INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 N3062

December, 1999

Subject: MPEG-2 Video Elementary Stream Supplemental Information

From: Video

Status: Approved

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND AUDIO: VIDEO 2nd Edition

Use of the extra_information_picture field for Content Description Data

---

ITU-T Rec. H.262 | ISO/IEC 13818-2.2/PDAM1

International Standard

INTERNATIONAL STANDARD

INFORMATION TECHNOLOGY -- GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: VIDEO 2nd Edition

Proposed Draft Amendment

Proposed Syntax Modifications

1) Replace the 6.3.9 semantics for extra_bit_picture and extra_information_picture with the following:

“

extra_bit_picture – This flag indicates the presence of the following extra information. If extra_bit_picture is set to ‘1’, extra_information_picture will follow it. If it is set to ‘0’, no further extra_information_picture bytes follow in this picture header.

extra_information_picture – A byte that contains data that describes the picture. The number of bytes is determined by the setting of the extra_bit_picture flag. The set of bytes may contain one or more consecutive instances of extra_information_picture_payload that is further described in 6.2.3.1 and 6.3.10.

”

2) Add the following after subclause 6.2.3 and subsequently increment sections 6.2.3.1 through 6.2.3.8:

“

6.2.3.1 extra_information_picture_payload

extra_information_picture_payload() {	**No. of bits**	**Mnemonic**
content_data_type	16	uimsbf
content_data_length	8	uimsbf
for (i=0; i<content_data_length; i++) {		
content_data	8	uimsbf
}		
}		

NOTE – This table represents the data that is to be sent. Between each byte are extra_bit_picture bits as specified in the picture_header definition. The formal syntax description shown above only defines the payload and thus does not explicitly show the extra_bit_picture bits.

”

3) Add the following after subclause 6.3.9 and increment the following sections:

“

6.3.10 extra_information_picture_payload

For the data defined for each extra_information_picture_payload, all data shall be packed into contiguous sets of 8 bits and encoded with each byte separated by one extra_bit_picture bit set to ‘1’ as described in subclause 6.2.3.

The information in extra_information_picture_payload does not affect the decoding process and may be ignored by decoders that conform to this Specification.

content_data_type – A 16 bit unsigned integer value that defines the type of the data as shown in Table 6-17.

Table 6-17. data_type codes

Value	Meaning
0000 0000 0000 0000	reserved
0000 0000 0000 0001	padding_bytes
0000 0000 0000 0010	capture_timecode
0000 0000 0000 0011	additional_pan-scan_parameters
0000 0000 0000 0100	active_region_window
0000 0000 0000 0101	coded_picture_length
0000 0000 0000 0110	reserved
…	reserved
…	reserved
1111 1111 1111 1111	reserved

content_data_length – A 8 bit unsigned integer value that defines the number of bytes of content_data. The use of a data_length_value field allows decoders to ignore bytes for data_types that it does not recognize or want to parse.

content_data -- This is an 8 bit data field whose meaning is defined depending on the content_data_type. A number of content_data fields is defined by the content_data_length.

6.3.10.1 padding_bytes

Use of this data_type value is intended to allow for padding bytes, equal to ‘0000 0000’, to be included in VBV calculations and defined later in a multi-pass system.

6.3.10.2 capture_timecode

The capture_timecode describes the source capture or creation time of the fields or frames of the content. It contains absolute timestamps for the associated frame or fields. Only one capture_timecode for each picture shall be present in the bitstream. This timecode does not take precedence over the PTS.

An equivalent timestamp represented in 27 MHz clock cycles is calculated from the following formula:

equivalent_timestamp = (60 * (60 * (units_of_hours + 10 * tens_of_hours) + (units_of_minutes + 10 * tens_of_minutes)) + units_of_seconds + 10 * tens_of_seconds) * 27,000,000 + time_offset

Two identical equivalent_timestamps calculated from consecutive frames or fields without an intervening time_discontinuity indicate both frames or fields were captured or created at the same instant in time.

capture_timecode

capture_timecode() {	**No. of bits**	**Mnemonic**
num_timecodes	2	uimsbf
frame_field_capture_timestamp()		
if (num_timecodes == ‘11’) {		
frame_field_capture_timestamp()		
}		
padding_bits	6	uimsbf
}		

frame_or_field_capture_timestamp

frame_or_field_capture_timestamp() {	**No. of bits**	**Mnemonic**
time_discontinuity	1	uimsbf
reserved_bit	1	bslbf
time_offset	26	simsbf
units_of_seconds	4	uimsbf
tens_of_seconds	3	uimsbf
units_of_minutes	4	uimsbf
tens_of_minutes	3	uimsbf
units_of_hours	4	uimsbf
tens_of_hours	2	uimsbf
}		

num_timecodes – A 2 bit field that indicates the number of timecodes associated with this picture.

Table 6-20. num_timecodes values

Value	Meaning
00	one timecode for the frame
01	one timecode for the first or only field
10	one timecode for the second field
11	two timecodes for the two fields

time_discontinuity -- A 1 bit flag that indicates if a discontinuity in time or timebase between the previous timecode and the current timecode has occurred. If set to '0', there has been no time discontinuity. If editing occurs that results in time or timebase discontinuities or if the previous field or frame timecode is unavailable, the time_discontinuity bit shall be set to '1'.

time_offset – A signed 26-bit field that is the number of 27 MHz system clock cycles offset from the units_of_seconds time indicating the equivalent timestamp for when the current field or frame was captured. Values are constrained to less than 27,000,000 in magnitude.

units_of_seconds – A 4 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in seconds modulus 10.

Table 6-21. units_of_seconds values

Value	Meaning
0000 – 1001	number of seconds modulo 10
1010 – 1111	forbidden

tens_of seconds – A 3 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in seconds divided by 10.

Table 6-22. tens_of_seconds values

Value	Meaning
000 – 101	number of seconds / 10
110 – 111	forbidden

units_of_minutes – A 4 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in minutes modulus 10.

Table 6-23. units_of_minutes values

Value	Meaning
0000 – 1001	number of minutes modulo 10
1010 – 1111	forbidden

tens_of_minutes – A 3 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in seconds divided by 10.

Table 6-24. tens_of_minutes values

Value	Meaning
000 – 101	number of minutes / 10
110 – 111	forbidden

units_of_hours – A 4 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in hours modulus 10.

Table 6-25. units_of_hours values

Value	Meaning
0000 – 1001	number of hours modulo 10
1010 – 1111	forbidden

tens_of_hours – A 2 bit field that is used to calculate the equivalent timestamp. This field represents the portion of the timestamp of this field or frame measured in hours divided by 10.

Table 6-26. tens_of_hours values

Value	Meaning
00 – 10	number of hours / 10
11	forbidden

6.3.10.2 additional_pan-scan_parameters

These fields define additional pan-scan parameters to support more than one display type. For example, if the pan-scan information encoded in the sequence_header, sequence_display_extension, and picture_display_extension is used to define the parameters needed for display on a 3÷4 display aspect ratio display, the additional_pan-scan_parameters can define the parameters needed for display on a 9÷16 aspect ratio display.

additional_pan-scan_parameters

additional_pan-scan_parameters() {	**No. of bits**	**Mnemonic**
aspect_ratio_information	4	uimsbf
display_size_present	1	bslbf
if (display_size_present) {		
display_horizontal_size	14	uimsbf
display_vertical_size	14	uimsbf
padding_bits	4	uimsbf
}		
for (i=0; i<number_of_frame_centre_offsets; i++) {		
frame_centre_horizontal_offset	16	simsbf
frame_centre_vertical_offset	16	simsbf
}		
padding_bits	3	uimsbf
}		

aspect_ratio_information – A 4-bit integer value that is defined in subclause 6.3.3 (sequence header).

display_size_present – A 1-bit flag, when set to '1', indicates the presence of the display_horizontal_size and display_vertical_size parameters. When set to '0', the previous values of display_ horizontal_size and display_vertical_size are assumed. For a specific aspect ratio, it is recommended that this field be set to '1'in the first picture header after any sequence_header().

display_horizontal_size – A 14-bit integer value that is defined in subclause 6.3.6 (sequence display extension). For a specific aspect ratio, the value of this parameter shall remain the same for the sequence.

display_vertical_size – A 14-bit integer value that is defined in subclause 6.3.6 (sequence display extension). For a specific aspect ratio, the value of this parameter shall remain the same for the sequence.

frame_centre_horizontal_offset – A 16-bit signed integer that is defined in subclause 6.3.12 (picture display extension). The value of the number_of_frame_centre_offsets can also be found in 6.3.12.

frame_centre_vertical_offset – A 16-bit signed integer that is defined in subclause 6.3.12 (picture display extension).

6.3.10.3 active_region_window

The active_region_window are integers that define the rectangle in the reconstructed frame that is intended to be displayed. This window shall not be larger than the rectangle defined by the horizontal_size and vertical_size defined in subclause 6.3.3. Only one active_region_window for each picture shall be present in the bitstream.

active_region_window

	No. of bits	**Mnemonic**
active_region_window() {		
top_left_x	16	uimsbf
top_left_y	16	uimsbf
active_region_horizontal_size	16	uimsbf
active_region_vertical_size	16	uimsbf
}		

top_left_x – A 16-bit integer that defines the sample number in the reconstructed frame that is the upper left corner of the active_region_window's rectangle.

top_left_y – A 16-bit integer that defines the line number in the reconstructed frame that contains the sample in the upper left corner of the active_region_window's rectangle.

active_region_horizontal_size – A 16-bit integer that together with active_region_vertical_size defines a rectangle that may be considered the active region. If this rectangle is smaller than the encoded frame size, then the display process may be expected to display only that portion of the encoded frame. This value cannot, by definition, be larger than the horizontal_size of the encoded frame. The value of '0' indicates that the size is unknown.

active_region_vertical_size – See the definition for active_region_horizontal_size. This value cannot, by definition, be larger than the vertical_size of the encoded frame. The value of '0' indicates that the size is unknown.

In the case that a given picture does not have the active_region_window(), then the most recently decoded active_region_window shall be used. Following a sequence_header, the active_region_window is assumed to be the same size as the entire encoded picture size.

6.3.10.4coded_picture_length

The coded_picture_length is the number of bytes included from the first byte immediately following the first slice_start_code of a picture to the first byte of the start code prefix immediately following the last macroblock of the frame. Only one coded_picture_length for each picture shall be present in the bitstream.

coded_picture_length

	No. of bits	**Mnemonic**
coded_picture_length() {		
picture_byte_count	32	uimsbf
}		

picture_byte_count – A 32 bit integer that defines the number of bytes from the first byte immediately following the first slice_start_code of a picture to the first byte of the start code prefix immediately following the last macroblock of the picture. A value of '0' indicates that the length is unknown.

"

4) Add the following after the last Annex:

“

Annex K

The Impact of Practices for Non-Progressive Sequence Bitstreams in Consideration of Progressive-Scan Display

(This annex does not form an integral part of this Recommendation | International Standard)

K.1 Progressive and Non-Progressive Encoding

This annex discusses the effect of encoding practices on the use of non-progressive ITU-T Rec. H.262 | ISO/IEC 13818-2 video sequences on systems with progressive-scan displays. It is intended primarily to encourage content producers to encode material in a manner that is free of unnecessary artifacts when played on systems with progressive-scan displays. While the display process is beyond the scope of this Recommendation | International Standard, a number of syntax elements are included in the bitstream that can help the display process, such as the sequence display extension and the picture display extension. This annex discusses the optimization of syntax usage in view of its impact on the display process.

The normative semantics of the progressive_frame flags describe the source temporal relationship between the fields within a coded picture of a non-progressive sequence, and decoders that display content on progressive-scan devices normally rely on this flag to pair fields for presentation.

The general display practice is as follows: if a picture is encoded as a progressive frame, the two fields are interleaved for presentation on the progressive-scan device; otherwise, some interlace-to-progressive conversion process is performed to convert the output field data to frame data for display. If the picture was actually generated with a progressive scan, but is encoded with an incorrect non-progressive source timing indication, the interlace-to-progressive conversion process will be erroneously applied and may result in serious artifacts and loss of vertical resolution on the display.

K.2 Video Source Timing Information Syntax

The represented video source sampling timing for pictures in non-progressive sequences (when progressive_sequence is ‘0’) depends on the progressive_frame flag in the picture coding extension defined in subclause 6.3.10. (See also Figures 6-2, 6-3, and 6-4.) It is important to note that in frame pictures of such sequences (when picture_structure is ‘11’), progressive_frame can be either ‘0’ or ‘1’ with no effect on the decoding process and thus serves only to indicate the source sampling timing.

The represented source sampling timing in a non-progressive sequence includes a field-time offset between the time of the fields of the picture whenever the one-bit progressive_frame flag is ‘0’. This includes the following cases:

- when picture_structure is ‘01’ (top field) or ‘10’ (bottom field) – in which case progressive_frame is required to be ‘0’, or
- when picture_structure is ‘11’ (frame picture) and progressive_frame is ‘0’ (non-progressive).

The represented source sampling timing is that of a frame picture sampled at a single time instant in the remaining case:

- when picture_structure is ‘11’ (frame picture) and progressive_frame is ‘1’ (progressive).

  In this last case the picture is indicated as progressive, as would be the case if progressive_sequence was ‘1’.

  The display process for progressive-scan display of progressive frames normally simply uses all

of the lines of the decoded picture with interleaving of the two fields. The display process for progressive-scan display of non-progressive frames usually differs substantially from this simple interleaving of fields.

K.3 Content Generation Practices

If the original source material that is to be encoded was sampled as full frames of progressive scan content, it is important that the progressive nature of the source material is properly represented in the video bitstream. Progressive content should therefore be encoded using a properly-paired progressive representation. If this practice is not followed, significant unnecessary artifacts may appear on systems using progressive-scan displays. It is similarly important to ensure that truly interlaced material be encoded with progressive_frame = '0' to avoid improper display processing on systems using progressive-scan displays.

If an entire source sequence consists of progressive frames, then if possible the sequence should simply be encoded as progressive frames with progressive_sequence set to '1'. In non-progressive sequences (when progressive_sequence is '0'), the progressive nature of individual frames can still be represented by encoding progressive pictures as frame pictures with progressive_frame equal to '1'.

> Experience has shown that content producers have sometimes neglected to properly signal the progressive nature of progressive frames encoded within non-progressive sequences. Certain video editing practices can also cause a progressive source to lose its progressive nature to some degree and thus to lose its ability to be encoded as properly progressive frames. The primary purpose of this annex is to help content producers to avoid creating video bitstreams that produce unnecessary artifacts as a result of these problems.

K.3.1 Frame-Rate Conversion Pre-Processing

Source material generated at some particular frame rate is commonly converted for encoding as a video bitstream at a different frame rate. If the source frame rate is moderately lower than the encoded frame rate, this is often done in non-progressive sequences by adding repeated single fields of encoded content using progressive_frame = '1' with repeat_first_field = '1'.

Currently the most common such practice is the conversion of 24 frame per second material to approximately 30 frames per second by a process known as 3:2 pull-down (also known as 2:3 pull-down, which is an arguably more technically correct term). In this process, a set of four progressive scanned pictures, denoted as pictures A, B, C, and D, are converted to ten fields of video content by repeating the first field of pictures B and D. The same pattern is then repeated again and again for each subsequent set of four pictures.

In the case of 3:2 pull-down use, the most important preferred practice consideration is that each source picture (A, B, C, and D) should be represented in the bitstream as a distinct encoded picture. In other words, that source pictures B and D in the pattern be encoded as distinct pictures with progressive_frame = '1' and repeat_first_field = '1'. One example of poor use of syntax would be to encode source pictures A and B as the first two encoded pictures (each with repeat_first_field = '0'), then encode the repeated first field of source picture B and the first of the two fields of source picture C together as the third encoded picture, then encode the second field of source picture C and the first field of source picture D together as the fourth encoded picture, and then encode the two fields of source picture D as the fifth encoded picture; and repeating this pattern for each set of four source pictures. This poor use of syntax would be likely to generate significant artifacts on a system with a progressive scan display, as the display process would most likely be unable to recover the correct field pairing and timing information needed to properly reconstruct the progressive format pictures.

In addition, it is very important that even the source pictures that do not have a repeated first field contain a proper representation of their progressive nature. This requires that source pictures A and C be encoded

as frame pictures with progressive_frame = '1' (as well as with repeat_first_field = '0'). In cases in which the encoder must attempt to infer the presence of 3:2 pull-down within a series of video source fields, the presence of repeated fields may be useful to determine that the source material is progressive and to determine the proper association of fields to frames (although the output quality will be significantly better if the actual source nature can be known to the encoder, rather than needing to be detected using such an imperfect detection process).

Following these preferred encoding practices requires that the progressive nature of source frames and the proper association of fields to frames be retained intact through the encoding process.

K.3.2 Detrimental Field-Oriented Editing Practices

Certain video editing practices which operate without awareness of the correct pairing of fields to form progressive frames can be detrimental to the progressive nature of source material and therefore should be avoided to the maximum possible extent, due to the artifacts they may create on systems which use progressive scan displays. In order to avoid the difficulties that may arise due to these practices, the complete chain of production, editing, and encoding processes should be designed in a manner to ensure that correct information regarding the progressive or interlaced nature of each part of the video content is preserved. In 3:2 pull-down processing, the editing practices should operate with the same pattern of field-paired processing as that of the underlying source material in order to avoid such difficulties.

K.3.2.1 Field-Oriented Scene Cuts

One example of such a harmful editing practice would be to switch between two progressive video sources in field-based processing between the two fields of a progressive frame, such as the occurrence of a scene cut just after the first field of source picture C in a 3:2 pull-down series as described in sub-clause Z.3.1. The result of a scene cut at such a location is to create a "stranded field" of source video content – an isolated field of source video which cannot be properly paired with another field to create an encoded progressive frame of source video content. If such a condition exists and must be encoded in a video bitstream, it is important to set progressive_frame to '0' on the particular picture containing the stranded field in order to properly signal its non-progressive nature.

K.3.2.2 Field-Structured Overlays and Compositing

Another case of field-oriented editing which may have an effect on whether or not a picture can be properly characterized as progressive is the insertion of moving text overlays and other such content. If such material is inserted in an otherwise-progressive scene using an interlace-oriented editing process, the source material can no longer be properly characterized as either truly progressive or truly interlaced. As a result, it becomes unclear whether a picture should be marked with progressive_frame = '0' or '1', and systems using a progressive scan display may have significant difficulty determining how to present the decoded pictures for display. To the extent possible, any such overlays and compositing of pictures should be performed using progressive scan representations to avoid such difficulties.

K.3.2.3 Field-Oriented Fades, Scene Transitions

If gradual transitions between progressive scan scene content is performed, such as a "fade in," a "fade out", a gradual scene switch or a "wipe" transition, such transitions should be performed using frame-oriented progressive scan processing to the maximum possible extent – operating on the properly-paired true progressive frames. Unless this practice is followed, the source material can no longer be properly characterized as either truly progressive or truly interlaced. As a result, it becomes unclear whether a picture should be marked with progressive_frame = '0' or '1', and systems using a progressive scan display may have significant difficulty determining how to present the decoded pictures for display.

K.3.2.4 Field-Oriented Special Effects

Special effects such as a zoom that may have been applied after the initial capture of the video content should similarly be performed with great care as to the progressive nature of source material. These effects should be applied in operation on properly paired progressive pictures rather than in an arbitrary fashion so as to preserve the progressive nature of the material.

K.3.2.5 Field-Oriented Speed Adjustments

If the number of field times associated with progressive scan source pictures is altered in order to adjust the speed of motion in a scene, care must be taken to ensure that the speed adjustments do not cause subsequent processing (e.g. 3:2 pull-down detection in an encoder as described in sub-clause Z.3.1) to lose track of the progressive nature of the pictures and the proper association of fields to pictures.

K.3.2.6 Frame Centering

If the frame centre to be indicated in a picture_display_extension() indicates differing values of frame_centre_vertical_offset or frame_centre_horizontal_offset for different fields in the same picture, this creates another form of indicated temporal change between fields of what might otherwise be progressive scan pictures. The underlying frame rate of the progressive scan source pictures would then be effectively altered by this process, potentially creating a displayed sequence that has field-oriented changes for each such progressive picture.

K.4 Post-Encoding Editing of the Progressive Frame Flag in Video Bitstreams

It is important to note that in frame pictures of non-progressive sequences (when progressive_sequence is '0'), the value of progressive_frame has no effect on the decoding process and thus serves only to indicate the source sampling timing.

Because of the importance of the value of progressive_frame to systems using progressive scan displays, it may be advisable in some cases to consider altering the value of this bit in bitstreams which have been created with improper settings of this flag. The restrictions on when such alteration can be performed for this purpose without harm to the decoding process are that:

- progressive_sequence must be '0' (a non-progressive sequence),
- picture_structure must be '11' (a frame picture),
- progressive_frame cannot be changed from '1' to '0' if repeat_first_field is '1', and
- progressive_frame cannot be changed unless chroma_420_format is also changed to have the same value as progressive_frame.

NOTE – These statements are noted for informational purposes only – as is the case with all statements in this Annex, these statements do not alter in any way the specifications found in the integral parts of this Recommendation | International Standard.

K.5 Post-Processing for Systems with Progressive Scan Displays

If the decoding system detects the presence of an isolated non-progressive frame within a series of progressive frames, this may indicate the presence of a stranded field as described in sub-clause Z.3.2.1. Display processing designers should consider providing some special handling for this situation.

If the decoding system detects the presence of a repetitive pattern of progressive frames with repeat_first_field = '1' mixed with non-progressive frames, this may indicate the behavior of an encoder which is unaware of the progressive nature of the source content frames and is only able to detect that a frame is progressive if its first field is repeated in the source video sequence. Display processing designers should consider treating even the frames with repeat_first_field = '0' and progressive_frame = '0' as being actually progressive scan frames if a persistent pattern repetitive pattern of this phenomenon is encountered in the bitstream.

"