INTERNATIONAL STANDARD

ISO/IEC 13818-2

Second edition
2000-12-15

AMENDMENT 1
2001-12-15

Corrected version
2002-08-01

Information technology — Generic coding of moving pictures and associated audio information: Video

AMENDMENT 1: Content description data

Technologies de l'information — Codage générique des images animées et du son associé: Données vidéo

AMENDEMENT 1: Données de description du contenu

Reference number
ISO/IEC 13818-2:2000/Amd.1:2001(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this Amendment may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to International Standard ISO/IEC 13818-2:2000 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. H.262/Amd.1.

This corrected version of ISO/IEC 13818-2:2000/Amd.1:2001 incorporates the following corrections:

— title of the amendment (cover page and page 1);

— edition number of ISO/IEC 13818-2:2000 (cover page).

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION: VIDEO

AMENDMENT 1

Content description data

1) Subclause 6.2.3

Replace subclause 6.2.3 by:

6.2.3 Picture header

picture_header() {	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if (picture_coding_type == 2 || picture_coding_type == 3) {		
full_pel_forward_vector	1	bslbf
forward_f_code	3	bslbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	bslbf
backward_f_code	3	bslbf
}		
while (nextbits() == '1') {		
extra_bit_picture /* with the value '1' */	1	uimsbf
content_description_data() /* with every 9th bit having the value '1' */		
}		
extra_bit_picture /* with the value '0' */	1	uimsbf
next_start_code()		
}		

2) New subclause 6.2.3.7.3

Insert new subclause 6.2.3.7.3:

6.2.3.7.3 Content description data

content_description_data() {	No. of bits	Mnemonic
data_type_upper	8	uimsbf
marker_bit	1	bslbf
data_type_lower	8	
marker_bit	1	bslbf
data_length	8	uimsbf
if (data_type == "Padding Bytes")		
padding_bytes()		
else if (data_type == "Capture Timecode")		
capture_timecode()		
else if (data_type == "Additional Pan-Scan Parameters")		
additional_pan_scan_parameters()		
else if (data_type == "Active Region Window")		
active_region_window()		
else if (data_type == "Coded Picture Length")		
coded_picture_length()		
else		
for (i = 0; i < data_length; i ++) {		
marker_bit	1	bslbf
reserved_content_description_data	8	uimsbf
}		
}		

3) New subclause 6.2.3.7.3.1

Insert new subclause 6.2.3.7.3.1:

6.2.3.7.3.1 Padding bytes

padding_bytes() {	No. of bits	Mnemonic
for (i = 0; i < data_length; i ++) {		
marker_bit	1	bslbf
padding_byte	8	bslbf
}		
}		

4) New subclause 6.2.3.7.3.2

Insert new subclause 6.2.3.7.3.2:

6.2.3.7.3.2 Capture timecode

capture_timecode() {	No. of bits	Mnemonic
marker_bit	1	bslbf
timecode_type	2	uimsbf
counting_type	3	uimsbf
reserved_bit	1	uimsbf
reserved_bit	1	uimsbf
reserved_bit	1	uimsbf
if (counting_type != 0) {		
marker_bit	1	bslbf
nframes_conversion_code	1	uimsbf
clock_divisor	7	uimsbf
marker_bit	1	bslbf
nframes_multiplier_upper	8	uimsbf
marker_bit	1	bslbf
nframes_multiplier_lower	8	
}		
frame_or_field_capture_timestamp()		
if (timecode_type == '11')		
frame_or_field_capture_timestamp()		
}		

5) New subclause 6.2.3.7.3.2.1

Insert new subclause 6.2.3.7.3.2.1:

6.2.3.7.3.2.1 Frame or field capture timestamp

frame_or_field_capture_timestamp() {	No. of bits	Mnemonic
if (counting_type != 0) {		
marker_bit	1	bslbf
nframes	8	uimsbf
}		
marker_bit	1	bslbf
time_discontinuity	1	uimsbf
prior_count_dropped	1	uimsbf
time_offset_part_a	6	simsbf
marker_bit	1	bslbf
time_offset_part_b	8	
marker_bit	1	bslbf
time_offset_part_c	8	
marker_bit	1	bslbf
time_offset_part_d	8	
marker_bit	1	bslbf
units_of_seconds	4	uimsbf
tens_of_seconds	4	uimsbf
marker_bit	1	bslbf
units_of_minutes	4	uimsbf
tens_of_minutes	4	uimsbf
marker_bit	1	bslbf
units_of_hours	4	uimsbf
tens_of_hours	4	uimsbf
}		

6) New subclause 6.2.3.7.3.3

Insert new subclause 6.2.3.7.3.3:

6.2.3.7.3.3 Additional pan-scan parameters

additional_pan_scan_parameters() {	No. of bits	Mnemonic
marker_bit	1	bslbf
aspect_ratio_information	4	uimsbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_size_present	1	bslbf
if (display_size_present == '1') {		
marker_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_horizontal_size_upper	6	uimsbf
marker_bit	1	bslbf
display_horizontal_size_lower	8	
marker_bit	1	bslbf
reserved_bit	1	bslbf
reserved_bit	1	bslbf
display_vertical_size_upper	6	uimsbf
marker_bit	1	bslbf
display_vertical_size_lower	8	
}		
for (i = 0; i < number_of_frame_centre_offsets; i ++) {		
marker_bit	1	bslbf
frame_centre_horizontal_offset_upper	8	simsbf
marker_bit	1	bslbf
frame_centre_horizontal_offset_lower	8	
marker_bit	1	bslbf
frame_centre_vertical_offset_upper	8	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset_lower	8	
}		
}		

7) New subclause 6.2.3.7.3.4

Insert new subclause 6.2.3.7.3.4:

6.2.3.7.3.4 Active region window

active_region_window() {	No. of bits	Mnemonic
marker_bit	1	bslbf
top_left_x_upper	8	uimsbf
marker_bit	1	bslbf
top_left_x_lower	8	
marker_bit	1	bslbf
top_left_y_upper	8	uimsbf
marker_bit	1	bslbf
top_left_y_lower	8	
marker_bit	1	bslbf
active_horizontal_size_upper	8	uimsbf
marker_bit	1	bslbf
active_horizontal_size_lower	8	
marker_bit	1	bslbf
active_vertical_size_upper	8	uimsbf
marker_bit	1	bslbf
active_vertical_size_lower	8	
}		

8) New subclause 6.2.3.7.3.5

Insert new subclause 6.2.3.7.3.5:

6.2.3.7.3.5 Coded picture length

coded_picture_length() {	No. of bits	Mnemonic
marker_bit	1	bslbf
picture_byte_count_part_a	8	uimsbf
marker_bit	1	bslbf
picture_byte_count_part_b	8	
marker_bit	1	bslbf
picture_byte_count_part_c	8	
marker_bit	1	bslbf
picture_byte_count_part_d	8	
}		

9) Subclause 6.3.9

Replace the semantics for extra_bit_picture and extra_information_picture with the following (removing the semantics for extra_information_picture):

extra_bit_picture – This flag indicates the presence of the following extra information. If extra_bit_picture is set to '1', content_description_data() shall follow it. If it is set to '0', no further content_description_data() shall follow in this picture header.

10) New subclause 6.3.21

Insert new subclause 6.3.21:

6.3.21 Content description data

data_type_upper, data_type_lower – Two 8-bit unsigned integer values containing the most significant and least significant bits, respectively, of the value of the 16-bit unsigned integer **data_type** that defines the type of content description data. The semantics of **data_type** are defined in Table 6-21.

Table 6-21 – data_type values

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
1111 1111 1111 1111	Reserved

data_length – An 8-bit unsigned integer specifying the remaining amount of data to follow within the remainder of the content description data structure, expressed in units of 9 bits. The number of bits of data which follows within the remainder of the content description data structure shall be equal to data_length * 9.

reserved_content_description_data – Reserved 8-bit unsigned integer. A decoder that encounters reserved_content_description_data in a bitstream shall ignore it (i.e. remove from the bitstream and discard). A bitstream conforming to this Specification shall not contain this syntax element.

In the case that a decoder encounters a data_type unsigned integer that is described as "reserved" in Table 6-21, the decoder shall discard the subsequent pairings of marker_bit and reserved_content_description_data which follow data_length in the bitstream. The number of such pairings shall be equal to data_length. This requirement allows future definition of compatible extensions to this Specification.

reserved_bit – Reserved 1-bit unsigned integer. Shall be equal to '0' in bitstreams conforming to this Specification. The value '1' is reserved for future backward-compatible use by ITU-T | ISO/IEC. A decoder conforming to this Specification shall allow either a value of '0' or '1' for reserved_bit.

11) New subclause 6.3.21.1

Insert new subclause 6.3.21.1:

6.3.21.1 Padding bytes

padding_byte – An 8-bit string which shall be equal to '0000 0000'. All other values are forbidden.

NOTE – The purpose of padding bytes is to allow inclusion of a number of bytes of data which are included in VBV calculations.

12) New subclause 6.3.21.2

Insert new subclause 6.3.21.2:

6.3.21.2 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 shall not take precedence over any timecode specified for presentation or decoding at a systems multiplex level, for example the presentation time stamps or decoding time stamps defined in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 (Systems).

timecode_type – A 2-bit integer that indicates the number of timestamps associated with this picture as defined in Table 6-22. The values '00', '10', and '11' shall only be used when picture_structure is equal to 'Frame Picture'. The value '00' indicates that the two fields that make up the frame have the same capture time. When timecode_type is equal to '11', the first timestamp pertains to the first field of the frame and the second timestamp pertains to the second field of the frame.

Table 6-22 – timecode_type values

Value	Meaning
00	one timestamp for the frame
01	one timestamp for the first or only field
10	one timestamp for the second field
11	two timestamps, one for each of two fields

counting_type – A 3-bit integer that indicates the method used for compensating the nframes counting parameter of the frame or field capture timestamps to reduce drift accumulation in the remaining parameters of each timestamp.

Table 6-23 – counting_type values

Value	Meaning
000	nframes parameter not used
001	no dropping of nframes count values
010	dropping of individual zero values of nframes count
011	dropping of individual max_nframes values of nframes count
100	dropping of the two lowest (values 0 and 1) nframes counts when units_of_seconds and tens_of_seconds are zero and units_of_minutes is not zero
101	dropping of unspecified individual nframes count values
110	dropping of unspecified numbers of unspecified nframes count values
111	reserved

nframes_conversion_code – A 1-bit unsigned integer that indicates a conversion factor to be used in determining the amount of time indicated by the nframes parameters of each frame or field capture timestamp. The factor specified is 1000 + nframes_conversion_code.

clock_divisor – A 7-bit unsigned integer that contains the number of divisions of the 27 MHz system clock to be applied for generating the equivalent timestamp for each frame or field capture timestamp.

nframes_multiplier_upper, nframes_multiplier_lower – The most significant and least significant bits, respectively, of nframes_multiplier.

nframes_multiplier – An unsigned integer multiplier used for generating the equivalent timestamp for each frame or field capture timestamp as specified by nframes_multiplier_upper and nframes_multiplier_lower.

13) New subclause 6.3.21.2.1

Insert new subclause 6.3.21.2.1:

6.3.21.2.1 Frame or field capture timestamp

nframes – An 8-bit unsigned integer containing the number of frame time increments to add in deriving the equivalent timestamp. The value of nframes shall not be greater than the value of max_nframes as derived by the following formula:

max_nframes = (26 999 999) / (nframes_multiplier *
(1000 + nframes_conversion_code) * clock_divisor)

where "/" indicates the division operator defined in 4.1.

time_discontinuity – A 1-bit flag that indicates if a discontinuity in time or timebase between the previous timestamp and the current timestamp has occurred. If set to '0', the time difference that can be calculated between the current and previous timestamps is the ideal display duration of the previous frame or field. If set to '1', the time difference that can be calculated between the current and previous timestamps has no defined meaning. If editing occurs that results in time or timebase discontinuities or if the previous field or frame timestamp is unavailable, the time_discontinuity bit shall be set to '1'.

prior_count_dropped – A 1-bit flag indicating whether the counting of one or more values of the nframes parameter was dropped in order to reduce drift accumulation in the remaining parameters of the timestamp. Shall be zero if counting_type is '001'. Shall be zero if counting_type is '010' and nframes is not equal to 1. Shall be zero if counting_type is '011' and nframes is not equal to 0. Shall be zero if counting_type is '100' and nframes is not equal to 2.

time_offset_part_a – A 6-bit integer containing the most significant bits of time_offset.

time_offset_part_b – An 8-bit unsigned integer containing the second most significant bits of time_offset.

time_offset_part_c – An 8 bit unsigned integer containing the third most significant bits of time_offset.

time_offset_part_d – An 8-bit unsigned integer containing the least significant bits of time_offset.

time_offset – A two's complement signed 30-bit integer that is the number of clock cycles (in original 27 MHz system clock cycles or with a clock frequency modified by clock_divisor) offset from the time specified by the other parameters of the frame or field capture timestamp in order to specify the equivalent timestamp for when the current field or frame was captured. When counting_type is 0, the value of time_offset shall be constrained by the encoder to be less than 27 000 000 in magnitude.

units_of_seconds – A 4-bit unsigned integer that is used to calculate the equivalent timestamp. It represents the portion of the timestamp of this field or frame measured in seconds modulo 10. Table 6-24 defines the allowed range of values.

Table 6-24 – units_of_seconds values

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

tens_of seconds – A 4-bit unsigned integer that is used to calculate the equivalent timestamp. It represents the portion of the timestamp of this field or frame measured in seconds divided by 10. Table 6-25 defines the allowed range of values.

Table 6-25 – tens_of_seconds values

Value	Meaning
0000-0101	number of seconds/10
0110-1111	forbidden

units_of_minutes – A 4-bit integer that is used to calculate the equivalent timestamp. It represents the portion of the timestamp of this field or frame measured in minutes modulo 10. Table 6-26 defines the allowed range of values.

Table 6-26 – units_of_minutes values

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

tens_of_minutes – A 4-bit integer that is used to calculate the equivalent timestamp. It represents the portion of the timestamp of this field or frame measured in seconds divided by 10. Table 6-27 defines the allowed range of values.

Table 6-27 – tens_of_minutes values

Value	Meaning
0000-0101	number of minutes/10
0110-1111	forbidden

units_of_hours – A 4-bit integer that is used to calculate the equivalent timestamp. It represents the portion of the timestamp of this field or frame measured in hours modulo 10. Table 6-28 defines the allowed range of values. Shall not exceed a value of "3" if tens_of_hours is equal to "2".

Table 6-28 – units_of_hours values

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

tens_of_hours – A 4-bit integer 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 6-29 defines the allowed range of values.

Table 6-29 – tens_of_hours values

Value	Meaning
0000-0010	number of hours/10
0011-1111	forbidden

When counting_type is 0, an equivalent timestamp represented in 27 MHz system clock cycles is defined by 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

When counting_type is 0, the values of the parameters within the timestamp shall be constrained by the encoder such that equivalent_timestamp shall not be less than 0 and shall not exceed 2 332 799 999 999.

When counting_type is not 0, an equivalent timestamp represented in 27 MHz system clock cycles is defined by 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 +
(nframes * (nframes_multiplier * (1000 + nframes_conversion_code)) +
time_offset) * clock_divisor

When counting_type is not 0, the values of the parameters within the timecode shall be constrained by the encoder such that equivalent_timestamp shall not be less than 0.

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

14) New subclause 6.3.21.3

Insert new subclause 6.3.21.3:

6.3.21.3 Additional pan-scan parameters

Additional pan-scan parameters allows carriage of pan-scan information for more than one display type. For example, if the information encoded for a pan-scan process 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.

aspect_ratio_information – A 4-bit integer value that is defined in 6.3.3 (sequence header). A value for aspect_ratio_information which is equal to the value specified in the sequence_header() shall not occur.

display_size_present – A 1-bit flag, when set to '1', indicates the presence of the display_horizontal_size_upper, display_horizontal_size_lower, display_vertical_size_upper, and display_vertical_size_lower parameters. When set to '0', the previous values of display_horizontal_size and display_vertical_size corresponding to the value of aspect_ratio_information shall be used. For a specific aspect ratio, this field should be set to '1' in the first picture header after any sequence_header(). Following a sequence_header(), the value of display_horizontal_size and display_vertical_size shall be the value defined in the sequence_display_extension().

display_horizontal_size_upper – The 6 most significant bits of display_horizontal_size.

display_horizontal_size_lower – The 8 least significant bits of display_horizontal_size.

display_horizontal_size – A 14-bit integer value defined in 6.3.6 (sequence display extension). For any specific value of aspect_ratio_information, the value of this parameter shall remain the same for the sequence.

display_vertical_size_upper – The 6 most significant bits of display_vertical_size.

display_vertical_size_lower – The 8 least significant bits of display_vertical_size.

display_vertical_size – A 14-bit integer value defined in 6.3.6 (sequence display extension). For any specific value of aspect_ratio_information, the value of this parameter shall remain the same for the sequence.

frame_centre_horizontal_offset_upper, frame_centre_horizontal_offset_lower – The 8 most significant and least significant bits, respectively, of frame_centre_horizontal_offset.

frame_centre_horizontal_offset – A 16-bit signed integer defined in 6.3.12 (picture display extension).

frame_centre_vertical_offset_upper, **frame_centre_vertical_offset_lower** – The 8 most significant and least significant bits, respectively, of frame_centre_vertical_offset.

frame_centre_vertical_offset – A 16-bit signed integer defined in 6.3.12 (picture display extension). Following a sequence_header(), the value zero shall be used for all frame centre offsets until a picture_display_extension() defines non-zero values.

number_of_frame_centre_offsets – An integer defined in 6.3.12. Following a sequence header, the value zero shall be used for all frame centre offsets until a picture display extension defines non-zero values.

15) New subclause 6.3.21.4

Insert new subclause 6.3.21.4:

6.3.21.4 Active region window

The Active Region Window contains 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 6.3.3. No more than one active_region_window for each picture shall be present in the bitstream. When a frame is coded as two field pictures, the active region window shall not be present in the second field picture.

top_left_x_upper, top_left_x_lower – The 8 most significant and least significant bits, respectively, of top_left_x.

top_left_x – A 16-bit integer that defines the sample number within a line of the luminance component in the reconstructed frame that, together with top_left_y, specifies the upper left corner of the active_region_window's rectangle.

top_left_y_upper, top_left_y_lower – The 8 most significant and least significant bits, respectively, of top_left_y.

top_left_y – A 16-bit integer that defines the line number for the luminance component in the reconstructed frame that, together with top_left_x, specifies the upper left corner of the active_region_window's rectangle.

active_region_horizontal_size_upper, active_region_horizontal_size_lower – The 8 most significant and least significant bits, respectively, of active_region_horizontal_size.

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

active_region_vertical_size_upper, active_region_vertical_size_lower – The 8 most significant and least significant bits, respectively, of active_region_vertical_size.

active_region_vertical_size – See the definition for active_region_horizontal_size. This value shall not 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 frame does not have an active region window present in the bitstream, then the most recently decoded active region window shall be used. Following a sequence header, the active region window parameters active_region_horizontal_size and active_region_vertical_size shall be reset to the values of horizontal_size and vertical_size as defined in the sequence header and top_left_x and top_left_y shall be reset to 0.

16) New subclause 6.3.21.5

Insert new subclause 6.3.21.5:

6.3.21.5 Coded picture length

The coded picture length specifies 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 picture. Not more than one coded picture length for each picture shall be present in the bitstream.

picture_byte_count_part_a, picture_byte_count_part_b, picture_byte_count_part_c, picture_byte_count_part_d – The 8 most significant, second most significant, third most significant, and least significant bits, respectively, of the picture_byte_count.

picture_byte_count – A 32-bit unsigned integer that indicates the number of bytes starting with the first byte of the first slice_start_code of the current picture and ending with the byte preceding the start code prefix immediately following the last macroblock of that picture. The value '0' is permitted. The value '0' indicates that the length of the picture is unknown.

17) Subclause E.1

Replace Table E.7 with the following:

Table E.7 – Picture header

#	Syntactic elements	SIMPLE	MAIN	SNR	SPATIAL	HIGH	4:2:2	Multi-view	Type	Comments
01	**temporal_reference**	x	x	x	x	x	x	x	I	
02	**picture_coding_type**	x	x	x	x	x	x	x	I	Simple Profile: I, P at Main level, I, P, B at Low level Main, SNR, Spatial, High and Multi-view Profile: I, P, B
03	**vbv_delay**	x	x	x	x	x	x	x	I	
04	**full_pel_forward_vector**	x	x	x	x	x	x	x	I	Set to "0" for ITU-T Rec. H.262 | ISO/IEC 13818-2
05	**forward_f_code**	x	x	x	x	x	x	x	I	Set to "111" for ITU-T Rec. H.262 | ISO/IEC 13818-2
06	**full_pel_backward_vector**	x	x	x	x	x	x	x	I	Set to "0" for ITU-T Rec. H.262 | ISO/IEC 13818-2
07	**backward_f_code**	x	x	x	x	x	x	x	I	Set to "111" for ITU-T Rec. H.262 | ISO/IEC 13818-2
08	content_description_data()	x	x	x	x	x	x	x	I	
09	picture_coding_extension()	x	x	x	x	x	x	x	I	
10	quant_matrix_extension()	x	x	x	x	x	x	x	I	
11	picture_display_extension()	x	x	x	x	x	x	x	P	
12	picture_spatial_scalable_extension()	o	o	o	x	x	o	o	I	
13	picture_temporal_scalable_extension()	o	o	o	o	o	o	x	I	
14	camera_parameters_extension()	o	o	o	o	o	o	x	P	

18) New Annex K

Insert new Annex K:

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 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 progressive-scan film-frame material to 30 000/1001 frame per second video by a process known as 3:2 pull down (also known as 2:3 pull down). In this process, each set of four consecutive progressive scanned pictures, denoted as pictures A, B, C, and D, is converted to ten fields of video content by repeating the first field of the second and fourth pictures (pictures B and D). The same pattern is then repeated again and again for each subsequent set of four pictures.

Each film picture is scanned to create two fields of alternating lines and, in every sequence of four pictures, repeating the transmission of the first field of every second and fourth picture after sending the first and second fields of the picture and adjusting which field is sent first to maintain an alternating field pattern, thus converting every four film-frame 24 Hz pictures to ten fields of 29.97 Hz video as follows:

- The top field of the first film picture (picture A) is sent; then
- The bottom field of the first film picture (picture A) is sent; then
- The top field of the second film picture (picture B) is sent; then
- The bottom field of the second film picture (picture B) is sent; then
- The top field of the second film picture (picture B) is sent again (typically by setting repeat_first_field to 1); then
- The bottom field of the third film picture (picture C) is sent; then
- The top field of the third film picture (picture C) is sent; then
- The bottom field of the fourth film picture (picture D) is sent; then
- The top field of the fourth film picture (picture D) is sent; then
- The bottom field of the fourth film picture (picture D) is sent again (typically by setting repeat_first_field to 1); then
- The process above repeats in a modulo 10 manner for subsequent fields.

This process slows down the overall timing by a factor of 1001/1000 (creating a source with approximately 23.976 film pictures per second) and represents the film in a manner suitable for use on systems with interlaced displays. 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.