8,...
29.97...Hz	30000	1001	0, 1001, 2002, 3003,...
59.94...Hz	60000	1001	0, 1001, 2002, 3003,...

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

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

by:

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

Following the previous item, replace:

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

by:

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