
**Information technology — Coding of
audio-visual objects —**

**Part 2:
Visual**

AMENDMENT 1: Studio profile

Technologies de l'information — Codage des objets audiovisuels —

Partie 2: Codage visuel

AMENDEMENT 1: Profil du studio



Reference number
ISO/IEC 14496-2:2001/Amd.1:2002(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/IEC 2002

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

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 14496-2:2001 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

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

AMENDMENT 1: Studio profile

- 1) Add the following text at the end of 'Overview of the object based non scalable syntax' of 'Introduction':

"

In order to preserve the lossless quality, or to restrict the maximum bit count of block data, the block based DPCM coding can be used for ISO/IEC 14496-2:2001 Amendment 1 (Studio Profile Amendment).

"

- 2) Replace text in 'Coding of Shapes' of 'Introduction',

"

In natural video scenes, VOPs are generated by segmentation of the scene according to some semantic meaning. For such scenes, the shape information is thus binary (binary shape). Shape information is also referred to as alpha plane. The binary alpha plane is coded on a macroblock basis by a coder which uses the context information, motion compensation and arithmetic coding.

"

with

"

In natural video scenes, VOPs are generated by segmentation of the scene according to some semantic meaning. For such scenes, the shape information is thus binary (binary shape). Shape information is also referred to as alpha plane. The binary alpha plane is coded on a macroblock basis by a coder which uses the context information, motion compensation and arithmetic coding. For high quality applications, the uncompressed binary alpha block coding is used.

"

- 3) Add the following text in 'Introduction' following 'Coding of Shapes':

"

Coding interlaced video

Each frame of interlaced video consists of two fields which are separated by one field-period. This part of ISO/IEC 14496 allows either the frame to be encoded as a VOP or the two fields to be encoded as two VOPs. Frame encoding or field encoding can be adaptively selected on a frame-by-frame basis. Frame encoding is typically preferred when the video scene contains significant detail with limited motion. Field encoding, in which the second field can be predicted from the first, works better when there is fast movement.

"

- 4) Replace text in 'Motion representation - macroblocks' of 'Introduction',

"

The choice of 16×16 blocks (referred to as macroblocks) for the motion-compensation unit is a result of the trade-off between the coding gain provided by using motion information and the overhead needed to represent it. Each macroblock can further be subdivided to 8×8 blocks for motion estimation and compensation depending on the overhead that can be afforded. In order to encode the highly active scene with higher vop rate, a Reduced Resolution VOP tool is provided. When this tool is used, the size of the macroblock used for motion compensation decoding is 32 x 32 pixels and the size of block is 16 x 16 pixels.

"

with

"

The choice of 16×16 blocks (referred to as macroblocks) for the motion-compensation unit is a result of the trade-off between the coding gain provided by using motion information and the overhead needed to represent it. Each macroblock can further be subdivided to 8×8 blocks for motion estimation and compensation depending on the overhead that can be afforded. In order to encode the highly active scene with higher vop rate, a Reduced Resolution VOP tool is provided. When this tool is used, the size of the macroblock used for motion compensation decoding is 32 x 32 pixels and the size of block is 16 x 16 pixels.

In frame encoding, the prediction from the previous reference frame can itself be either frame-based or field-based.

"

- 5) Replace text in 'Chrominance formats' of 'Introduction',

"

This part of ISO/IEC 14496 currently supports the 4:2:0 chrominance format.

"

with

"

This part of ISO/IEC 14496 currently supports the 4:2:0 chrominance format.

ISO/IEC 14496-2:2001 Amendment 1 also supports the 4:2:2 and 4:4:4 chrominance formats in addition.

"

- 6) Add the following text in 'Introduction' following 'Chrominance formats':

"

RGB color components

ISO/IEC 14496-2:2001 Amendment 1 supports coding of RGB color components. The resolution of each component shall be identical when input data is treated as RGB color components.

7) Add the following text at the end of 'Pixel depth' of 'Introduction':

"

ISO/IEC 14496-2:2001 Amendment 1 supports 8, 10 and 12 bits in luminance and chrominance or RGB planes.

"

8) Replace subclauses 3.38, 3.82, 3.107, and 3.131 with the following:

"

3.38 component: A matrix, block or single sample from one of the three matrices (luminance and two chrominance or green, blue and red color primaries) that make up a picture.

3.82 frame: A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and continuing through successive lines to the bottom of the frame. For interlaced video a frame consists of two fields, a top field and a bottom field. One of these fields will commence one field period later than the other.

3.107 macroblock: The four 8×8 blocks of luminance data and the two (for 4:2:0 chrominance format), four (for 4:2:2 chrominance format) or eight (for 4:4:4 chrominance format) corresponding 8×8 blocks of chrominance data coming from a 16×16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the sample data and sometimes to the coded representation of the sample values and other data elements defined in the macroblock header of the syntax defined in this part of ISO/IEC 14496. The usage is clear from the context.

3.131 picture: Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of N-bit numbers representing the luminance and two chrominance signals or rgb colour signals. A "coded VOP" was defined earlier. For progressive video, a picture is identical to a frame, while for interlaced video, a picture can refer to a frame, or the top field or the bottom field of the frame depending on the context.

"

9) Add the following subclauses in clause 3 and renumber the subsequent items.

"

3.6 B-field VOP: A field structure B-VOP.

3.7 B-frame VOP: A frame structure B-VOP.

3.20 bottom field: One of two fields that comprise a frame. Each line of a bottom field is spatially located immediately below the corresponding line of the top field.

3.33 coded B-frame: A B-frame VOP or a pair of B-field VOPs that is coded.

3.34 coded frame: A coded frame is a coded I-frame, a coded P-frame or a coded B-frame.

3.35 coded I-frame: An I-frame VOP or a pair of field VOPs that is coded where the first field VOP is an I-VOP and the second field VOP is an I-VOP or a P-VOP..

3.36 coded P-frame: A P-frame VOP or a pair of field VOPs that is coded.

- 3.42** **coded order:** The order in which the VOPs are transmitted and decoded. This order is not necessarily the same as the display order.
- 3.64** **display aspect ratio:** The ratio height/width (in spatial measurement units such as centimeters) of the intended display.
- 3.66** **display process:** The (non-normative) process by which reconstructed frames are displayed.
- 3.85** **fast forward playback:** The process of displaying a sequence, or parts of a sequence, of VOPs in display-order, faster than real-time.
- 3.86** **fast reverse playback:** The process of displaying a sequence, or parts of a sequence, of VOPs in the reverse of display order, faster than real-time.
- 3.88** **field:** For an interlaced video signal, a “field” is the assembly of alternate lines of a frame. Therefore an interlaced frame is composed of two fields, a top field and a bottom field.
- 3.89** **field-based prediction:** A prediction mode using only one field of the reference frame. The predicted block size is 16x16 luminance samples. Field-based prediction is not used in progressive frames.
- 3.90** **field period:** The reciprocal of twice the frame rate.
- 3.91** **field VOP; field structure VOP:** A field structure VOP is a coded VOP with vop_structure is equal to “Top field” or “Bottom field”.
- 3.99** **frame-based prediction:** A prediction mode using both fields of the reference frame.
- 3.102** **frame VOP; frame structure VOP:** A frame structure VOP is a coded VOP with vop_structure is equal to “Frame”.
- 3.103** **future reference frame (field):** A future reference frame (field) is a reference frame (field) that occurs at a later time than the current VOP in display order.
- 3.113** **I-field VOP:** A field structure I-VOP.
- 3.114** **I-frame VOP:** A frame structure I-VOP.
- 3.147** **RGB component:** A matrix, block or single sample representing one of the three primary colours. The symbols used for the rgb signals are Green, Blue and Red.
- 3.148** **P-field VOP:** A field structure P-VOP.
- 3.149** **P-frame VOP:** A frame structure P-VOP.

3.171 sample aspect ratio: (abbreviated to **SAR**). This specifies the relative distance between samples. It is defined (for the purposes of this specification) as the vertical displacement of the lines of luminance samples in a frame divided by the horizontal displacement of the luminance samples. Thus its units are (metres per line) ÷ (metres per sample)

3.182 skipped macroblock: A macroblock for which no data is encoded.

3.192 top field: One of two fields that comprise a frame. Each line of a top field is spatially located immediately above the corresponding line of the bottom field.

"

10) Add the following subclause 5.2.9 after subclause 5.2.8:

"

5.2.9 Definition of next_start_code_studio() function

The next_start_code_studio() function removes any zero bit and zero byte stuffing and locates the next start code.

next_start_code_studio() {	No. of bits	Mnemonic
while (!bytealigned())		
zero_bit	1	'0'
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
zero_byte	8	'0000 0000'
}		

This function checks whether the current position is byte aligned. If it is not, zero stuffing bits are present. After that any number of zero stuffing bytes may be present before the start code. Therefore start codes are always byte aligned and may be preceded by any number of zero stuffing bits.

"

11) Replace subclause 6.1.1 with the following:

"

6.1.1 Visual object sequence

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 profile_and_level_indication, and 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.

"

12) Replace subclause 6.1.2 with the following:

"

6.1.2 Visual object

A visual object commences with a `visual_object_start_code` and a visual object id, which are followed by a video object, a still texture object, a mesh object, or an FBA object.

For Studio Profiles, only video object type is supported.

"

13) Replace subclause 6.1.3 with the following:

"

6.1.3 Video object

A video object commences with a `video_object_start_code`, and is followed by one or more video object layers.

A video object layer commences with `video_object_layer_start_code` which may optionally be followed by `Group_of_StudioVideoObjectPlane()` and then by one or more coded VOPs. The order of the coded frames in the coded bitstream is the order in which the decoder processes them, which is not necessarily the display order.

"

14) Replace subclause 6.1.3.1 with the following:

"

6.1.3.1 Progressive and interlaced sequences

This part of ISO/IEC 14496 deals with coding of both progressive and interlaced sequences.

The sequence, at the output of the decoding process, consists of a series of reconstructed VOPs separated in time and are readied for display via the compositor.

For Studio Profiles particularly, the output of the decoding process for interlaced sequences consists of a series of reconstructed fields that are separated in time by a field period. The two fields of a frame may be coded separately (field-VOPs). Alternatively the two fields may be coded together as a frame (frame-VOPs). Both frame VOPs and field VOPs may be used in a single video sequence.

In progressive sequences each VOP in the sequence shall be a frame VOP. The sequence, at the output of the decoding process, consists of a series of reconstructed frames that are separated in time by a frame period.

"

15) Replace subclause 6.1.3.2 with the following :

"

6.1.3.2 Frame

A frame consists of three rectangular matrices of integers; a luminance matrix (Y), and two chrominance matrices (Cb and Cr).

The relationship between these Y, Cb and Cr components and the primary (analogue) Red, Green and Blue Signals (E'_R , E'_G and E'_B), the chromaticity of these primaries and the transfer characteristics of the source frame may be specified in the bitstream (or specified by some other means). This information does not affect the decoding process.

For Studio Profiles particularly, the three rectangular matrices can be the primary RGB colour matrices.

"

16) Add the following subclause in subclause 6.1.3 and renumber the subsequent items

"

6.1.3.3 Field

A field consists of every other line of samples in the three rectangular matrices of integers representing a frame.

A frame is the union of a top field and a bottom field. The top field is the field that contains the top-most line of each of the three matrices. The bottom field is the other one.

"

17) Replace subclause 6.1.3.3 with the following:

"

6.1.3.3 VOP

A reconstructed VOP is obtained by decoding a coded VOP. A coded VOP may have been derived from a progressive or interlaced frame or an interlaced field. A reconstructed VOP is either a reconstructed frame (when decoding a frame VOP), or one field of a reconstructed frame (when decoding a field VOP).

An I-frame VOP or a pair of field VOPs, where the first field VOP is an I-picture and the second field VOP is an I-VOP or a P-VOP, is called a coded I-frame.

A P-frame VOP or a pair of P-field VOPs is called a coded P-frame.

A B-frame VOP or a pair of B-field VOPs is called a coded B-frame.

A coded I-frame, a coded P-frame or a coded B-frame is called a coded frame.

6.1.1.4.1 Field VOPs

If field VOPs are used, then they shall occur in pairs (one top field followed by one bottom field, or one bottom field followed by one top field) and together constitute a coded frame. The two field VOPs that comprise a coded frame shall be encoded in the bitstream in the order in which they shall occur at the output of the decoding process.

When the first VOP of the coded frame is a P-field VOP, then the second VOP of the coded frame shall also be a P-field VOP. Similarly when the first VOP of the coded frame is a B-field VOP the second VOP of the coded frame shall also be a B-field VOP.

When the first VOP of the coded frame is a I-field VOP, then the second VOP of the frame shall be either an I-field VOP or a P-field VOP. If the second VOP is a P-field VOP, then certain restrictions apply (see 7.16.7.4.5).

6.1.1.4.2 Frame VOPs

When coding interlaced sequences using frame VOPs, the two fields of the frame shall be interleaved with one another and then the entire frame is coded as a single frame-VOP.

"

18) Replace the following text in subclause 6.1.3.5,

"

- 1) the modulo part (i.e. the full second units) of the time base for the next VOP after the GOV header in display order

"

with

"

- 1) the modulo part (i.e. the full second units) of the time base for the next VOP after the GOV header in display order. For Studio Profiles particularly, SMPTE 12M time code information that is not used by the decoding process.

"

19) Replace the following text in subclause 6.1.3.6,

"

6.1.3.6 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

The luminance and chrominance samples are positioned as shown in Figure 6-1. The two variations in the vertical and temporal positioning of the samples for interlaced VOPs are shown in Figure 6-2 and Figure 6-3.

Figure 6-4 shows the vertical and temporal positioning of the samples in a progressive frame.

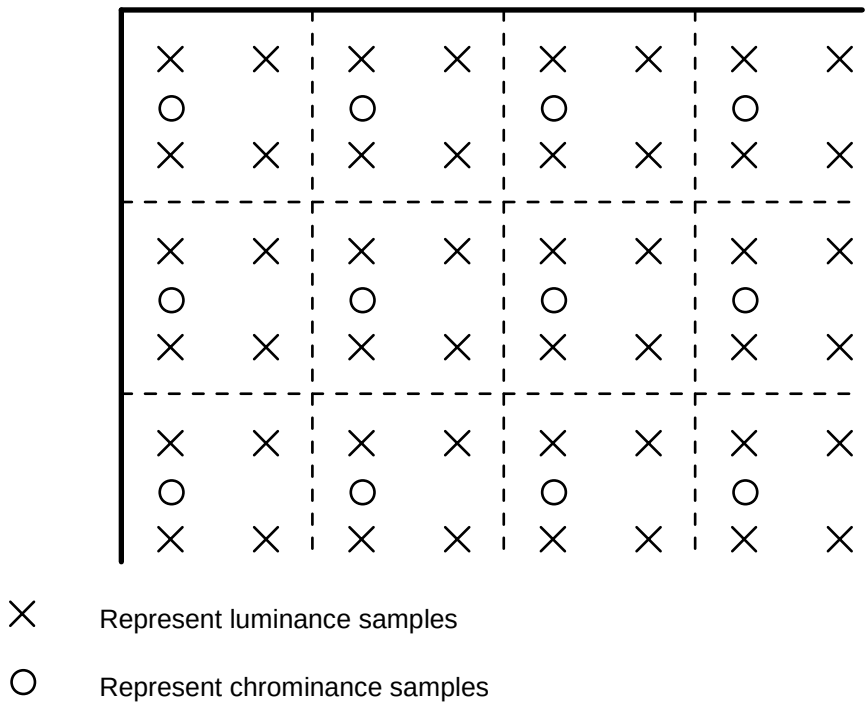


Figure 6-1 — The position of luminance and chrominance samples in 4:2:0 data

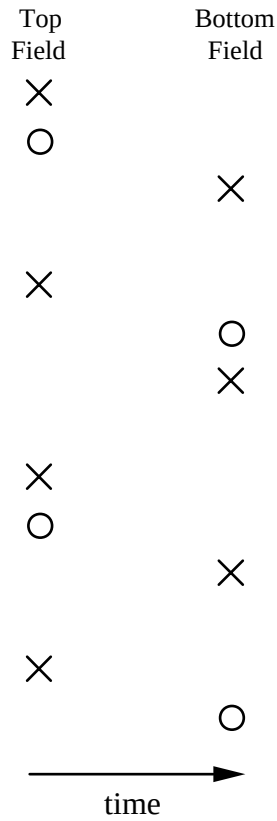


Figure 6-2 — Vertical and temporal positions of samples in an interlaced frame with top_field_first=1

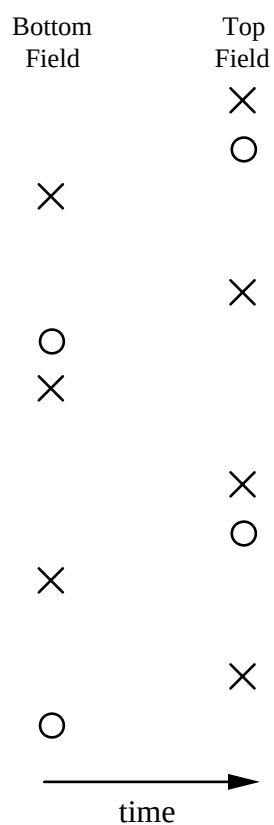


Figure 6-3 — Vertical and temporal position of samples in an interlaced frame with `top_field_first=0`

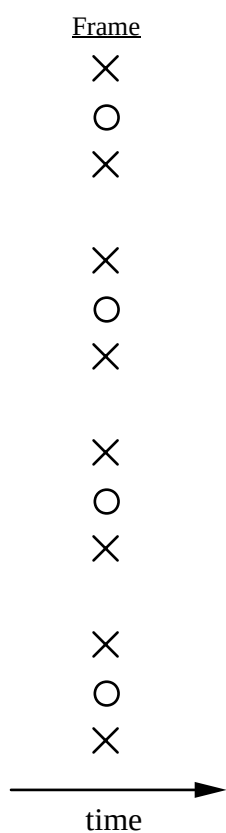


Figure 6-4 — Vertical and temporal positions of samples in a progressive frame

The binary alpha plane for each VOP is represented by means of a bounding rectangle as described in clause F.2, and it has always the same number of lines and pixels per line as the luminance plane of the VOP bounding rectangle. The positions between the luminance and chrominance pixels of the bounding rectangle are defined in this clause according to the 4:2:0 format. For the progressive case, each 2x2 block of luminance pixels in the bounding rectangle associates to one chrominance pixel. For the interlaced case, each 2x2 block of luminance pixels of the same field in the bounding rectangle associates to one chrominance pixel of that field.

In order to perform the padding process on the two chrominance planes, it is necessary to generate a binary alpha plane which has the same number of lines and pixels per line as the chrominance planes. Therefore, when non-scalable shape coding is used, this binary alpha plane associated with the chrominance planes is created from the binary alpha plane associated with the luminance plane by the subsampling process defined below:

For each 2x2 block of the binary alpha plane associated with the luminance plane of the bounding rectangle (of the same frame for the progressive and of the same field for the interlaced case), the associated pixel value of the binary alpha plane associated with the chrominance planes is set to 255 if any pixel of said 2x2 block of the binary alpha plane associated with the luminance plane equals 255.

"

with

"

6.1.3.6 Format

6.1.3.6.1 4:2:0 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in both horizontal and vertical dimensions. The Y-matrix shall have an even number of lines and samples.

If the matrices represent RGB colour primary matrices, this 4:2:0 format shall not be applied.

NOTE — When interlaced frames are coded as rectangular field VOPs, the VOP reconstructed from each of these field VOPs shall have a Y-matrix with half the number of lines of the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by four.

The luminance and chrominance samples are positioned as shown in Figure 6-1. The two variations in the vertical and temporal positioning of the samples for interlaced VOPs are shown in Figure 6-2 and Figure 6-3.

Figure 6-4 shows the vertical and temporal positioning of the samples in a progressive frame.

In each field of an interlaced frame, the chrominance samples do not lie (vertically) mid way between the luminance samples of the field. This is so that the spatial location of the chrominance samples in the frame is the same whether the frame is represented as a single frame-VOP or two field-VOPs.

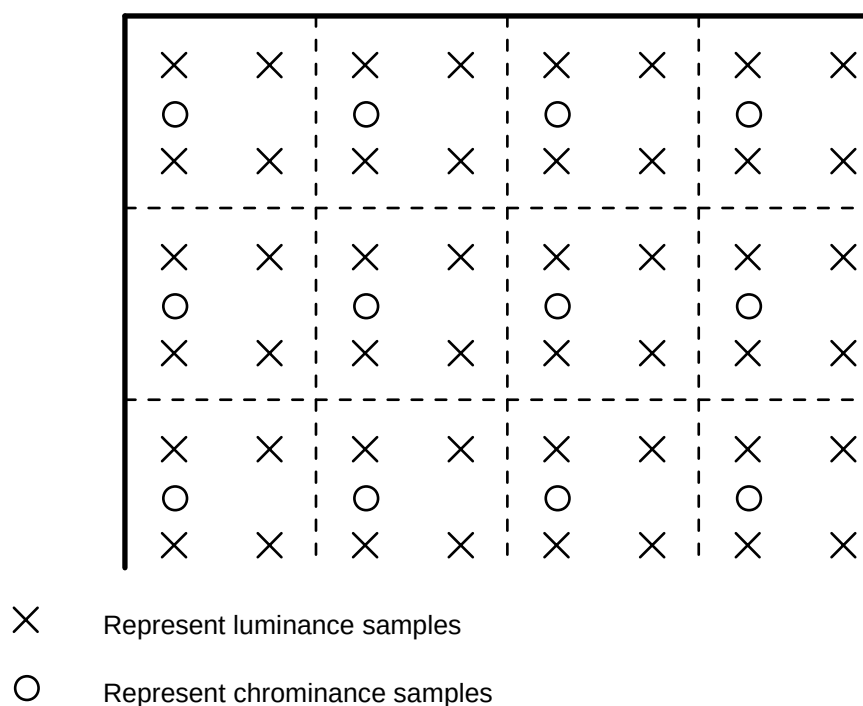


Figure 6-1 — The position of luminance and chrominance samples in 4:2:0 data

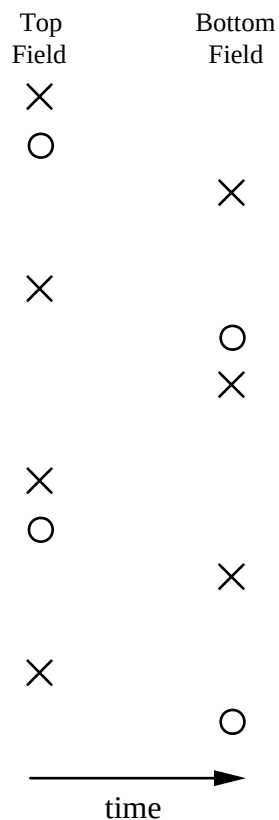


Figure 6-2 — Vertical and temporal positions of samples in an interlaced frame with top_field_first=1

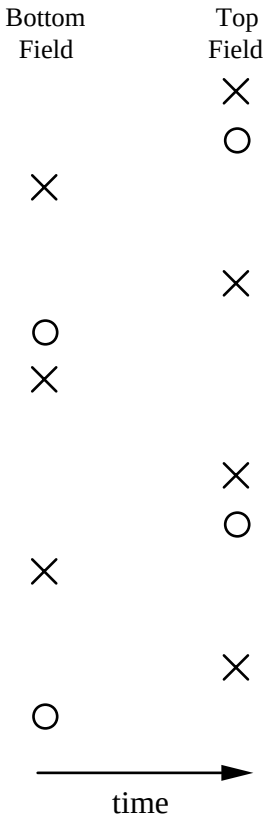


Figure 6-3 — Vertical and temporal position of samples in an interlaced frame with top_field_first=0

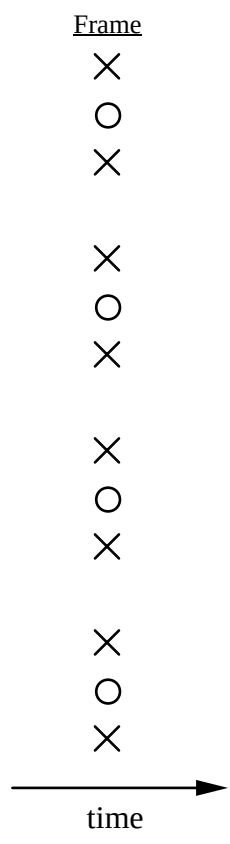


Figure 6-4 — Vertical and temporal positions of samples in a progressive frame

The binary alpha plane for each VOP is represented by means of a bounding rectangle as described in clause F.2, and it always has the same number of lines and pixels per line as the luminance plane of the VOP bounding rectangle. The positions between the luminance and chrominance pixels of the bounding rectangle are defined in this clause according to the 4:2:0 format. For the progressive case, each 2x2 block of luminance pixels in the bounding rectangle associates to one chrominance pixel. For the interlaced case, each 2x2 block of luminance pixels of the same field in the bounding rectangle associates to one chrominance pixel of that field.

In order to perform the padding process on the two chrominance planes, it is necessary to generate a binary alpha plane which has the same number of lines and pixels per line as the chrominance planes. Therefore, when non-scalable shape coding is used, this binary alpha plane associated with the chrominance planes is created from the binary alpha plane associated with the luminance plane by the subsampling process defined below:

For each 2x2 block of the binary alpha plane associated with the luminance plane of the bounding rectangle (of the same frame for the progressive and of the same field for the interlaced case), the associated pixel value of the binary alpha plane associated with the chrominance planes is set to 255 if any pixel of said 2x2 block of the binary alpha plane associated with the luminance plane equals 255.

6.1.3.6.2 4:2:2 Format

In this format the Cb and Cr matrices shall be one half the size of the Y-matrix in the horizontal dimension and the same size as the Y-matrix in the vertical dimension. The Y-matrix shall have an even number of samples.

If the matrices represent RGB colour primar matrices, this 4:2:2 format shall not be applied.

NOTE — When interlaced frames are coded as rectangular field VOPs, the VOP reconstructed from each of these field VOPs shall have a Y-matrix with half the number of lines of the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by two.

The luminance and chrominance samples are positioned as shown in Figure AMD1-1.

In order to clarify the organisation, Figure AMD1-2 shows the (vertical) positioning of the samples when the frame is separated into two fields.

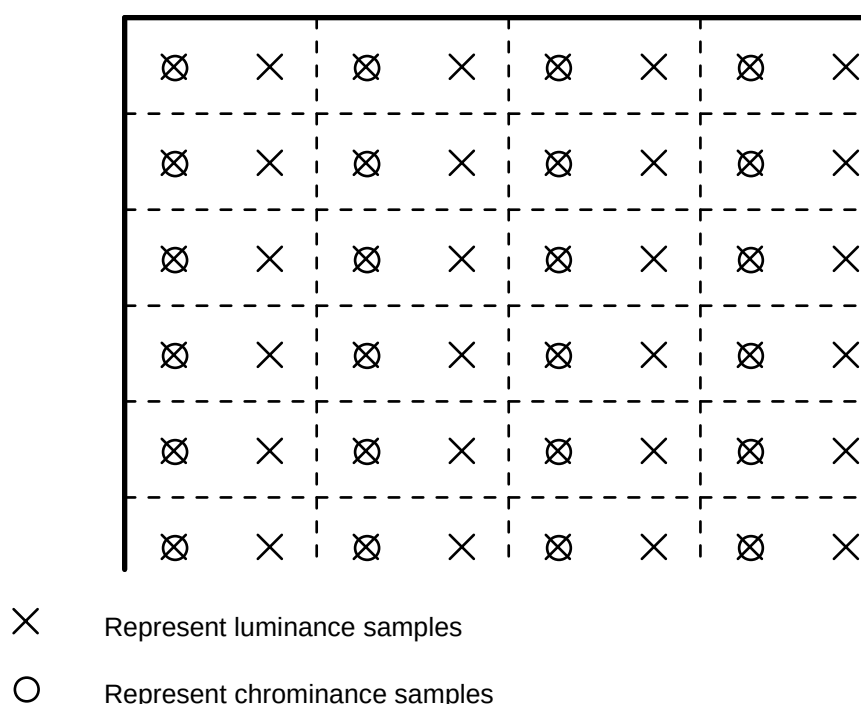


Figure AMD1-1 — The position of luminance and chrominance samples. 4:2:2 data.

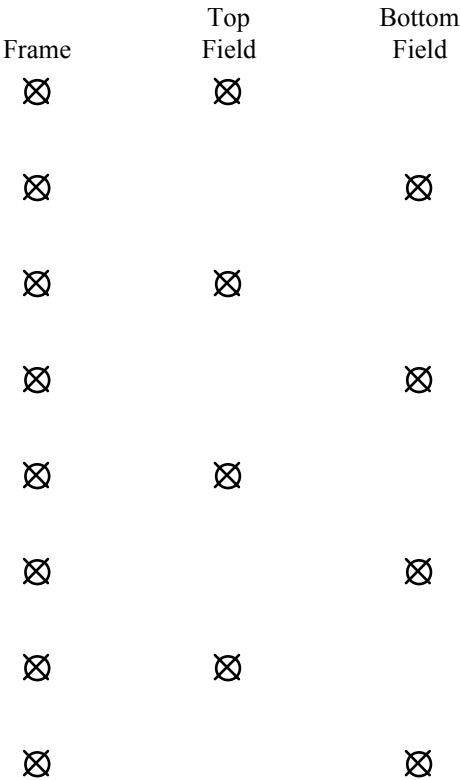


Figure AMD1-2 — Vertical positions of samples with 4:2:2 and 4:4:4 data

6.1.3.6.3 4:4:4 Format

In this format the Cb and Cr matrices shall be the same size as the Y-matrix in the horizontal and the vertical dimensions.

If the matrices are treated as RGB colour primary matrices, the matrices shall follow this format.

NOTE — When interlaced frames are coded as field rectangular VOPs, the VOP reconstructed from each of these field VOPs shall have a Y-matrix with half the number of lines of the corresponding frame. Thus the total number of lines in the Y-matrix of an entire frame shall be divisible by two.

The luminance and chrominance samples are positioned as shown in Figures AMD1-2 and AMD1-3.

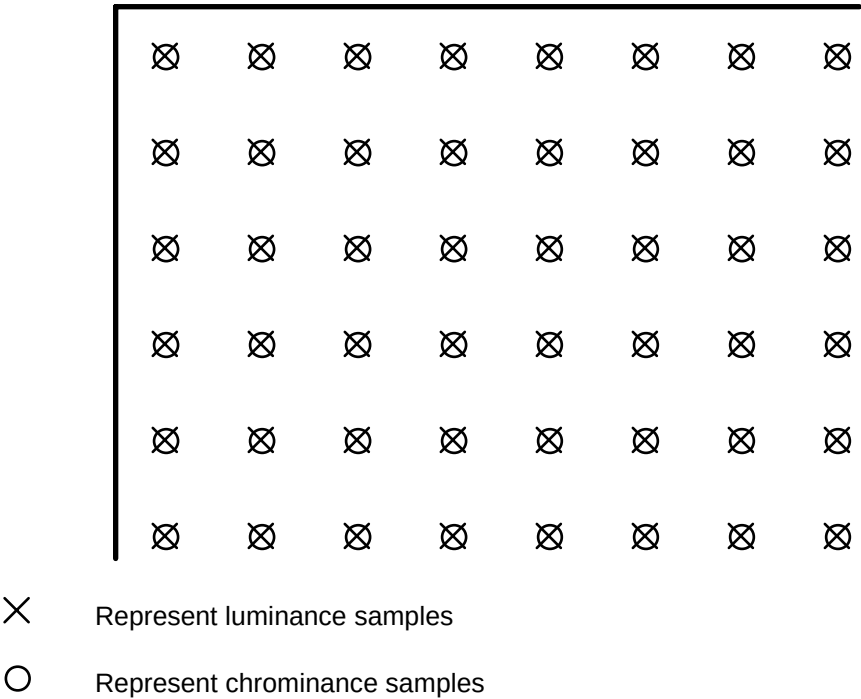


Figure AMD1-3 — The position of luminance and chrominance samples. 4:4:4 data.

"

20) Replace the following text in subclause 6.1.3.8,

"

A macroblock contains a section of the luminance component and the spatially corresponding chrominance components. The term macroblock can either refer to source and decoded data or to the corresponding coded data elements. A skipped macroblock is one for which no information is transmitted. Presently there is only one chrominance format for a macroblock, namely, 4:2:0 format. The orders of blocks in a macroblock is illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in Figure 6-5.

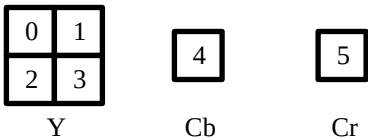


Figure 6-5 — 4:2:0 Macroblock structure

The organisation of VOPs into macroblocks is as follows.

For the case of a progressive VOP, the interlaced flag (in the VOP header) is set to “0” and the organisation of lines of luminance VOP into macroblocks is called frame organization and is illustrated in Figure 6-6. In this case, frame DCT coding is employed.

For the case of interlaced VOP, the interlaced flag is set to “1” and the organisation of lines of luminance VOP into macroblocks can be either frame organization or field organization and thus both frame and field DCT coding may be used in the VOP.

- In the case of frame DCT coding, each luminance block shall be composed of lines from two fields alternately. This is illustrated in Figure 6-6.
- In the case of field DCT coding, each luminance block shall be composed of lines from only one of the two fields. This is illustrated in Figure 6-7.

Only frame DCT coding is applied to the chrominance blocks. It should be noted that field based predictions may be applied for these chrominance blocks which will require predictions of 8x4 regions (after half-sample filtering).

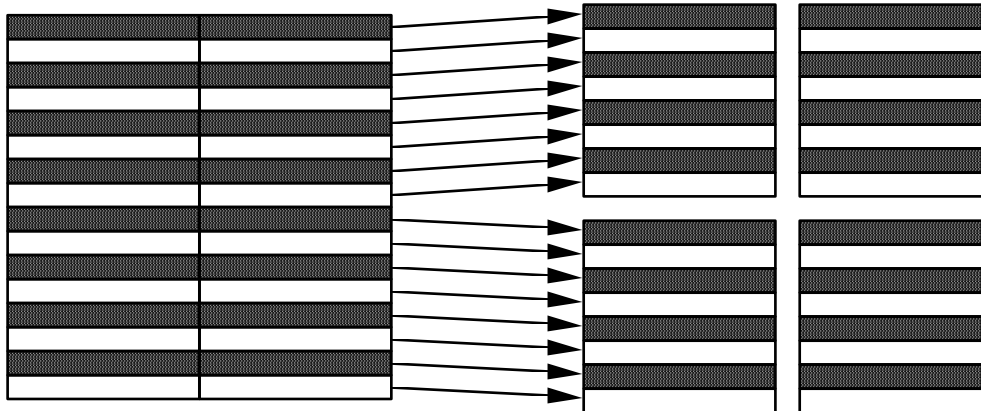


Figure 6-6 — Luminance macroblock structure in frame DCT coding

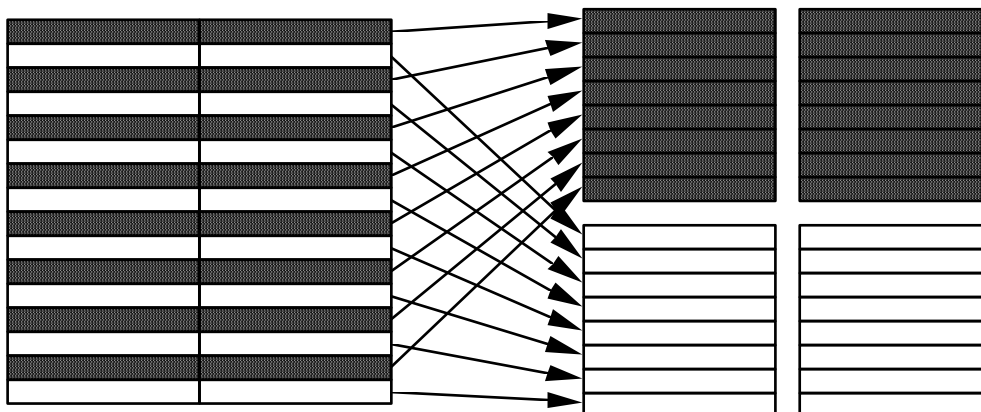


Figure 6-7 — Luminance macroblock structure in field DCT coding

"

with
"

A macroblock contains a section of the luminance component and the spatially corresponding chrominance components. The term macroblock can either refer to source and decoded data or to the corresponding coded data elements. A skipped macroblock is one for which no information is transmitted. There are three chrominance formats for a macroblock, namely, 4:2:0, 4:2:2 and 4:4:4 formats. The order of blocks in a macroblock shall be different for each different chrominance format and are illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in Figure 6-5.

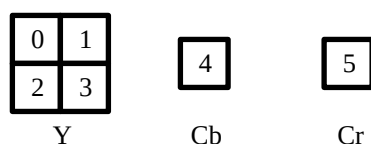


Figure 6-5 — 4:2:0 Macroblock structure

A 4:2:2 Macroblock consists of 8 blocks. This structure holds 4 Y, 2 Cb and 2 Cr Blocks and the block order is depicted in Figure AMD1-4.

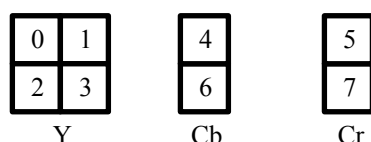


Figure AMD1-4 — 4:2:2 Macroblock structure

A 4:4:4 Macroblock consists of 12 blocks. This structure holds 4 Y, 4 Cb and 4 Cr (or 4 G, 4 B and 4 R) Blocks and the block order is depicted in Figure AMD1-5.

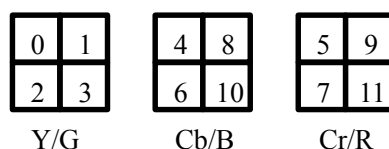


Figure AMD1-5 — 4:4:4 Macroblock structure

In frame VOPs, where both frame and field DCT coding may be used, the internal organisation within the macroblock is different in each case.

- In the case of frame DCT coding, each block shall be composed of lines from two fields alternately. This is illustrated in Figure 6-6.
- In the case of field DCT coding, each block shall be composed of lines from only one of the two fields. This is illustrated in Figure 6-7.

In the case of chrominance blocks the structure depends upon the chrominance format that is being used. In the case of 4:2:2 and 4:4:4 formats (where there are two blocks in the vertical dimension of the macroblock) the chrominance blocks are treated in exactly the same manner as the luminance blocks. However, in the 4:2:0 format the chrominance blocks shall always be organised in frame structure for the purposes of DCT coding. It should however be noted that field based predictions may be made for these blocks which will, in the general case, require that predictions for 8x4 regions (after half-sample filtering) must be made.

In field pictures, each picture only contains lines from one of the fields. In this case each block consists of lines taken from successive lines in the picture as illustrated by Figure 6-6.

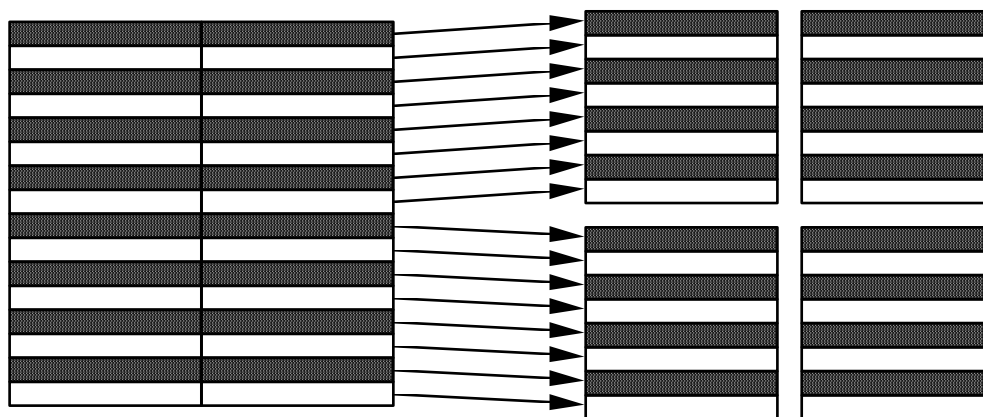


Figure 6-6 — Luminance macroblock structure in frame DCT coding

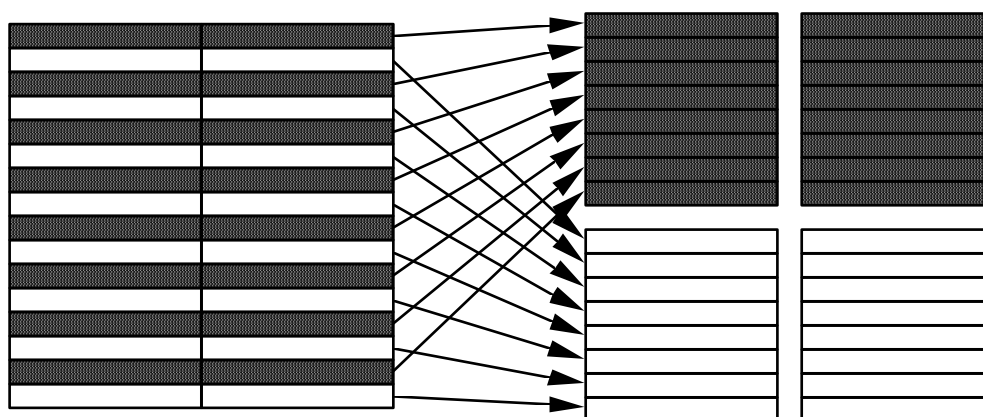


Figure 6-7 — Luminance macroblock structure in field DCT coding

"

21) Add the following subclause 6.1.3.10 after subclause 6.1.3.9:

"

6.1.3.10 Field

A field consists of every other line of samples in the three rectangular matrices of integers representing a frame.

A frame is the union of a top field and a bottom field. The top field is the field that contains the top-most line of each of the three matrices. The bottom field is the other one.

Only when `profile_and_level_indication` indicates the studio profile, a coded VOP may be a frame VOP or a field VOP. A reconstructed VOP is either a reconstructed frame (when decoding a frame VOP), or one field of a reconstructed frame (when decoding a field VOP).

6.1.3.10.1 Field VOPs

If field VOPs are used then they shall occur in pairs (one top field followed by one bottom field, or one bottom field followed by one top field) and together constitute a coded frame. The two field VOPs that comprise a coded frame shall be encoded in the bitstream in the order in which they shall occur at the output of the decoding process.

When the first VOP of the coded frame is a P-field VOP, then the second VOP of the coded frame shall also be a P-field VOP. Similarly when the first VOP of the coded frame is a B-field VOP the second VOP of the coded frame shall also be a B-field VOP.

When the first VOP of the coded frame is a I-field VOP, then the second VOP of the frame shall be either an I-field VOP or a P-field VOP. If the second VOP is a P-field VOP then certain restrictions apply,.

6.1.3.10.2 Frame VOPs

When coding interlaced sequences using frame VOPs, the two fields of the frame shall be interleaved with one another and then the entire frame is coded as a single frame-VOP.

"

22) Add the following subclauses 6.1.3.11 after subclause 6.1.3.10:

"

6.1.3.11 Slice

A slice is a series of an arbitrary number of consecutive macroblocks. The first and last macroblocks of a slice shall not be skipped macroblocks. Every slice shall contain at least one macroblock. Slices shall not overlap. The position of slices may change from picture to picture.

The first and last macroblock of a slice shall be in the same horizontal row of macroblocks.

Slices shall occur in the bitstream in the order in which they are encountered, starting at the upper-left of the picture and proceeding by raster-scan order from left to right and top to bottom (illustrated in the Figures of this clause as alphabetical order).

6.1.3.11.1 The general slice structure

In the most general case it is not necessary for the slices to cover the entire picture. Figure AMD1-6 shows this case. Those areas that are not enclosed in a slice are not encoded and no information is encoded for such areas (in the specific picture).

If the slices do not cover the entire picture then it is a requirement that if the picture is subsequently used to form predictions then predictions shall only be made from those regions of the picture that were enclosed in slices. It is the responsibility of the encoder to ensure this.

This specification does not define what action a decoder shall take in the regions between the slices.

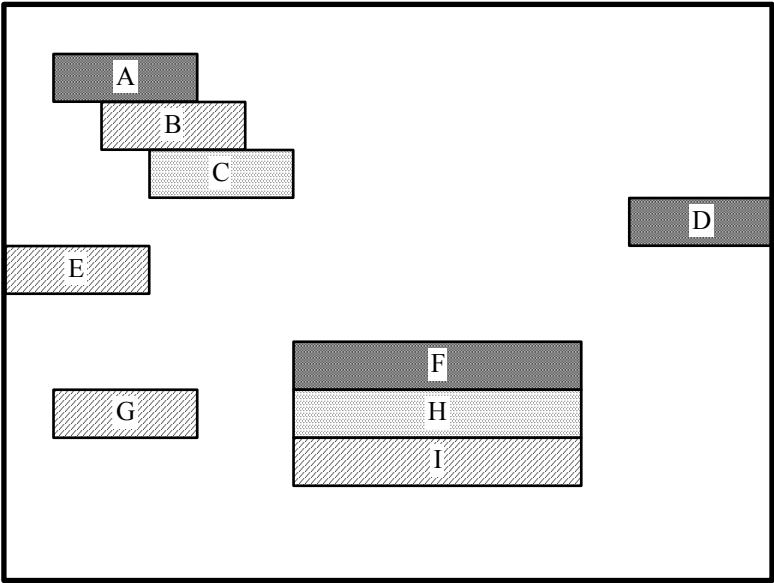


Figure AMD1-6 — The most general slice structure.

6.1.3.11.2 Restricted slice structure

In certain defined levels of defined profiles a restricted slice structure illustrated in Figure AMD1-7 shall be used. In this case every macroblock in the picture shall be enclosed in a slice.

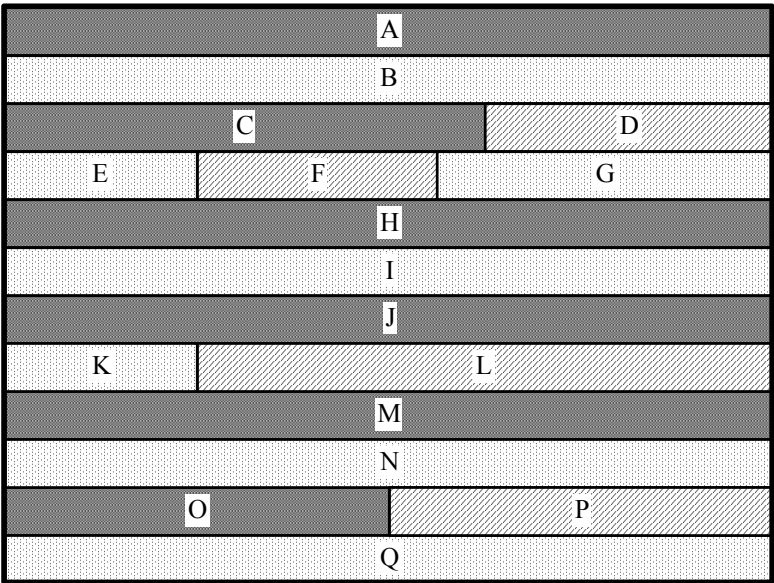


Figure AMD1-7 — Restricted slice structure.

Where a defined level of a defined profile requires that the slice structure obeys the restrictions detailed in this clause, the term “restricted slice structure” may be used.

”

23) Add the following text in subclause 6.2.1 before paragraph 5 (after Table 6-2):

"

Only when profile_and_level_indication indicates a studio profile, byte alignment shall be achieved by inserting bits with the value zero before the start code prefix such that the first bit of the start code prefix is the first (most significant) bit of a byte.

"

24) Replace Table 6-3 in subclause 6.2.1 with the following:

"

Table 6-3 — Start code values

name	start code value (hexadecimal)
video_object_start_code	00 through 1F
video_object_layer_start_code	20 through 2F
reserved	30 through AF
visual_object_sequence_start_code	B0
visual_object_sequence_end_code	B1
user_data_start_code	B2
group_of_vop_start_code	B3
video_session_error_code	B4
visual_object_start_code	B5
vop_start_code	B6
slice_start_code	B7
extension_start_code	B8
reserved	B9
fba_object_start_code	BA
fba_object_plane_start_code	BB
mesh_object_start_code	BC
mesh_object_plane_start_code	BD
still_texture_object_start_code	BE
texture_spatial_layer_start_code	BF
texture_snr_layer_start_code	C0
texture_tile_start_code	C1
texture_shape_layer_start_code	C2
reserved	C3-C5
System start codes (see Note)	C6 through FF
NOTE — System start codes are defined in ISO/IEC 14496-1:1999.	

"

25) Replace VisualObjectSequence() in subclause 6.2.2 with the following:

"

VisualObjectSequence() {	No. of bits	Mnemonic
do {		
visual_object_sequence_start_code	32	bslbf
profile_and_level_indication	8	uimsbf
if (profile_and_level_indication == 11100001-11101000) {		
next_start_code_studio()		
extension_and_user_data(0)		
StudioVisualObject()		
} else {		
while (next_bits() == user_data_start_code) {		
user_data()		
}		
VisualObject()		
}		
} while (nextbits() != visual_object_sequence_end_code)		
visual_object_sequence_end_code	32	bslbf
}		

"

26) Add the following subclause 6.2.13 after subclause 6.2.12:

"

6.2.13 Studio Video Object

6.2.13.1 Studio Visual Object

StudioVisualObject() {	No. of bits	Mnemonic
visual_object_start_code	32	bslbf
visual_object_verid	4	uimsbf
visual_object_type	4	uimsbf
next_start_code_studio()		
extension_and_user_data(1)		
if (visual_object_type == "video ID") {		
video_object_start_code	32	bslbf
StudioVideoObjectLayer()		
} else {		
/* Other visual object types are not supported in StudioVisualObject() */		
}		
}		

6.2.13.2 Extension and user data

extension_and_user_data(i) {	No. of bits	Mnemonic
while ((next_bits() == extension_start_code)		
(next_bits() == user_data_start_code)) {		
if ((i==2 i==4) &&		
(next_bits() == extension_start_code))		
extension_data(i)		
if (next_bits() == user_data_start_code)		
user_data_studio()		
}		
}		

6.2.13.2.1 Extension data

extension_data(i) {	No. of bits	Mnemonic
while (next_bits()== extension_start_code) {		
extension_start_code	32	bslbf
/* NOTE - i never takes the value 0 because extension_data() is never called in a VisualObjectSequence() */		
/* NOTE - i never takes the value 1 because extension_data() is never called in a StudioVisualObject() */		
if (i == 2) { /* Called in StudioVideoObjectLayer() */		
if (next_bits() == "Sequence Display Extension ID")		
sequence_display_extension()		
else if (next_bits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "VLC Code Extension ID")		
vlc_code_extension()		
}		
/* NOTE - i never takes the value 3 because extension_data() is never called in a Group_of_StudioVideoObjectPlane() */		
if (i == 4) { /* Called in VideoObjectPlane() */		
if (nextbits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "Copyright Extension ID")		
copyright_extension()		
else if (nextbits() == "Picture Display Extension ID")		
picture_display_extension()		
else if(nextbits() == "Camera Prameters Extension ID")		
camera_parameters_extension()		
else if (nextbits() == "ITU-T Extension ID")		
ITU-T_extension()		
else if (nextbits() == "VLC Code Extension ID")		
vlc_code_extension()		
}		
}		

6.2.13.2.2 User data Studio

user_data_studio() {	No. of bits	Mnemonic
user_data_start_code	32	bslbf
while(next_bits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	uimsbf
}		
next_start_code_studio()		
}		

6.2.13.2.3 Sequence display extension

sequence_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
video_format	3	uimsbf
video_range	1	bslbf
colour_description	1	uimsbf
if (colour_description) {		
colour_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coefficients	8	uimsbf
}		
display_horizontal_size	14	uimsbf
marker_bit	1	bslbf
display_vertical_size	14	uimsbf
next_start_code_studio()		
}		

6.2.13.2.4 Quant matrix extension

quant_matrix_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
load_intra_quantiser_matrix	1	uimsbf
if (load_intra_quantiser_matrix)		
intra_quantiser_matrix[64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix	1	uimsbf
if (load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_intra_quantiser_matrix	1	uimsbf
if (load_chroma_intra_quantiser_matrix)		
chroma_intra_quantiser_matrix[64]	8 * 64	uimsbf
load_chroma_non_intra_quantiser_matrix	1	uimsbf
if (load_chroma_non_intra_quantiser_matrix)		
chroma_non_intra_quantiser_matrix[64]	8 * 64	uimsbf
if (video_object_layer_shape == 'grayscale') {		
for(i=0; i<aux_comp_count; i++) {		
load_intra_quantiser_matrix_grayscale[i]	1	uimsbf
if (load_intra_quantiser_matrix_grayscale[i])		
intra_quantiser_matrix_grayscale [i][64]	8 * 64	uimsbf
load_non_intra_quantiser_matrix_grayscale[i]	1	uimsbf
if (load_non_intra_quantiser_matrix_grayscale[i])		
non_intra_quantiser_matrix_grayscale [i][64]	8 * 64	uimsbf
}		
}		
next_start_code_studio()		
}		

6.2.13.2.5 Picture display extension

picture_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
for (i=0; i<number_of_frame_centre_offsets; i++) {		
frame_centre_horizontal_offset	16	simsbf
marker_bit	1	bslbf
frame_centre_vertical_offset	16	simsbf
marker_bit	1	bslbf
}		
next_start_code_studio()		
}		

6.2.13.2.6 Copyright extension

copyright_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
copyright_flag	1	uimsbf
copyright_identifier	8	uimsbf
original_or_copy	1	uimsbf
reserved	7	bslbf
marker_bit	1	bslbf
copyright_number_1	20	uimsbf
marker_bit	1	bslbf
copyright_number_2	22	uimsbf
marker_bit	1	bslbf
copyright_number_3	22	uimsbf
next_start_code_studio()		
}		

6.2.13.2.7 Camera Parameters extension

camera_parameters_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
reserved	1	uimsbf
camera_id	7	simsbf
marker_bit	1	bslbf
height_of_image_device	22	uimsbf
marker_bit	1	bslbf
focal_length	22	uimsbf
marker_bit	1	bslbf
f_number	22	uimsbf
marker_bit	1	bslbf
vertical_angle_of_view	22	uimsbf
marker_bit	1	bslbf
camera_position_x_upper	16	simsbf
marker_bit	1	bslbf
camera_position_x_lower	16	
marker_bit	1	bslbf
camera_position_y_upper	16	simsbf
marker_bit	1	bslbf
camera_position_y_lower	16	
marker_bit	1	bslbf
camera_position_z_upper	16	simsbf
marker_bit	1	bslbf
camera_position_z_lower	16	
marker_bit	1	bslbf
camera_direction_x	22	simsbf
marker_bit	1	bslbf
camera_direction_y	22	simsbf
marker_bit	1	bslbf
camera_direction_z	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_x	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_y	22	simsbf
marker_bit	1	bslbf
image_plane_vertical_z	22	simsbf
marker_bit	1	bslbf
reserved	32	bslbf
next_start_code_studio()		
}		

6.2.13.2.8 ITU-T extension

ITU-T extension () {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
ITU-T_data	1	bslbf
}		
next_start_code_studio()		
}		
NOTE – The construct with the while-statement prevents start code emulation.		

6.2.13.2.9 VLC code extension

vlc_code_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
load_vlc_code	4	uimsbf
if (load_intra_vlc_code == 1) {		
for (j=0; j< 12; j++) {		
for (i=0; i< 22; i++) {		
Intra_vlc_length[j][i]	4	uimsbf
Intra_vlc_code[j][i]	16	uimsbf
}		
}		
}		
if (load_inter_vlc_code == 1) {		
for (j=0; j< 12; j++) {		
for (i=0; i< 22; i++) {		
inter_vlc_length[j][i]	4	uimsbf
inter_vlc_code[j][i]	16	uimsbf
}		
}		
}		
next_start_code_studio()		
}		

6.2.13.3 Studio Video Object Layer

StudioVideoObjectLayer() {	No. of bits	Mnemonic
video_object_layer_start_code	32	bslbf
random_accessible_vol	1	bslbf
video_object_type_indication	8	uimsbf
video_object_layer_verid	4	uimsbf
video_object_layer_shape	2	uimsbf
video_object_layer_shape_extension	4	uimsbf
progressive_sequence	1	bslbf
if (video_object_layer_shape != "binary only") {		
rgb_components	1	uimsbf
chroma_format	2	uimsbf

bits_per_pixel	4	uimsbf
}		
if (video_object_layer_shape == "rectangular") {		
marker_bit	1	bslbf
video_object_layer_width	14	uimsbf
marker_bit	1	bslbf
video_object_layer_height	14	uimsbf
marker_bit	1	bslbf
}		
aspect_ratio_info	4	uimsbf
if (aspect_ratio_info == "extended_PAR") {		
par_width	8	uimsbf
par_height	8	uimsbf
}		
frame_rate_code	4	uimsbf
first_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
latter_half_bit_rate	15	uimsbf
marker_bit	1	bslbf
first_half_vbv_buffer_size	15	uimsbf
marker_bit	1	bslbf
latter_half_vbv_buffer_size	3	uimsbf
first_half_vbv_occupancy	11	uimsbf
marker_bit	1	bslbf
latter_half_vbv_occupancy	15	uimsbf
marker_bit	1	bslbf
low_delay	1	uimsbf
mpeg2_stream	1	uimsbf
if (video_object_layer_shape == "grayscale") {		
for(i=0; i<aux_comp_count; i++) {		
alpha_bits_per_pixel[i]	4	uimsbf
minimum_alpha_level[i]	12	uimsbf
maximum_alpha_level[i]	12	uimsbf
}		
composition_method	1	bslbf
linear_composition	1	bslbf
}		
if (video_object_layer_shape != "binary only") {		
sprite_enable	1	bslbf
if (sprite_enable) {		
sprite_width	20	uimsbf
marker_bit	1	bslbf
sprite_height	20	uimsbf
marker_bit	1	bslbf
sprite_left_coordinate	20	uimsbf
marker_bit	1	bslbf
sprite_top_coordinate	20	uimsbf
marker_bit	1	bslbf
no_of_sprite_warping_points	6	uimsbf

sprite_warping_accuracy	2	uimsbf
sprite_brightness_change	1	bslbf
sprite_defocusing	1	bslbf
sprite_lens_distortion	1	bslbf
}		
}		
next_start_code_studio()		
extension_and_user_data(2)		
if (sprite_enable)		
StudioVideoObjectPlane()		
do {		
if (next_bits() == group_of_vop_start_code)		
Group_of_StudioVideoObjectPlane()		
StudioVideoObjectPlane()		
} while ((next_bits() == group_of_vop_start_code)		
(next_bits() == vop_start_code))		
}		

6.2.13.4 Group of Studio Video Object Plane

Group_of_StudioVideoObjectPlane() {	No. of bits	Mnemonic
group_vop_start_codes	32	bslbf
time_code_smpte12m	64	
closed_gov	1	bslbf
broken_link	1	bslbf
next_start_code_studio()		
extension_and_user_data(3)		
}		

6.2.13.5 Studio Video Object Plane

StudioVideoObjectPlane() {	No. of bits	Mnemonic
vop_start_code	32	bslbf
time_code_smpte12m	64	
temporal_reference	10	bslbf
vop_structure	2	uimsbf
vop_coding_type	2	uimsbf
vop_coded	1	bslbf
if (vop_coded == '0') {		
next_start_code_studio()		
extension_and_user_data(4)		
return()		
}		
if (video_object_layer_shape != "rectangular") {		
if (!(sprite_enable && vop_coding_type == "I")) {		
vop_width	14	uimsbf
marker_bit	1	bslbf
vop_height	14	uimsbf

marker_bit	1	bslbf
vop_horizontal_mc_spatial_ref	14	simsbf
marker_bit	1	bslbf
vop_vertical_mc_spatial_ref	14	simsbf
}		
}		
top_field_first	1	bslbf
repeat_first_field	1	bslbf
progressive_frame	1	bslbf
if (vop_coding_type == "I")		
intra_predictors_reset	1	bslbf
}		
if (video_object_layer_shape != "binary only") {		
alternate_scan	1	bslbf
frame_pred_frame_dct	1	bslbf
dct_precision	2	uimsbf
intra_dc_precision	2	uimsbf
q_scale_type	1	bslbf
if ((vop_coding_type != "I")		
&& !(sprite_enable && vop_coding_type == "S")) {		
vop_fcode[0][0]	4	uimsbf
vop_fcode[0][1]	4	uimsbf
vop_fcode[1][0]	4	uimsbf
vop_fcode[1][1]	4	uimsbf
dead_zone_disable	1	bslbf
}		
}		
if (video_object_layer_shape == "grayscale") {		
for(i=0; i<aux_comp_count; i++) {		
alpha_dct_precision[i]	2	uimsbf
alpha_intra_dc_precision[i]	2	uimsbf
alpha_q_scale_type[i]	1	bslbf
}		
}		
if (sprite_enable && vop_coding_type == "S") {		
if (no_sprite_points > 0)		
sprite_trajectory()		
if (sprite_brightness_change)		
vop_sprite_brightness_change	1	bslbf
if (vop_sprite_brightness_change)		
brightness_change_factor()		
if (sprite_defocusing)		
defocusing_control()		
if (sprite_lens_distortion)		
lens_distortion_parameter()		
next_start_code_studio()		
return()		
}		
composite_display_flag	1	bslbf

if (composite_display_flag) {		
v_axis	1	bslbf
field_sequence	3	uimsbf
sub_carrier	1	bslbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
while (nextbits() == '1') {		
extra_bit_picture /* with the value '1' */	1	uimsbf
extra_information_picture	8	uimsbf
}		
extra_bit_picture /* with the value '0' */	1	uimsbf
next_start_code_studio()		
extension_and_user_data(4)		
do {		
StudioSlice()		
} while (nextbits() == slice_start_code)		
}		

6.2.13.6 Studio sprite coding

sprite_trajectory() {	No. of bits	Mnemonic
for (i=0; i < no_of_sprite_warping_points; i++) {		
warping_mv_code(du[i])		
warping_mv_code(dv[i])		
}		
}		

warping_mv_code(d) {	No. of bits	Mnemonic
dmv_length	2-19	uimsbf
if (dmv_length != '00')		
dmv_code	1-21	uimsbf
marker_bit	1	bslbf
}		

brightness_change_factor() {	No. of bits	Mnemonic
brightness_change_factor_size	1-4	uimsbf
brightness_change_factor_code	5-10	uimsbf
}		

defocusing_control() {	No. of bits	Mnemonic
defocusing_control_parameter	9-12	uimsbf
marker_bit	1	bslbf
}		

lens_distortion_parameter() {	No. of bits	Mnemonic
lens_distortion_parameter_1	16	bslbf
marker_bit	1	bslbf
lens_distortion_parameter_2	16	bslbf
marker_bit	1	bslbf
lens_center_horizontal	14	bslbf
marker_bit	1	bslbf
lens_center_vertical	14	bslbf
marker_bit	1	bslbf
}		

6.2.13.7 Studio Slice

StudioSlice() {	No. of bits	Mnemonic
slice_start_code	32	uimsbf
macroblock_number	1-14	vlclbf
if (video_object_layer_shape != "binary only")		
quantiser_scale_code	5	uimsbf
if(video_object_layer_shape=="grayscale")		
for(i=0; i<aux_comp_count; i++)		
alpha_quantiser_scale_code[i]	5	uimsbf
if (nextbits() == 1) {		
slice_extension_flag	1	bslbf
intra_slice	1	uimsbf
slice_VOP_id_enable	1	uimsbf
slice_VOP_id	6	uimsbf
while (nextbits() == 1) {		
extra_bit_slice	1	uimsbf
extra_information_slice	8	uimsbf
}		
}		
extra_bit_slice	1	uimsbf
do {		
StudioMacroblock()		
} while (next_bits() != '000 0000 0000 0000 0000 0000')		
next_start_code_studio()		
}		

6.2.13.8 Studio Macroblock

StudioMacroblock() {	No. of bits	Mnemonic
if (vop_coding_type != "B") {		
if (video_object_layer_shape != "rectangular")		
Studio_mb_binary_shape_coding ()		
if (video_object_layer_shape != "binary only") {		
if (!transparent_mb()) {		
if (vop_coding_type != "I")		
not_coded	1	bslbf
if (vop_coding_type == "I" !not_coded) {		
compression_mode	1	bslbf
if (compression_mode == "DCT") { /* DCT		
*/		
StudioMacroblock_modes()		
if (macroblock_quant)		
quantiser_scale_code	5	uimsbf
if (macroblock_motion_forward)		
motion_vectors(0)		
if (macroblock_pattern)		
coded_block_pattern()		
for (i = 0; i < block_count; i++)		
if(!transparent_block(i))		
StudioBlock(i)		
} /* -end- DCT */		
else { /* DPCM */		
dpcm_scan_order	1	bslbf
StudioDPCMBlock() // Y or G		
StudioDPCMBlock() // U or B		
StudioDPCMBlock() // V or R		
} /* -end- DPCM */		
}		
}		
}		
}		
if(video_object_layer_shape=="grayscale"		
&& !transparent_mb()) {		
for (j=0 ; j<aux_comp_count ; j++) {		
if(macroblock_intra		
compression_mode == "DPCM") {		
coda_i[j]	1	bslbf
if(coda_i[j]=="coded") {		
alpha_compression_mode[j]	1	bslbf
if(alpha_compression_mode[j]=="DCT") {		
alpha_macroblock_quant[j]	1	bslbf
if (alpha_macroblock_quant[j])		
alpha_quantiser_scale_code[j]	5	uimsbf
for(i=0;i<alpha_block_count;i++)		
if(!transparent_block(i))		
StudioAlphaBlock(j,i)		

} /* -end- DCT */		
else { /* DPCM */		
alpha_dpcm_scan_order[j]	1	bslbf
StudioDPCMBlock() // alpha block		
} /* -end- DPCM */		
}		
} else { /* inter macroblock */		
coda_pb[j]	1-2	vlcblf
if(coda_pb[j]=="coded"){		
alpha_compression_mode[j]	1	bslbf
if(alpha_compression_mode[j]== "DCT")		
{		
cbpa[j]	1-6	vlcblf
if (cbpa[j] != 'no DCT coeff') {		
alpha_macroblock_quant[j]	1	bslbf
if (alpha_macroblock_quant[j])		
alpha_quantiser_scale_code[j]	5	uimsbf
}		
for(i=0 ;i<alpha_block_count ;i++)		
if(!transparent_block(i))		
StudioAlphaBlock(j,i)		
} else {		
alpha_dpcm_scan_order[j]	1	bslbf
StudioDPCMBlock(j) // Alpha		
}		
}		
} /* -end- inter */		
} /* -roop- aux_comp_count */		
} /* -end- grayscale */		
}		
NOTE The value of block_count is 6 in the 4:2:0 format. The value of alpha_block_count is 4.		

6.2.13.8.1 Studio MB Binary Shape Coding

Studio_mb_binary_shape_coding () {	No. of bits	Mnemonic
bab_type	1-6	vlcblf
if (vop_coding_type == 'P'){		
if (bab_type == 1){		
mvs_x	1-18	vlcblf
mvs_y	1-18	vlcblf
}		
}		
if (bab_type == 4) {		
inferior_symbol_macroblock	1	bslbf
cbbp	3-7	vlcblf
for (i=0; i<4; i++){		
if (coded_binary_block_pattern[i]=='1'){		

inferior_symbol_block	1	bslbf
scan_direction	1	bslbf
backward_load_flag	1	bslbf
clp	3-24	vlclbf
for (j=0; j<8; j++){		
if (coded_line_pattern[j]=='1')		
lbp	1-23	vlclbf
}		
}		
}		
}		
}		

6.2.13.8.2 Studio Macroblock modes

StudioMacroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlclbf
if (macroblock_motion_forward) {		
if (vop_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((vop_structure == "Frame picture") &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra macoblock_pattern)){		
dct_type	1	uimsbf
}		
}		

6.2.13.8.3 Motion vectors

motion_vectors (s) {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if (mv_format == field)		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
} else {		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0, s)		
motion_vertical_field_select[1][s]	1	uimsbf
motion_vector(1, s)		
}		
}		

6.2.13.8.4 Motion vector

motion_vector (r, s) {	No. of bits	Mnemonic
motion_code[r][s][0]	1-11	vlclbf
if ((vop_fcode[s][0] != 1) && (motion_code[r][s][0] != 0))		
motion_residual[r][s][0]	1-8	uimsbf
motion_code[r][s][1]	1-11	vlclbf
if ((vop_fcode[s][1] != 1) && (motion_code[r][s][1] != 0))		
motion_residual[r][s][1]	1-8	uimsbf
}		

6.2.13.8.5 Coded block pattern

coded_block_pattern () {	No. of bits	Mnemonic
coded_block_pattern_420	3-9	vlclbf
if (chroma_format == 4:2:2)		
coded_block_pattern_1	2	uimsbf
if (chroma_format == 4:4:4)		
coded_block_pattern_2	6	uimsbf
}		

6.2.13.9 Studio Block

The detailed syntax for the term “DCT coefficient” is fully described in clause 7.

StudioBlock(i) {	No. of bits	Mnemonic
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	2-14	vlclbf
if(dct_dc_size_luminance != 0)		
dct_dc_differential	1-15	vlclbf
if (dct_dc_size_luminance > 8)		
marker_bit	1	bslbf
} else {		
dct_dc_size_chrominance	2-15	vlclbf
if(dct_dc_size_chrominance !=0)		
dct_dc_differential	1-15	vlclbf
if (dct_dc_size_chrominance > 8)		
marker_bit	1	bslbf
}		
} else {		
First DCT coefficient	2-24	vlcfbf
}		
while (next_bits() != End of block)		
Subsequent DCT coefficients	3-24	vlclbf
End of block	3-15	vlcfbf
}		
}		

6.2.13.9.1 Studio Alpha Block

The syntax for DCT coefficient decoding is the same as for block(i) in 6.2.13.9.

StudioAlphaBlock(j, i) {	No. of bits	Mnemonic
last = 0		
if (alpha_pattern_code[j][i]) {		
if (macroblock_intra compression_mode == "DPCM") {		
dct_dc_size_alpha	2-14	vlclbf
if(dct_dc_size_alpha != 0)		
dct_dc_differential	1-15	vlclbf
if (dct_dc_size_alpha > 8)		
marker_bit	1	bslbf
} else {		
First DCT coefficient	2-24	vlcfbf
}		
while (next_bits() != End of block)		
Subsequent DCT coefficients	3-24	vlclbf
End of block	3-15	vlcfbf
}		
}		

6.2.13.10 Studio DPCM Block

StudioDPCMBlock() {	No. of bits	Mnemonic
block_mean	8-12	uimsbf
rice_parameter	4	uimsbf
for(i=0 ;i<8 or 16 ; i++) {		
for(j=0 ;j<8 or 16;j++) {		
rice_prefix_code	1-12	vlclbf
if (rice_prefix_code == '0000 0000 0001')		
dpcm_residual	4-12	uimsbf
else		
rice_suffix_code	0-12	uimsbf
}		
}		
}		

"

27) Add the following text at the end of subclause 6.3.1:

"

Only when profile_and_level_indication indicates a studio profile, extension layers can be inserted in the bitstream. At each point where extensions are allowed in the bitstream any number of the extensions from the defined allowable set may be included. However each type of extension shall not occur more than once.

In the case that a decoder encounters an extension with an extension identification that is described as “reserved” in this specification the decoder shall discard all subsequent data until the next start code. This requirement allows future definition of compatible extensions to this specification.

Table AMD1-1 — extension_start_code_identifier codes.

extension_start_code_identifier	Name
0000	reserved
0001	reserved
0010	Sequence Display Extension ID
0011	Quant Matrix Extension ID
0100	Copyright Extension ID
0101	VLC code Extension ID
0110	reserved
0111	Picture Display Extension ID
1000 - 1010	reserved
1011	Camera Parameters Extension ID
1100	ITU-T extension ID
1101	reserved
...	...
1111	reserved

"

28) Add the following subclause 6.3.13 after subclause 6.3.12:

"

6.3.13 Studio Video Object

6.3.13.1 Studio Visual Object

visual_object_start_code: The visual_object_start_code is the bit string '000001B5' in hexadecimal. It identifies the beginning of a visual object header.

visual_object_verid: This is a 4-bit code which identifies the version number of the visual object. Its meaning is defined in Table AMD1-2.

Table AMD1-2 -- Meaning of visual_object_verid

visual_object_verid	Meaning
0000	reserved
0001	object type listed in Table 9-1
0010	object type listed in Table V2-39
0011	reserved
0100	object type listed in Table AMD1-48
0101 – 1111	reserved

visual_object_type: The visual_object_type is a 4-bit code given in Table AMD1-3 which identifies the type of the visual object.

Table AMD1-3 -- Meaning of visual object type

code	visual object type
0000	reserved
0001	video ID
0010	still texture ID
0011	mesh ID
0100	face ID
0101	reserved
:	:
:	:
1111	reserved

video_object_start_code: The video_object_start_code is a string of 32 bits. The first 27 bits are '0000 0000 0000 0000 0000 0001 000' in binary and the last 5-bits represent one of the values in the range of '00000' to '11111' in binary. The video_object_start_code identifies the beginning of a video object header.

6.3.13.2 Extension and user data Studio

6.3.13.2.1 Extension data

extension_start_code: The extension_start_code is the bit string '000001B8' in hexadecimal. It identifies the beginning of extensions.

6.3.13.2.2 User data Studio

user_data_start_code: The user_data_start_code is the bit string '000001B2' in hexadecimal. It identifies the beginning of user data. The user data continues until receipt of another start code.

user_data: This is an 8 bit integer, an arbitrary number of which may follow one another. User data is defined by users for their specific applications. In the series of consecutive user_data bytes there shall not be a string of 23 or more consecutive zero bits.

6.3.13.2.3 Sequence display extension

This specification does not define the display process. The information in this extension does not affect the decoding process and may be ignored by decoders that conform to this specification.

extension_start_code_identifier: This is a 4-bit integer which identifies the extension. See Table AMD1-1.

video_format: This is a three bit integer indicating the representation of the pictures before being coded in accordance with this specification. Its meaning is defined in Table AMD1-4. If the sequence_display_extension() is not present in the bitstream then the video format may be assumed to be "Unspecified video format".

Table AMD1-4 -- Meaning of video_format

video_format	Meaning
000	component
001	PAL
010	NTSC
011	SECAM
100	MAC
101	Unspecified video format
110	reserved
111	reserved

video_range: This one-bit flag indicates the black level and range of the luminance and chrominance signals. In the case that `sequence_display_extension()` is not present in the bitstream, `video_range` is assumed to have the value 0 (a range of Y from 16 to 235 for 8-bit video).

colour_description: A flag which if set to '1' indicates the presence of `colour primaries`, `transfer characteristics` and `matrix coefficients` in the bitstream.

colour primaries: This 8-bit integer describes the chromaticity coordinates of the source primaries, and is defined in Table AMD1-5.

Table AMD1-5 -- Colour Primaries

Value	Primaries
0	(forbidden)
1	Recommendation ITU-R BT.709 primary x y green 0,300 0,600 blue 0,150 0,060 red 0,640 0,330 white D65 0,3127 0,3290
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	Recommendation ITU-R BT.470-2 System M primary x y green 0,21 0,71 blue 0,14 0,08 red 0,67 0,33 white C 0,310 0,316
5	Recommendation ITU-R BT.470-2 System B, G primary x y green 0,29 0,60 blue 0,15 0,06 red 0,64 0,33 white D65 0,313 0,329
6	SMPTE 170M primary x y green 0,310 0,595 blue 0,155 0,070 red 0,630 0,340 white D65 0,3127 0,3290
7	SMPTE 240M (1987) primary x y green 0,310 0,595 blue 0,155 0,070 red 0,630 0,340 white D65 0,3127 0,3291
8	Generic film (colour filters using Illuminant C) primary x y green 0,243 0,692 (Wratten 58) blue 0,145 0,049 (Wratten 47) red 0,681 0,319 (Wratten 25)
9-255	reserved

In the case that `sequence_display_extension()` is not present in the bitstream or `colour_description` is zero the chromaticity is assumed to be that corresponding to colour_primaries having the value 1.

transfer_characteristics: This 8-bit integer describes the opto-electronic transfer characteristic of the source picture, and is defined in Table AMD1-6.

Table AMD1-6 -- Transfer Characteristics

Value	Transfer Characteristic
0	(forbidden)
1	Recommendation ITU-R BT.709 $V = 1,099 L_C^{0,45} - 0,099$ for $1 \geq L_C \geq 0,018$ $V = 4,500 L_C$ for $0,018 > L_C \geq 0$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	Recommendation ITU-R BT.470-2 System M Assumed display gamma 2,2
5	Recommendation ITU-R BT.470-2 System B, G Assumed display gamma 2,8
6	SMPTE 170M $V = 1,099 L_C^{0,45} - 0,099$ for $1 \geq L_C \geq 0,018$ $V = 4,500 L_C$ for $0,018 > L_C \geq 0$
7	SMPTE 240M (1987) $V = 1,1115 L_C^{0,45} - 0,1115$ for $L_C \geq 0,0228$ $V = 4,0 L_C$ for $0,0228 > L_C$
8	Linear transfer characteristics i.e. $V = L_C$
9	Logarithmic transfer characteristic (100:1 range) $V = 1,0 - \log_{10}(L_C)/2$ for $1 = L_C = 0,01$ $V = 0,0$ for $0,01 > L_C$
10	Logarithmic transfer characteristic (316.22777:1 range) $V = 1,0 - \log_{10}(L_C)/2,5$ for $1 = L_C = 0,0031622777$ $V = 0,0$ for $0,0031622777 > L_C$
11-255	reserved

In the case that `sequence_display_extension()` is not present in the bitstream or `colour_description` is zero the transfer characteristics are assumed to be those corresponding to transfer_characteristics having the value 1.

matrix_coefficients: This 8-bit integer describes the matrix coefficients used in deriving luminance and chrominance signals from the green, blue, and red primaries, and is defined in Table AMD1-7.

In this table:

$E'Y$ is analogue with values between 0 and 1

$E'PB$ and $E'PR$ are analogue between the values -0,5 and 0,5

$E'R$, $E'G$ and $E'B$ are analogue with values between 0 and 1

White is defined as $E'Y=1$, $E'PB=0$, $E'PR=0$; $E'R=E'G=E'B=1$.

Y , Cb and Cr are related to $E'Y$, $E'PB$ and $E'PR$ by the following formulae:

if **video_range=0**:

$$Y = (219 * 2^{n-8} * E'Y) + 2^{n-4}.$$

$$Cb = (224 * 2^{n-8} * E'PB) + 2^{n-1}$$

$$Cr = (224 * 2^{n-8} * E'PR) + 2^{n-1}$$

if **video_range=1**:

$$Y = ((2^n - 1) * E'Y)$$

$$Cb = ((2^n - 1) * E'PB) + 2^{n-1}$$

$$Cr = ((2^n - 1) * E'PR) + 2^{n-1}$$

for n bit video.

For example, for 8 bit video,

video_range=0 gives a range of Y from 16 to 235, Cb and Cr from 16 to 240;

video_range=1 gives a range of Y from 0 to 255, Cb and Cr from 0 to 255.

NOTE - The decoding process given by this specification limits output sample values for Y , Cr and Cb to the range [0:255]. Thus sample values outside the range implied by the above equations may occasionally occur at the output of the decoding process. In particular the sample values 0 and 255 may occur.

Table AMD1-7 -- Matrix Coefficients

Value	Matrix
0	(forbidden)
1	Recommendation ITU-R BT.709 $E'_Y = 0,7154 E'_G + 0,0721 E'_B + 0,2125 E'_R$ $E'_{PB} = -0,386 E'_G + 0,500 E'_B - 0,115 E'_R$ $E'_{PR} = -0,454 E'_G - 0,046 E'_B + 0,500 E'_R$
2	Unspecified Video Image characteristics are unknown.
3	reserved
4	FCC $E'_Y = 0,59 E'_G + 0,11 E'_B + 0,30 E'_R$ $E'_{PB} = -0,331 E'_G + 0,500 E'_B - 0,169 E'_R$ $E'_{PR} = -0,421 E'_G - 0,079 E'_B + 0,500 E'_R$
5	Recommendation ITU-R BT.470-2 System B, G $E'_Y = 0,587 E'_G + 0,114 E'_B + 0,299 E'_R$ $E'_{PB} = -0,331 E'_G + 0,500 E'_B - 0,169 E'_R$ $E'_{PR} = -0,419 E'_G - 0,081 E'_B + 0,500 E'_R$
6	SMPTE 170M $E'_Y = 0,587 E'_G + 0,114 E'_B + 0,299 E'_R$ $E'_{PB} = -0,331 E'_G + 0,500 E'_B - 0,169 E'_R$ $E'_{PR} = -0,419 E'_G - 0,081 E'_B + 0,500 E'_R$
7	SMPTE 240M (1987) $E'_Y = 0,701 E'_G + 0,087 E'_B + 0,212 E'_R$ $E'_{PB} = -0,384 E'_G + 0,500 E'_B - 0,116 E'_R$ $E'_{PR} = -0,445 E'_G - 0,055 E'_B + 0,500 E'_R$
8-255	reserved

In the case that `sequence_display_extension()` is not present in the bitstream or `colour_description` is zero the matrix coefficients are assumed to be those corresponding to matrix_coefficients having the value 1.

display_horizontal_size: See `display_vertical_size`.

display_vertical_size: `display_horizontal_size` and `display_vertical_size` together define a rectangle which may be considered as the “intended display’s” active region. If this rectangle is smaller than the encoded frame size then the display process may be expected to display only a portion of the encoded frame. Conversely if the display rectangle is larger than the encoded frame size then the display process may be expected to display the reconstructed frames on a portion of the display device rather than on the whole display device.

`display_horizontal_size` shall be in the same units as `horizontal_size` (samples of the encoded frames).

`display_vertical_size` shall be in the same units as `vertical_size` (lines of the encoded frames).

`display_horizontal_size` and `display_vertical_size` do not affect the decoding process but may be used by the display process that is not standardised in this specification.