BS ISO/IEC 14496-29:2015

BSI Standards Publication

Information technology — Coding of audio-visual objects

Part 29: Web video coding

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO/IEC 14496-29:2015

National foreword

This British Standard is the UK implementation of ISO/IEC 14496-29:2015.

The UK participation in its preparation was entrusted to Technical Committee IST/37, Coding of picture, audio, multimedia and hypermedia information.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 79219 9
ICS 35.040

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 May 2015.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 14496-29:2015

INTERNATIONAL STANDARD

ISO/IEC 14496-29

First edition
2015-04-01

Information technology — Coding of audio-visual objects —

Part 29:
Web video coding

Technologies de l'information — Codage des objets audiovisuels —

Partie 29: Codage vidéo Web

Reference number
ISO/IEC 14496-29:2015(E)

Documents
Document Sharing Hub
© ISO/IEC 2015

Full document details: BS ISO/IEC 14496-29:2015

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested 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.org
Web www.iso.org

Published in Switzerland

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 14496-29:2015

Contents

Page

1 Scope **1**
2 Normative references **1**
3 Definitions **1**
4 Abbreviations **7**
5 Conventions **8**
5.1 Arithmetic operators 8
5.2 Logical operators 8
5.3 Relational operators 8
5.4 Bit-wise operators 9
5.5 Assignment operators 9
5.6 Range notation 9
5.7 Mathematical functions 9
5.8 Order of operation precedence 10
5.9 Variables, syntax elements, and tables 11
5.10 Text description of logical operations 12
5.11 Processes 13
6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships **13**
6.1 Bitstream formats 13
6.2 Source, decoded, and output picture formats 14
6.3 Spatial subdivision of pictures and slices 15
6.4 Inverse scanning processes and derivation processes for neighbours 16
7 Syntax and semantics **26**
7.1 Normative Syntax and Semantics 26
7.2 Specification of syntax functions, categories, and descriptors 28
7.3 Syntax in tabular form 30
7.4 Semantics 42
8 Decoding process **70**
8.1 NAL unit decoding process 71
8.2 Slice decoding process 72
8.3 Intra prediction process 82
8.4 Inter prediction process 95
8.5 Transform coefficient decoding process and picture construction process prior to deblocking filter process 107
8.6 (void) 118
8.7 Deblocking filter process 118
9 Parsing process **126**
9.1 Parsing process for Exp-Golomb codes 127
9.2 CAVLC parsing process for transform coefficient levels 131
Annex A (normative) **Profiles and levels** **142**
A.1 Requirements on video decoder capability 142
A.2 Profiles 142
A.3 Levels 143
Annex B (normative) **Byte stream format** **155**
B.1 Byte stream NAL unit syntax and semantics 155
B.2 Byte stream NAL unit decoding process 156
B.3 Decoder byte-alignment recovery (informative) 156

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

Annex C (normative) **Hypothetical reference decoder** **158**
C.1 Operation of coded picture buffer (CPB) 161
C.2 Operation of the decoded picture buffer (DPB) 163
C.3 Bitstream conformance 165
C.4 Decoder conformance 166
Annex D (normative) **Supplemental enhancement information** **170**
Annex E (normative) **Video usability information** **171**
E.1 VUI syntax 172
E.2 VUI semantics 173

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects*

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description*

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J Graphics Framework eXtensions (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

— *Part 24: Audio and systems interaction*

— *Part 25: 3D Graphics Compression Model*

— *Part 26: Audio conformance*

— *Part 27: 3D Graphics conformance*

— *Part 28: Composite font representation*

— *Part 29: Web video coding*

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

Introduction

This International Standard specifies Web Video Coding, a technology that is compatible with the Constrained Baseline Profile of ISO/IEC 14996-10. Only the subset that is specified in Annex A for the Constrained Baseline Profile is a normative specification, while all remaining aspects are informative. This text is derived from ISO/IEC 14996-10, with which the section numbers in this specification are aligned, and that specification may additionally be consulted if desired, as an aid to understanding this Specification.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

INTERNATIONAL STANDARD ISO/IEC 14496-29:2015(E)

Information technology — Coding of audio-visual objects — Part 29: Web video coding

1 Scope

This Part of ISO/IEC 14496 specifies Web Video Coding for coding of audio-visual objects.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

– ISO 11664-1, *Colorimetry — Part 1: CIE standard colorimetric observers*.

– ISO/IEC 14496-10: *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*

3 Definitions

For the purposes of this document, the following definitions apply:

3.1 **access unit**: A set of *NAL units* that are consecutive in *decoding order* and contain exactly one *primary coded picture*. In addition to the *primary coded picture*, an access unit may also contain one *auxiliary coded picture*, or other *NAL units* not containing *slices* of a *coded picture*. The decoding of an access unit always results in a *decoded picture*.

3.2 **AC transform coefficient**: Any *transform coefficient* for which the *frequency index* in one or both dimensions is non-zero.

3.3 **bitstream**: A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.

3.4 **block**: An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.

3.5 **[void]**

3.6 **broken link**: A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.

3.7 **byte**: A sequence of 8 bits, written and read with the most significant bit on the left and the least significant bit on the right. When represented in a sequence of data bits, the most significant bit of a byte is first.

3.8 **byte-aligned**: A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*. A bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 14496-29:2015

3.9 **byte stream**: An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.

3.10 **can**: A term used to refer to behaviour that is allowed, but not necessarily required.

3.11 **[void]**

3.12 **chroma**: An adjective specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours. The symbols used for a chroma array or sample are Cb and Cr.

NOTE – The term chroma is used rather than the term chrominance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term chrominance.

3.13 **coded frame**: A *coded representation* of a *frame*.

3.14 **coded picture**: A *coded representation* of a *picture*.

3.15 **coded picture buffer (CPB)**: A first-in first-out buffer containing *access units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C.

3.16 **coded representation**: A data element as represented in its coded form.

3.17 **[void]**

3.18 **coded slice NAL unit**: A *NAL unit* containing a *slice* that is not a *slice* of an *auxiliary coded picture*.

3.19 **coded video sequence**: A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *accessunits* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.

3.20 **component**: An array or single sample from one of the three arrays (*luma* and two *chroma*) that make up a *frame* in 4:2:0 colour format.

3.21 **DC transform coefficient**: A *transform coefficient* for which the *frequency index* is zero in all dimensions.

3.22 **decoded picture**: A *decoded picture* is derived by decoding a *coded picture*. A *decoded picture* is a decoded *frame*.

3.23 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.

3.24 **decoder**: An embodiment of a *decoding process*.

3.25 **decoder under test (DUT)**: A *decoder* that is tested for conformance to this International Standard by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing of the output of the two *decoders*.

3.26 **decoding order**: The order in which *syntax elements* are processed by the *decoding process*.

3.27 **decoding process**: The process specified in this International Standard that reads a *bitstream* and derives *decoded pictures* from it.

3.28 **[void]**

3.29 **display process**: A process not specified in this International Standard having, as its input, the cropped decoded *pictures* that are the output of the *decoding process*.

3.30 **emulation prevention byte**: A *byte* equal to 0x03 that may be present within a *NAL unit*. The presence of emulation prevention bytes ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* contains a *start code prefix*.

3.31 **encoder**: An embodiment of an *encoding process*.

3.32 **encoding process**: A process, not specified in this International Standard, that produces a *bitstream* conforming to this International Standard.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

3.33 **flag**: A variable that can take one of the two possible values 0 and 1.

3.34 **frame**: A *frame* contains an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0 format.

3.35 **frame macroblock**: A *macroblock* representing samples of a *coded frame.* All *macroblocks* of a *coded frame* are **frame macroblock**s.

3.36 **[void]**

3.37 ***frequency index:*** *A one-dimensional or two-dimensional* index associated wit*h a transform coefficient* prior to an *inverse transform* part of the *decoding process.*

3.38 **hypothetical reference decoder (HRD)**: A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.

3.39 **hypothetical stream scheduler (HSS)**: A hypothetical delivery mechanism for the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*. The HSS is used for checking the conformance of a *bitstream* or a *decoder*.

3.40 **I slice**: A *slice* that is decoded using *intra prediction* only.

3.41 **informative**: A term used to refer to content provided in this International Standard that is not an integral part of this International Standard. Informative content does not establish any mandatory requirements for conformance to this International Standard.

3.42 **instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *primary coded picture* is an *IDR picture*.

3.43 **instantaneous decoding refresh (IDR) picture**: A *coded picture* for which the variable IdrPicFlag is equal to 1. An IDR picture causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after the decoding of the IDR picture. All *coded pictures* that follow an IDR picture in *decoding order* can be decoded without *inter prediction* from any *picture* that precedes the IDR picture in *decoding order*. The first *picture* of each *coded video sequence* in *decoding order* is an IDR picture.

3.44 **inter coding**: Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *inter prediction*.

3.45 **inter prediction**: A *prediction* derived from decoded samples of *reference pictures* other than the current *decoded picture*.

3.46 **interpretation sample value**: A possibly-altered value corresponding to a decoded sample value of an *auxiliary coded picture* that may be generated for use in the *display process*. Interpretation sample values are not used in the *decoding process* and have no normative effect on the *decoding process*.

3.47 **intra coding**: Coding of a *block, macroblock*, *slice*, or *picture* that uses *intra prediction.*

3.48 **intra prediction**: A *prediction* derived from the decoded samples of the same decoded *slice*.

3.49 **intra slice**: See *I slice*.

3.50 **inverse transform**: A part of the *decoding process* by which a set of *transform coefficients* are converted into spatial-domain values, or by which a set of *transform coefficients* are converted into *DC transform coefficients*.

3.51 **layer**: One of a set of syntactical structures in a non-branching hierarchical relationship. Higher layers contain lower layers. The coding layers are the *coded video sequence*, *picture*, *slice*, and *macroblock* layers.

3.52 **level**: A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this International Standard. The same set of levels is defined for all *profiles*, with most aspects of the definition of each level being in common across different *profiles.* Individual implementations may, within specified constraints, support a different level for each supported *profile*. In a different context, a level is the value of a *transform coefficient* prior to *scaling* (see the definition of *transform coefficient level*).

3.53 **list**: A one-dimensional array of *syntax elements* or variables.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

3.54 **luma**: An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colours. The symbol or subscript used for luma is Y or L.

NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.

3.55 **macroblock**: A 16x16 *block* of *luma* samples and two corresponding *blocks* of *chroma* samples of a *picture* that has three sample arrays, or a 16x16 *block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes. The division of a *slice* into macroblocks is a *partitioning*.

3.56 **macroblock address**: a macroblock address is the index of a *macroblock* in a *macroblock raster scan* of the *picture* starting with zero for the top-left *macroblock* in a *picture*.

3.57 **macroblock location**: The two-dimensional coordinates of a *macroblock* in a *picture* denoted by (x, y). For the top left *macroblock* of the *picture* (x, y) is equal to (0, 0). x is incremented by 1 for each *macroblock* column from left to right. y is incremented by 1 for each *macroblock* row from top to bottom.

3.58 **macroblock partition**: A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *macroblock* for *inter prediction* for a *picture* that has three sample arrays or a *block* of *luma* samples resulting from a *partitioning* of a *macroblock* for *inter prediction* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.

3.59 **matrix**: A two-dimensional array of *syntax elements* or variables.

3.60 **may**: A term used to refer to behaviour that is allowed, but not necessarily required. In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.

3.61 **memory management control operation**: Seven operations that control *reference picture marking*.

3.62 **motion vector**: A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.

3.63 **must**: A term used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this International Standard. This term is used exclusively in an *informative* context.

3.64 **NAL unit**: A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

3.65 **NAL unit stream**: A sequence of *NAL units*.

3.66 **non-reference frame**: A *frame* coded with nal_ref_idc equal to 0.

3.67 **non-reference picture**: A *picture* coded with nal_ref_idc equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.

3.68 **note**: A term used to prefix *informative* remarks. This term is used exclusively in an *informative* context.

3.69 **output order**: The order in which the *decoded pictures* are output from the *decoded picture buffer*.

3.70 **P slice**: A *slice* that may be decoded using *intraprediction* or *inter prediction* using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.

3.71 **parameter**: A *syntax element* of a *sequence parameter set* or a *picture parameter set*. Parameter is also used as part of the defined term *quantisation parameter*.

3.72 **partitioning**: The division of a set into subsets such that each element of the set is in exactly one of the subsets.

3.73 **picture**: A collective term for a *frame*.

3.74 **picture parameter set**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by the pic_parameter_set_id *syntax element* found in each *slice header*.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

3.75 **picture order count**: A variable that is associated with each *coded picture* and has a value that is non-decreasing with increasing *picture* position in *output order* relative to the first output *picture* of the previous *IDR picture* in *decoding order* or relative to the previous *picture*, in *decoding order*, that contains a *memory management control operation* that marks all *reference pictures* as "unused for reference".

3.76 **prediction**: An embodiment of the *prediction process*.

3.77 **prediction process**: The use of a *predictor* to provide an estimate of the sample value or data element currently being decoded.

3.78 **predictive slice**: See *P slice*.

3.79 **predictor**: A combination of specified values or previously decoded sample values or data elements used in the *decoding process* of subsequent sample values or data elements.

3.80 **primary coded picture**: The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this International Standard. The primary coded picture contains all *macroblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures. *e*.

3.81 **profile**: A specified subset of the syntax of this International Standard.

3.82 **quantisation parameter**: A variable used by the *decoding process* for *scaling* of *transform coefficient levels*.

3.83 **random access**: The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.

3.84 **raster scan**: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.

3.85 **raw byte sequence payload (RBSP):** A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit*. An RBSP is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and followed by zero or more subsequent bits equal to 0.

3.86 **raw byte sequence payload (RBSP) stop bit:** A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*. The location of the end of the *string of data bits* within an *RBSP* can be identified by searching from the end of the *RBSP* for the *RBSP stop bit*, which is the last non-zero bit in the *RBSP*.

3.87 **recovery point**: A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.

3.88 **reference frame**: A *reference frame* may be used for *inter prediction* when *P slices* of a *coded frame* are decoded. See also *reference picture*.

3.89 **reference index**: An index into a *reference picture list*.

3.90 **reference picture**: A *picture* with nal_ref_idc not equal to 0. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.

3.91 **reference picture list**: A list of *reference pictures* that is used for *inter prediction* of a *P slice*. *F*or the *decoding process* of a *P slice*, there is one reference picture list.

3.92 **reference picture list 0**: A *reference picture list* used for *inter prediction* of a *Pslice*. All *inter prediction* used for *P slices* uses reference picture list 0.

3.93 **reference picture marking**: Specifies, in the bitstream, how the *decoded pictures* are marked for *inter prediction*.

3.94 **reserved**: The term reserved, when used in the clauses specifying some values of a particular *syntax element*, are for future use by ITU-T | ISO/IEC. These values shall not be used in *bitstreams* conforming to this International Standard, but may be used in future extensions of this International Standard by ITU-T | ISO/IEC.

3.95 **residual**: The decoded difference between a *prediction* of a sample or data element and its decoded value.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

3.96 **run**: A number of consecutive data elements represented in the decoding process. In one context, the number of zero-valued *transform coefficient levels* preceding a non-zero *transform coefficient level* in the list of *transform coefficient levels* generated by a *zig-zag scan*. In other contexts, run refers to a number of *macroblocks*.

3.97 **sample aspect ratio**: Specifies, for assisting the *display process*, which is not specified in this International Standard, the ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *frame*. Sample aspect ratio is expressed as *h*:*v*, where *h* is horizontal width and *v* is vertical height (in arbitrary units of spatial distance).

3.98 **scaling**: The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.

3.99 **sequence parameter set**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded video sequences* as determined by the content of a seq_parameter_set_id *syntax element* found in the *picture parameter set* referred to by the pic_parameter_set_id *syntax element* found in each *slice header*.

3.100 **shall**: A term used to express mandatory requirements for conformance to this International Standard. When used to express a mandatory constraint on the values of *syntax elements* or on the results obtained by operation of the specified *decoding process*, it is the responsibility of the *encoder* to ensure that the constraint is fulfilled. When used in reference to operations performed by the *decoding process*, any *decoding process* that produces identical results to the *decoding process* described herein conforms to the *decoding process* requirements of this International Standard.

3.101 **should**: A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this International Standard.

3.102 **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

3.103 **slice**: An integer number of *macroblocks* ordered consecutively in the *raster scan* within the *primary coded picture*. The *macroblock addresses* are derived from the first *macroblock address* in a slice (as represented in the *slice header*) and, when a *picture* is coded using three separate colour planes, a colour plane identifier.

3.104 **[void]**

3.105 **[void]**

3.106 **slice header**: A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.

3.107 **source**: Term used to describe the video material or some of its attributes before encoding.

3.108 **start code prefix**: A unique sequence of three *bytes* equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*. The location of a start code prefix can be used by a *decoder* to identify the beginning of a new *NAL unit* and the end of a previous *NAL unit*. Emulation of start code prefixes is prevented within *NAL units* by the inclusion of *emulation prevention bytes*.

3.109 **string of data bits (SODB)**: A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*. Within an SODB, the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.

3.110 **sub-macroblock**: One quarter of the samples of a *macroblock*, i.e., an 8x8 *luma block* and two corresponding *chroma blocks* of which one corner is located at a corner of the *macroblock* for a *picture* that has three sample arrays or an 8x8 *luma block* of which one corner is located at a corner of the *macroblock* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.

3.111 **sub-macroblock partition**: A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *sub-macroblock* for *inter prediction* for a *picture* that has three sample arrays or a *block* of *luma* samples resulting from a *partitioning* of a *sub-macroblock* for *inter prediction* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

3.112 **syntax element**: An element of data represented in the *bitstream*.

3.113 **syntax structure**: Zero or more *syntax elements* present together in the *bitstream* in a specified order.

3.114 **transform coefficient**: A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in an *inverse transform* part of the *decoding process*.

3.115 **transform coefficient level**: An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.

3.116 **universal unique identifier (UUID)**: An identifier that is unique with respect to the space of all universal unique identifiers.

3.117 **unspecified:** The term unspecified, when used in the clauses specifying some values of a particular *syntax element*, indicates that the values have no specified meaning in this International Standard and will not have a specified meaning in the future as an integral part of this International Standard.

3.118 **variable length coding (VLC)**: A reversible procedure for entropy coding that assigns shorter bit strings to *symbols* expected to be more frequent and longer bit strings to *symbols* expected to be less frequent.

3.119 **VCL NAL unit**: A collective term for *coded slice NAL units*.

3.120 **zig-zag scan**: A specific sequential ordering of *transform coefficient levels* from (approximately) the lowest spatial frequency to the highest. Zig-zag scan is used for *transform coefficient levels* in *frame macroblocks.*

4 Abbreviations

For the purposes of this International Standard, the following abbreviations apply:

CAVLC Context-based Adaptive Variable Length Coding

CBR Constant Bit Rate

CPB Coded Picture Buffer

DPB Decoded Picture Buffer

DUT Decoder under test

FIFO First-In, First-Out

HRD Hypothetical Reference Decoder

HSS Hypothetical Stream Scheduler

IDR Instantaneous Decoding Refresh

LSB Least Significant Bit

MB Macroblock

MSB Most Significant Bit

NAL Network Abstraction Layer

RBSP Raw Byte Sequence Payload

SEI Supplemental Enhancement Information

SODB String Of Data Bits

UUID Universal Unique Identifier

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

VBR Variable Bit Rate

VCL Video Coding Layer

VLC Variable Length Coding

VUI Video Usability Information

5 Conventions

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, integer division and arithmetic shift operations are specifically defined. Numbering and counting conventions generally begin from 0.

5.1 Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, 7/4 and −7/−4 are truncated to 1 and −7/4 and 7/−4 are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^{y} f(i)$	The summation of f(i) with i taking all integer values from x up to and including y.
x % y	Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

5.2 Logical operators

The following logical operators are defined as follows:

x&&y	Boolean logical "and" of x and y.
x || y	Boolean logical "or" of x and y.
!	Boolean logical "not".
x ? y : z	If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

5.3 Relational operators

The following relational operators are defined as follows:

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
==	Equal to.

© ISO/IEC 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

!= Not equal to.

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.4 Bit-wise operators

The following bit-wise operators are defined as follows:

& Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

| Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

5.5 Assignment operators

The following arithmetic operators are defined as follows:

= Assignment operator.

++ Increment, i.e., x++ is equivalent to x=x + 1; when used in an array index, evaluates to the value of the variable prior to the increment operation.

−− Decrement, i.e., x— is equivalent to x= x− 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation.

+= Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3).

−= Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3).

5.6 Range notation

The following notation is used to specify a range of values:

x = y..z x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers.

5.7 Mathematical functions

The following mathematical functions are defined as follows:

$BitDepth_Y$ and $BitDepth_C$ are both specified to be equal to 8 in this standard

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

$$\text{Abs}(\ x\ ) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases} \tag{5-1}$$

Ceil(x) the smallest integer greater than or equal to x. (5-2)

$$\text{Clip1}_Y(\ x\ ) = \text{Clip3}(\ 0,\ (\ 1 << \text{BitDepth}_Y\ ) - 1,\ x\ ) \tag{5-3}$$

$$\text{Clip1}_C(\ x\ ) = \text{Clip3}(\ 0,\ (\ 1 << \text{BitDepth}_C\ ) - 1,\ x\ ) \tag{5-4}$$

$$\text{Clip3}(\ x,\ y,\ z\ ) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \tag{5-5}$$

Floor(x) the greatest integer less than or equal to x. (5-6)

$$\text{InverseRasterScan}(\ a,\ b,\ c,\ d,\ e\ ) = \begin{cases} (a\%(d/b))*b & ; \quad e == 0 \\ (a/(d/b))*c & ; \quad e == 1 \end{cases} \tag{5-7}$$

Log2(x) returns the base-2 logarithm of x. (5-8)

Log10(x) returns the base-10 logarithm of x. (5-9)

$$\text{Median}(\ x,\ y,\ z\ ) = x + y + z - \text{Min}(\ x,\ \text{Min}(\ y,\ z\ )\ ) - \text{Max}(\ x,\ \text{Max}(\ y,\ z\ )\ ) \tag{5-10}$$

$$\text{Min}(\ x,\ y\ ) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases} \tag{5-11}$$

$$\text{Max}(\ x,\ y\ ) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases} \tag{5-12}$$

$$\text{Round}(\ x\ ) = \text{Sign}(\ x\ ) * \text{Floor}(\ \text{Abs}(\ x\ ) + 0.5\ ) \tag{5-13}$$

$$\text{Sign}(\ x\ ) = \begin{cases} 1 & ; \quad x >= 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{5-14}$$

$$\text{Sqrt}(\ x\ ) = \sqrt{x} \tag{5-15}$$

5.8 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– operations of a higher precedence are evaluated before any operation of a lower precedence,

– operations of the same precedence are evaluated sequentially from left to right.

Table 5-1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x– –"
"!x", "–x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", "$\frac{x}{y}$", "x % y"
"x + y", "x – y" (as a two-argument operator), "$\sum_{i=x}^{y} f(i)$"
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = = y", "x != y"
"x & y"
"x | y"
"x && y"
"x | | y"
"x ? y : z"
"x = y", "x += y", "x –= y"

5.9 Variables, syntax elements, and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the bitstream are referred to as syntax functions. These functions are specified in subclause 7.2 and assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream. Syntax functions are described by their names, which are constructed

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

as syntax element names and end with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in subclause 5.7) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing. Thus, an element of a matrix s at horizontal position x and vertical position y may be denoted either as s[x, y] or as s_{yx}.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.10 Text description of logical operations

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
  statement 0
else if ( condition 1 )
  statement 1
…
else /* informative remark on remaining condition */
  statement n
```

may be described in the following manner:

… as follows / … the following applies:

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- …
- Otherwise (informative remark on remaining condition), statement n

Each "If…Otherwise, if…Otherwise, …" statement in the text is introduced with "… as follows" or "… the following applies" immediately followed by "If … ". The last condition of the "If…Otherwise, if…Otherwise, …" is always an "Otherwise, …". Interleaved "If…Otherwise, if…Otherwise, …" statements can be identified by matching "… as follows" or "… the following applies" with the ending "Otherwise, …".

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0a  &&  condition 0b )
  statement 0
else if ( condition 1a  ||  condition 1b )
  statement 1
…
```


© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

```
else
  statement n
```

may be described in the following manner:

… as follows / … the following applies:

- If all of the following conditions are true, statement 0
  - condition 0a
  - condition 0b
- Otherwise, if any of the following conditions are true, statement 1
  - condition 1a
  - condition 1b
- …
- Otherwise, statement n

In the text, a statement of logical operations as would be described in pseudo-code as:

```
if( condition 0 )
  statement 0
if ( condition 1 )
  statement 1
```

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

5.11 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as the input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

In the specification of a process, a specific macroblock may be referred to by the variable name having a value equal to the address of the specific macroblock.

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

6.1 Bitstream formats

This subclause specifies the relationship between the NAL unit stream and byte stream, either of which are referred to as the bitstream.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this International Standard. The byte stream format is specified in Annex B.

6.2 Source, decoded, and output picture formats

This subclause specifies the relationship between source and decoded frames that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of frames (called pictures) in decoding order.

The source and decoded pictures are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome), with or without an auxiliary array.
- Luma and two Chroma (YCbCr or YCgCo), with or without an auxiliary array.
- Green, Blue and Red (GBR, also known as RGB), with or without an auxiliary array.
- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ), with or without an auxiliary array.

For convenience of notation and terminology in this Specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E.

Table 6-1 –Chroma Format

chroma_format_idc	Chroma Format
1	4:2:0

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

For the purposes of this version of this Specification, the value of chroma_format_idc shall be inferred to be equal to 1 and the chroma format shall be inferred to be 4:2:0 as shown in Table 6-1.

The width and height of the luma sample arrays are each an integer multiple of 16. In coded video sequences using 4:2:0 chroma sampling, the width and height of chroma sample arrays are each an integer multiple of 8. The width or height of pictures output from the decoding process need not be an integer multiple of 16 and can be specified using a cropping rectangle.

The syntax for the luma and (when present) chroma arrays are ordered such when data for all three colour components is present, the data for the luma array is first, followed by any data for the Cb array, followed by any data for the Cr array, unless otherwise specified.

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a coded video sequence is equal to 8, regardless of whether the sample is a sample of the luma array or a sample of the chroma arrays.

The nominal vertical and horizontal relative locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

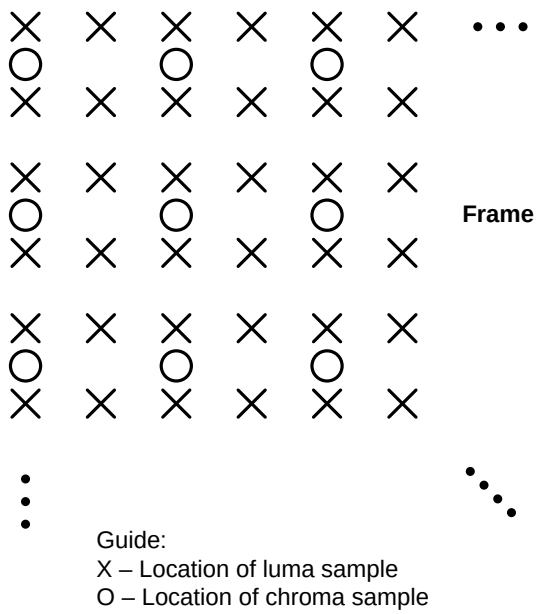

Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

The samples are processed in units of macroblocks. The luma array for each macroblock is 16 samples in both width and height. The variables MbWidthC and MbHeightC, which specify the width and height, respectively, of the chroma arrays for each macroblock, are derived as follows:

MbWidthC and MbHeightC are derived as

$$\text{MbWidthC} = 16 / 2 \tag{6-1}$$
$$\text{MbHeightC} = 16 / 2 \tag{6-2}$$

6.3 Spatial subdivision of pictures and slices

This subclause specifies how a picture is partitioned into slices and macroblocks. Pictures are divided into slices. A slice is a sequence of macroblocks. Each macroblock is comprised of one 16x16 luma array and two corresponding chroma sample arrays. Each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-2.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

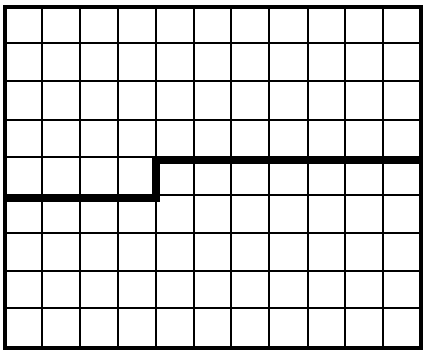

Figure 6-2 – A picture with 11 by 9 macroblocks that is partitioned into two slices

6.4 Inverse scanning processes and derivation processes for neighbours

This subclause specifies inverse scanning processes; i.e., the mapping of indices to locations, and derivation processes for neighbours.

6.4.1 Inverse macroblock scanning process

Input to this process is a macroblock address mbAddr.

Output of this process is the location (x, y) of the upper-left luma sample for the macroblock with address mbAddr relative to the upper-left sample of the picture.

The inverse macroblock scanning process is specified as follows:

$$x = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6\text{-}3)$$

$$y = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6\text{-}4)$$

6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process

Macroblocks or sub-macroblocks may be partitioned, and the partitions are scanned for inter prediction as shown in Figure 6-3. The outer rectangles refer to the samples in a macroblock or sub-macroblock, respectively. The rectangles refer to the partitions. The number in each rectangle specifies the index of the inverse macroblock partition scan or inverse sub-macroblock partition scan.

The functions MbPartWidth( ), MbPartHeight( ), SubMbPartWidth( ), and SubMbPartHeight( ) describing the width and height of macroblock partitions and sub-macroblock partitions are specified in Tables 7-9 and 7-12. MbPartWidth( ) and MbPartHeight( ) are set to appropriate values for each macroblock, depending on the macroblock type. SubMbPartWidth( ) and SubMbPartHeight( ) are set to appropriate values for each sub-macroblock of a macroblock with mb_type equal to P_8x8 or P_8x8ref0, depending on the sub-macroblock type.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

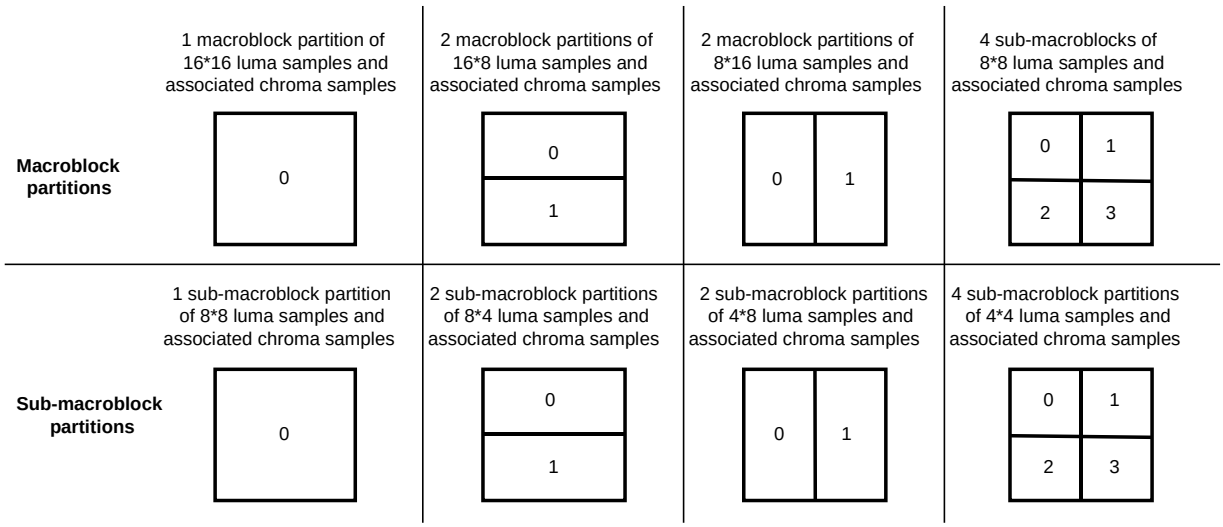

Figure 6-3 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans

6.4.2.1 Inverse macroblock partition scanning process

Input to this process is the index of a macroblock partition mbPartIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the macroblock partition mbPartIdx relative to the upper-left sample of the macroblock.

The inverse macroblock partition scanning process is specified by

x = InverseRasterScan(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 0) (6-5)

y = InverseRasterScan(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 1) (6-6)

6.4.2.2 Inverse sub-macroblock partition scanning process

Inputs to this process are the index of a macroblock partition mbPartIdx and the index of a sub-macroblock partition subMbPartIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the sub-macroblock partition subMbPartIdx relative to the upper-left sample of the sub-macroblock.

The inverse sub-macroblock partition scanning process is specified as follows:

– If mb_type is equal to P_8x8 or P_8x8ref0

x = InverseRasterScan(subMbPartIdx, SubMbPartWidth(sub_mb_type[mbPartIdx]),
SubMbPartHeight(sub_mb_type[mbPartIdx]), 8, 0) (6-7)

y = InverseRasterScan(subMbPartIdx, SubMbPartWidth(sub_mb_type[mbPartIdx]),
SubMbPartHeight(sub_mb_type[mbPartIdx]), 8, 1) (6-8)

– Otherwise (mb_type is not equal to P_8x8 or P_8x8ref0),

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

x = InverseRasterScan(subMbPartIdx, 4, 4, 8, 0) (6-9)

y = InverseRasterScan(subMbPartIdx, 4, 4, 8, 1) (6-10)

6.4.3 Inverse 4x4 luma block scanning process

Input to this process is the index of a 4x4 luma block luma4x4BlkIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the 4x4 luma block with index luma4x4BlkIdx relative to the upper-left luma sample of the macroblock.

Figure 6-4 shows the scan for the 4x4 luma blocks.

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 6-4 – Scan for 4x4 luma blocks

The inverse 4x4 luma block scanning process is specified by

x = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 0) +
InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 0) (6-11)

y = InverseRasterScan(luma4x4BlkIdx / 4, 8, 8, 16, 1) +
InverseRasterScan(luma4x4BlkIdx % 4, 4, 4, 8, 1) (6-12)

6.4.4 (void)

6.4.5 Inverse 8x8 luma block scanning process

Input to this process is the index of an 8x8 luma block luma8x8BlkIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the 8x8 luma block with index luma8x8BlkIdx relative to the upper-left luma sample of the macroblock.

Figure 6-5 shows the scan for the 8x8 luma blocks.

0	1
2	3

Figure 6-5 – Scan for 8x8 luma blocks

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

The inverse 8x8 luma block scanning process is specified by:

$$x = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 0) \quad (6\text{-}13)$$

$$y = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 1) \quad (6\text{-}14)$$

6.4.6 (void)

6.4.7 Inverse 4x4 chroma block scanning process

Input to this process is the index of a 4x4 chroma block chroma4x4BlkIdx.

Output of this process is the location (x, y) of the upper-left chroma sample for a 4x4 chroma block with index chroma4x4BlkIdx relative to the upper-left chroma sample of the macroblock.

The inverse 4x4chroma block scanning process is specified by

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (6\text{-}15)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (6\text{-}16)$$

6.4.8 Derivation process of the availability for macroblock addresses

Input to this process is a macroblock address mbAddr.

Output of this process is the availability of the macroblock mbAddr.

NOTE – The meaning of availability is determined when this process is invoked.

The macroblock is marked as available, unless any of the following conditions are true, in which case the macroblock is marked as not available:

– mbAddr < 0,

– mbAddr > CurrMbAddr,

– the macroblock with address mbAddr belongs to a different slice than the macroblock with address CurrMbAddr.

6.4.9 Derivation process for neighbouring macroblock addresses and their availability

The outputs of this process are:

– mbAddrA: the address and availability status of the macroblock to the left of the current macroblock,

– mbAddrB: the address and availability status of the macroblock above the current macroblock,

– mbAddrC: the address and availability status of the macroblock above-right of the current macroblock,

– mbAddrD: the address and availability status of the macroblock above-left of the current macroblock.

Figure 6-6 shows the relative spatial locations of the macroblocks with mbAddrA, mbAddrB, mbAddrC, and mbAddrD relative to the current macroblock with CurrMbAddr.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr	

Figure 6-6 – Neighbouring macroblocks for a given macroblock

Input to the process in subclause 6.4.8 is mbAddrA = CurrMbAddr − 1 and the output is whether the macroblock mbAddrA is available. In addition, mbAddrA is marked as not available when CurrMbAddr % PicWidthInMbs is equal to 0.

Input to the process in subclause 6.4.8 is mbAddrB = CurrMbAddr − PicWidthInMbs and the output is whether the macroblock mbAddrB is available.

Input to the process in subclause 6.4.8 is mbAddrC = CurrMbAddr − PicWidthInMbs + 1 and the output is whether the macroblock mbAddrC is available. In addition, mbAddrC is marked as not available when (CurrMbAddr + 1) % PicWidthInMbs is equal to 0.

Input to the process in subclause 6.4.8 is mbAddrD = CurrMbAddr − PicWidthInMbs − 1 and the output is whether the macroblock mbAddrD is available. In addition, mbAddrD is marked as not available when CurrMbAddr % PicWidthInMbs is equal to 0.

6.4.10 (void)

6.4.11 Derivation processes for neighbouring macroblocks, blocks, and partitions

Subclause 6.4.11.1 specifies the derivation process for neighbouring macroblocks.

Subclause 6.4.11.4 specifies the derivation process for neighbouring 4x4 luma blocks.

Subclause 6.4.11.5 specifies the derivation process for neighbouring 4x4 chroma blocks.

Subclause 6.4.11.7 specifies the derivation process for neighbouring partitions.

Table 6-2 specifies the values for the difference of luma location (xD, yD) for the input and the replacement for N in mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, luma4x4BlkIdxN, cb4x4BlkIdxN, cr4x4BlkIdxN, and chroma4x4BlkIdxN for the output. These input and output assignments are used in subclauses 6.4.11.1 to 6.4.11.7. The variable predPartWidth is specified when Table 6-2 is referred to.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

Table 6-2 – Specification of input and output assignments for subclauses 6.4.11.1 to 6.4.11.7

N	xD	yD
A	−1	0
B	0	−1
C	predPartWidth	−1
D	−1	−1

Figure 6-7 illustrates the relative location of the neighbouring macroblocks, blocks, or partitions A, B, C, and D to the current macroblock, partition, or block, when the current macroblock, partition, or block is in frame coding mode.

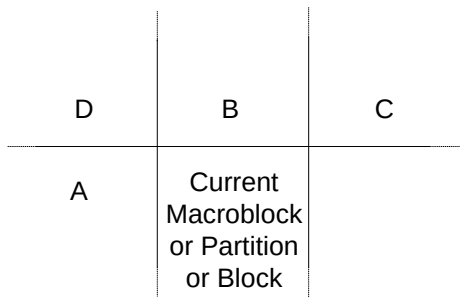

Figure 6-7 – Determination of the neighbouring macroblock, blocks, and partitions (informative)

6.4.11.1 Derivation process for neighbouring macroblocks

Outputs of this process are:

– mbAddrA: the address of the macroblock to the left of the current macroblock and its availability status,

– mbAddrB: the address of the macroblock above the current macroblock and its availability status.

mbAddrN (with N being A or B) is derived as specified by the following ordered steps:

1. The difference of luma location (xD, yD) is set according to Table 6-2.
2. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) equal to (xD, yD), and the output is assigned to mbAddrN.

6.4.11.2 (void)

6.4.11.3 (void)

6.4.11.4 Derivation process for neighbouring 4x4 luma blocks

Input to this process is a 4x4 luma block index luma4x4BlkIdx.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

Outputs of this process are:

– mbAddrA: either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock and its availability status,

– luma4x4BlkIdxA: the index of the 4x4 luma block to the left of the 4x4 block with index luma4x4BlkIdx and its availability status,

– mbAddrB: either equal to CurrMbAddr or the address of the macroblock above the current macroblock and its availability status,

– luma4x4BlkIdxB: the index of the 4x4 luma block above the 4x4 block with index luma4x4BlkIdx and its availability status.

mbAddrN and luma4x4BlkIdxN (with N being A or B) are derived as specified by the following ordered steps:

1. The difference of luma location (xD, yD) is set according to Table 6-2.
2. The inverse 4x4 luma block scanning process as specified in subclause 6.4.3 is invoked with luma4x4BlkIdx as the input and (x, y) as the output.
3. The luma location (xN, yN) is specified by

$$xN = x + xD \tag{6-17}$$

$$yN = y + yD \tag{6-18}$$

4. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
5. The variable luma4x4BlkIdxN is derived as follows:
   – If mbAddrN is not available, luma4x4BlkIdxN is marked as not available.
   – Otherwise (mbAddrN is available), the derivation process for 4x4 luma block indices as specified in subclause 6.4.13.1 is invoked with the luma location (xW, yW) as the input and the output is assigned to luma4x4BlkIdxN.

6.4.11.5 Derivation process for neighbouring 4x4 chroma blocks

Input to this process is a 4x4 chroma block index chroma4x4BlkIdx.

Outputs of this process are:

– mbAddrA (either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock) and its availability status,

– chroma4x4BlkIdxA (the index of the 4x4 chroma block to the left of the 4x4 chroma block with index chroma4x4BlkIdx) and its availability status,

– mbAddrB (either equal to CurrMbAddr or the address of the macroblock above the current macroblock) and its availability status,

– chroma4x4BlkIdxB (the index of the 4x4 chroma block above the 4x4 chroma block with index chroma4x4BlkIdx) and its availability status.

mbAddrN and chroma4x4BlkIdxN (with N being A or B) are derived as specified by the following ordered steps:

1. The difference of chroma location (xD, yD) is set according to Table 6-2.
2. The inverse 4x4 chroma block scanning process as specified in subclause 6.4.7 is invoked with chroma4x4BlkIdx as the input and(x, y) as the output.
3. The chroma location (xN, yN) is specified by

$$xN = x + xD \tag{6-19}$$

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

$$yN = y + yD \quad (6\text{-}20)$$

4. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for chroma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
5. The variable chroma4x4BlkIdxN is derived as follows:
   - If mbAddrN is not available, chroma4x4BlkIdxN is marked as not available.
   - Otherwise (mbAddrN is available), the derivation process for 4x4 chroma block indices as specified in subclause 6.4.13.2 is invoked with the chroma location (xW, yW) as the input and the output is assigned to chroma4x4BlkIdxN.

6.4.11.6 (void)

6.4.11.7 Derivation process for neighbouring partitions

Inputs to this process are:

- a macroblock partition index mbPartIdx
- a current sub-macroblock type currSubMbType
- a sub-macroblock partition index subMbPartIdx

Outputs of this process are:

- mbAddrA\mbPartIdxA\subMbPartIdxA: specifying the macroblock or sub-macroblock partition to the left of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,
- mbAddrB\mbPartIdxB\subMbPartIdxB: specifying the macroblock or sub-macroblock partition above the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,
- mbAddrC\mbPartIdxC\subMbPartIdxC: specifying the macroblock or sub-macroblock partition to the right-above of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,
- mbAddrD\mbPartIdxD\subMbPartIdxD: specifying the macroblock or sub-macroblock partition to the left-above of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status.

mbAddrN, mbPartIdxN, and subMbPartIdxN (with N being A, B, C, or D) are derived as specified by the following ordered steps:

1. The inverse macroblock partition scanning process as described in subclause 6.4.2.1 is invoked with mbPartIdx as the input and (x, y) as the output.
2. The location of the upper-left luma sample inside a macroblock partition (xS, yS) is derived as follows:
   - If mb_type is equal to P_8x8 or P_8x8ref0, the inverse sub-macroblock partition scanning process as described in subclause 6.4.2.2 is invoked with subMbPartIdx as the input and (xS, yS) as the output.
   - Otherwise, (xS, yS) are set to (0, 0).
3. The variable predPartWidth in Table 6-2 is specified as follows:
   - If mb_type is equal to P_Skip, predPartWidth = 16.
   - Otherwise, if mb_type is equal to P_8x8 or P_8x8ref0, predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx]).
   - Otherwise, predPartWidth = MbPartWidth(mb_type).

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

4. The difference of luma location (xD, yD) is set according to Table 6-2.
5. The neighbouring luma location (xN, yN) is specified by

$$xN = x + xS + xD \quad (6\text{-}21)$$

$$yN = y + yS + yD \quad (6\text{-}22)$$

6. The derivation process for neighbouring locations as specified in subclause 6.4.12 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
7. Depending on mbAddrN, the following applies:
   - If mbAddrN is not available, the macroblock or sub-macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN is marked as not available.
   - Otherwise (mbAddrN is available), the following ordered steps are specified:
     a. Let mbTypeN be the syntax element mb_type of the macroblock with macroblock address mbAddrN and, when mbTypeN is equal to P_8x8 or P_8x8ref0, let subMbTypeN be the syntax element list sub_mb_type of the macroblock with macroblock address mbAddrN.
     b. The derivation process for macroblock and sub-macroblock partition indices as specified in subclause 6.4.13.4 is invoked with the luma location (xW, yW), the macroblock type mbTypeN, and, when mbTypeN is equal to P_8x8 or P_8x8ref0, the list of sub-macroblock types subMbTypeN as the inputs and the outputs are the macroblock partition index mbPartIdxN and the sub-macroblock partition index subMbPartIdxN.
     c. When the partition given by mbPartIdxN and subMbPartIdxN is not yet decoded, the macroblock partition mbPartIdxN and the sub-macroblock partition subMbPartIdxN are marked as not available.

     NOTE – The latter condition is, for example, the case when mbPartIdx = 2, subMbPartIdx = 3, xD = 4, yD = −1, i.e., when neighbour C of the last 4x4 luma block of the third sub-macroblock is requested.

6.4.12 Derivation process for neighbouring locations

Input to this process is a luma or chroma location (xN, yN) expressed relative to the upper left corner of the current macroblock.

Outputs of this process are:

- mbAddrN: either equal to CurrMbAddr or to the address of neighbouring macroblock that contains (xN, yN) and its availability status,
- (xW, yW): the location (xN, yN) expressed relative to the upper-left corner of the macroblock mbAddrN (rather than relative to the upper-left corner of the current macroblock).

Let maxW and maxH be variables specifying maximum values of the location components xN, xW, and yN, yW, respectively. maxW and maxH are derived as follows:

- If this process is invoked for neighbouring luma locations,

$$maxW = maxH = 16 \quad (6\text{-}23)$$

- Otherwise (this process is invoked for neighbouring chroma locations),

$$maxW = MbWidthC \quad (6\text{-}24)$$

$$maxH = MbHeightC \quad (6\text{-}25)$$

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

6.4.12.1 Specification for neighbouring locations in frames

The derivation process for neighbouring macroblock addresses and their availability in subclause 6.4.9 is invoked with mbAddrA, mbAddrB, mbAddrC, and mbAddrD as well as their availability status as the output.

Table 6-3 specifies mbAddrN depending on (xN, yN).

Table 6-3 – Specification of mbAddrN

xN	yN	mbAddrN
< 0	< 0	mbAddrD
< 0	0..maxH − 1	mbAddrA
0..maxW − 1	< 0	mbAddrB
0..maxW − 1	0..maxH − 1	CurrMbAddr
> maxW − 1	< 0	mbAddrC
> maxW − 1	0..maxH − 1	not available
	> maxH − 1	not available

The neighbouring location (xW, yW) relative to the upper-left corner of the macroblock mbAddrN is derived as

$$xW = (xN + maxW) \% maxW \tag{6-26}$$

$$yW = (yN + maxH) \% maxH \tag{6-27}$$

6.4.13 Derivation processes for block and partition indices

Subclause 6.4.13.1 specifies the derivation process for 4x4 luma block indices.

Subclause 6.4.13.2 specifies the derivation process for 4x4 chroma block indices.

Subclause 6.4.13.3 specifies the derivation process for 8x8 luma block indices.

Subclause 6.4.13.4 specifies the derivation process for macroblock and sub-macroblock partition indices.

6.4.13.1 Derivation process for 4x4 luma block indices

Input to this process is a luma location (xP, yP) relative to the upper-left luma sample of a macroblock.

Output of this process is a 4x4 luma block index luma4x4BlkIdx.

The 4x4 luma block index luma4x4BlkIdx is derived by

$$luma4x4BlkIdx = 8 * (yP / 8) + 4 * (xP / 8) + 2 * ((yP \% 8) / 4) + ((xP \% 8) / 4) \tag{6-28}$$

6.4.13.2 Derivation process for 4x4 chroma block indices

Input to this process is a chroma location (xP, yP) relative to the upper-left chroma sample of a macroblock.

Output of this process is a 4x4 chroma block index chroma4x4BlkIdx.

The 4x4 chroma block index chroma4x4BlkIdx is derived by

$$chroma4x4BlkIdx = 2 * (yP / 4) + (xP / 4) \tag{6-29}$$

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

6.4.13.3 Derivation process for 8x8 luma block indices

Input to this process is a luma location (xP, yP) relative to the upper-left luma sample of a macroblock.

Outputs of this process is an 8x8 luma block index luma8x8BlkIdx.

The 8x8 luma block index luma8x8BlkIdx is derived by

$$luma8x8BlkIdx = 2 * (yP / 8) + (xP / 8) \quad (6\text{-}30)$$

6.4.13.4 Derivation process for macroblock and sub-macroblock partition indices

Inputs to this process are:

– a luma location (xP, yP) relative to the upper-left luma sample of a macroblock,

– a macroblock type mbType,

– when mbType is equal to P_8x8 or P_8x8ref0, a list of sub-macroblock types subMbType with 4 elements.

Outputs of this process are:

– a macroblock partition index mbPartIdx,

– a sub-macroblock partition index subMbPartIdx.

The macroblock partition index mbPartIdx is derived as follows:

– If mbType specifies an I macroblock type, mbPartIdx is set equal to 0.

– Otherwise (mbType does not specify an I macroblock type), mbPartIdx is derived by

$$mbPartIdx = (16 / MbPartWidth(mbType)) * (yP / MbPartHeight(mbType)) + (xP / MbPartWidth(mbType)) \quad (6\text{-}31)$$

The sub-macroblock partition index subMbPartIdx is derived as follows:

– If mbType is not equal to P_8x8 or P_8x8ref0, subMbPartIdx is set equal to 0.

– Otherwise (mbType is equal to P_8x8 or P_8x8ref0), subMbPartIdx is derived by

$$subMbPartIdx = (8 / SubMbPartWidth(subMbType[mbPartIdx])) * ((yP \% 8) / SubMbPartHeight(subMbType[mbPartIdx])) + ((xP \% 8) / SubMbPartWidth(subMbType[mbPartIdx])) \quad (6\text{-}32)$$

7 Syntax and semantics

7.1 Normative Syntax and Semantics

7.1.1 Normative and Informative Technologies

The normative requirements of this specification extend only to the technologies required to implement the profile specified in A.2.1. All other aspects of this specification are informative only, and not normative. Specifically, a conforming decoder is not required to handle a number of technologies, including but not limited to the following:

a) field coding (i.e. frame_mbs_only_flag equal to 0);

b) color sampling formats other than 4:2:0;

c) picture size scaling;

d) $BitDepth_Y$ and $BitDepth_C$ values other than 8;

© ISO/IEC 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

e) bipredictive and switching slice types (i.e. slice types other than I and P slices);

f) weighted prediction modes other than the default (i.e. weighted_pred_flag or weighted_bipred_idc not equal to 0);

g) entropy coding modes other than CAVLC (i.e. entropy_coding_mode_flag not equal to 0);

h) 8x8 inverse transform block size;

i) arbitrary slice order;

j) more than one slice group per picture;

7.1.2 Method of specifying syntax in tabular form

The syntax tables specify a superset of the syntax of all allowed bitstreams. Additional constraints on the syntax may be specified, either directly or indirectly, in other clauses.

NOTE – An actual decoder should implement means for identifying entry points into the bitstream and means to identify and handle non-conforming bitstreams. The methods for identifying and handling errors and other such situations are not specified here.

The following table lists examples of pseudo code used to describe the syntax. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	Descriptor
/* A statement can be a syntax element with an associated descriptor or can be an expression used to specify conditions for the existence, type, and quantity of syntax elements, as in the following two examples */	
syntax_element	ue(v)
conditioning statement	
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */	
{	
statement	
statement	
…	
}	
/* A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */	
while(condition)	
statement	
/* A "do … while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */	
do	
statement	
while(condition)	
/* An "if … else" structure specifies a test of whether a condition is true, and if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

if(condition)	
primary statement	
else	
alternative statement	
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */	
for(initial statement; condition; subsequent statement)	
primary statement	

7.2 Specification of syntax functions, categories, and descriptors

The functions presented here are used in the syntactical description. These functions assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream.

byte_aligned( ) is specified as follows:

- If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of byte_aligned( ) is equal to TRUE.
- Otherwise, the return value of byte_aligned( ) is equal to FALSE.

more_data_in_byte_stream( ), which is used only in the byte stream NAL unit syntax structure specified in Annex B, is specified as follows:

- If more data follow in the byte stream, the return value of more_data_in_byte_stream( ) is equal to TRUE.
- Otherwise, the return value of more_data_in_byte_stream( ) is equal to FALSE.

more_rbsp_data( ) is specified as follows:

- If there is no more data in the RBSP, the return value of more_rbsp_data( ) is equal to FALSE.
- Otherwise, the RBSP data is searched for the last (least significant, right-most) bit equal to 1 that is present in the RBSP. Given the position of this bit, which is the first bit (rbsp_stop_one_bit) of the rbsp_trailing_bits( ) syntax structure, the following applies:
  - If there is more data in an RBSP before the rbsp_trailing_bits( ) syntax structure, the return value of more_rbsp_data( ) is equal to TRUE.
  - Otherwise, the return value of more_rbsp_data( ) is equal to FALSE.

  The method for enabling determination of whether there is more data in the RBSP is specified by the application (or in Annex B for applications that use the byte stream format).

more_rbsp_trailing_data( ) is specified as follows:

- If there is more data in an RBSP, the return value of more_rbsp_trailing_data( ) is equal to TRUE.
- Otherwise, the return value of more_rbsp_trailing_data( ) is equal to FALSE.

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

next_bits(n) provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. Provides a look at the next n bits in the bitstream with n being its argument. When used within the byte stream as specified in Annex B, next_bits(n) returns a value of 0 if fewer than n bits remain within the byte stream.

read_bits(n) reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element

- b(8): byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function read_bits(8).
- ce(v): context-adaptive variable-length entropy-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.2.
- f(n): fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function read_bits(n).
- i(n): signed integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a two's complement integer representation with most significant bit written first.
- me(v): mapped Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- se(v): signed integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- te(v): truncated Exp-Golomb-coded syntax element with left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.
- ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

nal_unit(NumBytesInNALunit) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_ref_idc	u(2)
nal_unit_type	u(5)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
for(i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++)	
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
}	

7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

7.3.2.1 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	**Descriptor**
seq_parameter_set_data( )	
rbsp_trailing_bits( )	
}	

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.2.1.1 Sequence parameter set data syntax

seq_parameter_set_data( ) {	**Descriptor**
profile_idc	u(8)
constraint_set0_flag /* normally equal to 1 */	u(1)
constraint_set1_flag /* normally equal to 1 */	u(1)
constraint_set2_flag /* normally equal to 1 */	u(1)
constraint_set3_flag	u(1)
constraint_set4_flag /* equal to 0; ignored by decoders */	u(1)
constraint_set5_flag /* equal to 0; ignored by decoders */	u(1)
reserved_zero_2bits /* equal to 0 */	u(2)
level_idc	u(8)
seq_parameter_set_id	ue(v)
log2_max_frame_num_minus4	ue(v)
pic_order_cnt_type	ue(v)
if(pic_order_cnt_type == 0)	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
else if(pic_order_cnt_type == 1) {	
delta_pic_order_always_zero_flag	u(1)
offset_for_non_ref_pic	se(v)
offset_for_top_to_bottom_field	se(v)
num_ref_frames_in_pic_order_cnt_cycle	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)	
offset_for_ref_frame[i **]**	se(v)
}	
max_num_ref_frames	ue(v)
gaps_in_frame_num_value_allowed_flag	u(1)
pic_width_in_mbs_minus1	ue(v)
pic_height_in_map_units_minus1	ue(v)
frame_mbs_only_flag /*equal to 1*/	u(1)
direct_8x8_inference_flag	u(1)
frame_cropping_flag	u(1)
if(frame_cropping_flag) {	
frame_crop_left_offset	ue(v)
frame_crop_right_offset	ue(v)
frame_crop_top_offset	ue(v)
frame_crop_bottom_offset	ue(v)
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
}	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.2.2 Picture parameter set RBSP syntax

pic_parameter_set_rbsp( ) {	**Descriptor**
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag /*equal to zero*/	u(1)
bottom_field_pic_order_in_frame_present_flag	u(1)
num_slice_groups_minus1 /*equal to zero*/	ue(v)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
weighted_pred_flag /* = 0 */	u(1)
weighted_bipred_idc /* = 0 */	u(2)
pic_init_qp_minus26 /* relative to 26 */	se(v)
pic_init_qs_minus26 /* relative to 26 */	se(v)
chroma_qp_index_offset	se(v)
deblocking_filter_control_present_flag	u(1)
constrained_intra_pred_flag	u(1)
redundant_pic_cnt_present_flag /* equal to zero*/	u(1)
rbsp_trailing_bits( )	
}	

7.3.2.3 Supplemental enhancement information RBSP syntax

sei_rbsp( ) {	**Descriptor**
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.2.3.1 Supplemental enhancement information message syntax

sei_message( ) {	**Descriptor**
payloadType = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

7.3.2.4 Access unit delimiter RBSP syntax

access_unit_delimiter_rbsp( ) {	**Descriptor**
primary_pic_type	u(3)
rbsp_trailing_bits( )	
}	

7.3.2.5 End of sequence RBSP syntax

end_of_seq_rbsp( ) {	**Descriptor**
}	

7.3.2.6 End of stream RBSP syntax

end_of_stream_rbsp( ) {	**Descriptor**
}	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.2.7 Filler data RBSP syntax

filler_data_rbsp( ) {	**Descriptor**
while(next_bits(8) == 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

7.3.2.8 Slice layer RBSP syntax

slice_layer_rbsp( ) {	**Descriptor**
slice_header( )	
slice_data( ) /* all categories of slice_data( ) syntax */	
rbsp_slice_trailing_bits( )	
}	

7.3.2.9 (void)

7.3.2.10 RBSP slice trailing bits syntax

rbsp_slice_trailing_bits( ) {	**Descriptor**
rbsp_trailing_bits( )	
}	

7.3.2.11 RBSP trailing bits syntax

rbsp_trailing_bits( ) {	**Descriptor**
rbsp_stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned( ))	
rbsp_alignment_zero_bit /* equal to 0 */	f(1)
}	

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.3 Slice header syntax

slice_header( ) {	**Descriptor**
first_mb_in_slice	ue(v)
slice_type	ue(v)
pic_parameter_set_id	ue(v)
frame_num	u(v)
if(IdrPicFlag)	
idr_pic_id	ue(v)
if(pic_order_cnt_type = = 0) {	
pic_order_cnt_lsb	u(v)
if(bottom_field_pic_order_in_frame_present_flag)	
delta_pic_order_cnt_bottom	se(v)
}	
if(pic_order_cnt_type = = 1 && !delta_pic_order_always_zero_flag) {	
delta_pic_order_cnt[0 **]**	se(v)
if(bottom_field_pic_order_in_frame_present_flag)	
delta_pic_order_cnt[1 **]**	se(v)
}	
if(slice_type = = P) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag)	
num_ref_idx_l0_active_minus1	ue(v)
}	
ref_pic_list_modification( )	
if(nal_ref_idc != 0)	
dec_ref_pic_marking( )	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
disable_deblocking_filter_idc	ue(v)
if(disable_deblocking_filter_idc != 1) {	
slice_alpha_c0_offset_div2	se(v)
slice_beta_offset_div2	se(v)
}	
}	
}	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.3.1 Reference picture list modification syntax

ref_pic_list_modification( ) {	**Descriptor**
if(slice_type % 5 != 2 && slice_type % 5 != 4) {	
ref_pic_list_modification_flag_l0	u(1)
if(ref_pic_list_modification_flag_l0)	
do {	
modification_of_pic_nums_idc	ue(v)
if(modification_of_pic_nums_idc == 0 || modification_of_pic_nums_idc == 1)	
abs_diff_pic_num_minus1	ue(v)
else if(modification_of_pic_nums_idc == 2)	
long_term_pic_num	ue(v)
} while(modification_of_pic_nums_idc != 3)	
}	
}	

7.3.3.2 (void)

7.3.3.3 Decoded reference picture marking syntax

dec_ref_pic_marking( ) {	**Descriptor**
if(IdrPicFlag) {	
no_output_of_prior_pics_flag	u(1)
long_term_reference_flag	u(1)
} else {	
adaptive_ref_pic_marking_mode_flag	u(1)
if(adaptive_ref_pic_marking_mode_flag)	
do {	
memory_management_control_operation	ue(v)
if(memory_management_control_operation == 1 || memory_management_control_operation == 3)	
difference_of_pic_nums_minus1	ue(v)
if(memory_management_control_operation == 2)	
long_term_pic_num	ue(v)
if(memory_management_control_operation == 3 || memory_management_control_operation == 6)	
long_term_frame_idx	ue(v)
if(memory_management_control_operation == 4)	
max_long_term_frame_idx_plus1	ue(v)
} while(memory_management_control_operation != 0)	
}	
}	

© ISO/IEC 2015 – All rights reserved
Documents Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.4 Slice data syntax

slice_data( ) {	**Descriptor**
CurrMbAddr = first_mb_in_slice	
moreDataFlag = 1	
prevMbSkipped = 0	
do {	
if(slice_type != I) {	
mb_skip_run	ue(v)
prevMbSkipped = (mb_skip_run > 0)	
for(i=0; i<mb_skip_run; i++)	
CurrMbAddr = NextMbAddress(CurrMbAddr)	
if(mb_skip_run > 0)	
moreDataFlag = more_rbsp_data( )	
}	
if(moreDataFlag)	
macroblock_layer( )	
moreDataFlag = more_rbsp_data( )	
CurrMbAddr = NextMbAddress(CurrMbAddr)	
} while(moreDataFlag)	
}	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.5 Macroblock layer syntax

macroblock_layer( ) {	**Descriptor**
mb_type	ue(v)
if(mb_type == I_PCM) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
for(i = 0; i < 256; i++)	
pcm_sample_luma[i **]**	u(v)
for(i = 0; i < 128)	
pcm_sample_chroma[i **]**	u(v)
} else {	
if(mb_type != I_4x4 && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) == 4)	
sub_mb_pred(mb_type)	
else	
mb_pred(mb_type)	
if(MbPartPredMode(mb_type, 0) != Intra_16x16)	
coded_block_pattern	
if(CodedBlockPatternLuma > 0 || CodedBlockPatternChroma > 0 || MbPartPredMode(mb_type, 0) == Intra_16x16) {	
mb_qp_delta	se(v)
residual( )	
}	
}	
}	

7.3.5.1 Macroblock prediction syntax

mb_pred(mb_type) {	**Descriptor**
if(MbPartPredMode(mb_type, 0) == Intra_4x4 || MbPartPredMode(mb_type, 0) == Intra_16x16) {	
if(MbPartPredMode(mb_type, 0) == Intra_4x4)	
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {	
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	u(1)
if(!prev_intra4x4_pred_mode_flag**[** luma4x4BlkIdx **]**)	
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	u(3)
}	
intra_chroma_pred_mode	ue(v)
} else {	
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[mbPartIdx **]**	te(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)	
for(compIdx = 0; compIdx < 2; compIdx++)	
mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**	se(v)
}	
}	

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.5.2 Sub-macroblock prediction syntax

sub_mb_pred(mb_type) {	**Descriptor**
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)	
sub_mb_type[mbPartIdx **]**	ue(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)	
if(num_ref_idx_l0_active_minus1 > 0 && mb_type != P_8x8ref0	
ref_idx_l0[mbPartIdx **]**	te(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)	

for(subMbPartIdx = 0;
subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]);
subMbPartIdx++)	

for(compIdx = 0; compIdx < 2; compIdx++)	
mvd_l0[mbPartIdx **][** subMbPartIdx **][** compIdx **]**	se(v)
}	

7.3.5.3 Residual data syntax

residual( ) {	**Descriptor**
residual_luma(i16x16DClevel, i16x16AClevel, level4x4)	
Intra16x16DCLevel = i16x16DClevel	
Intra16x16ACLevel = i16x16AClevel	
LumaLevel4x4 = level4x4	
for(iCbCr = 0; iCbCr < 2; iCbCr++)	
if(CodedBlockPatternChroma & 3) /* chroma DC residual present */	
residual_block(ChromaDCLevel[iCbCr], 0, 3, 4)	
else	
for(i = 0; i <4; i++)	
ChromaDCLevel[iCbCr][i] = 0	
for(iCbCr = 0; iCbCr < 2; iCbCr++)	
for(i4x4 = 0; i4x4 < 4; i4x4++)	
if(CodedBlockPatternChroma & 2) /* chroma AC residual present */	
residual_block(ChromaACLevel[iCbCr][i4x4], 0,14,15)	
else	
for(i = 0; i < 15; i++)	
ChromaACLevel[iCbCr][i4x4][i] = 0	
}	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.5.3.1 Residual luma syntax

residual_luma(i16x16DClevel, i16x16AClevel, level4x4) {	**Descriptor**
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)	
residual_block(i16x16DClevel, 0, 15, 16)	
for(i8x8 = 0; i8x8 < 4; i8x8++)	
for(i4x4 = 0; i4x4 < 4; i4x4++)	
if(CodedBlockPatternLuma & (1 << i8x8))	
if(MbPartPredMode(mb_type, 0) = = Intra_16x16)	
residual_block(i16x16AClevel[i8x8*4+ i4x4], 0, 14, 15)	
else	
residual_block(level4x4[i8x8* 4 + i4x4], 0, 15, 16)	
else if(MbPartPredMode(mb_type, 0) = = Intra_16x16)	
for(i = 0; i < 15; i++)	
i16x16AClevel[i8x8 * 4 + i4x4][i] = 0	
else	
for(i = 0; i < 16; i++)	
level4x4[i8x8 * 4 + i4x4][i] = 0	
}	

© ISO/IEC 2015 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015

7.3.5.3.2 Residual block CAVLC syntax

residual_block(coeffLevel, startIdx, endIdx, maxNumCoeff) {	**Descriptor**
for(i = 0; i < maxNumCoeff; i++)	
coeffLevel[i] = 0	
coeff_token	ce(v)
if(TotalCoeff(coeff_token) > 0) {	
if(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)	
suffixLength = 1	
else	
suffixLength = 0	
for(i = 0; i < TotalCoeff(coeff_token); i++)	
if(i < TrailingOnes(coeff_token)) {	
trailing_ones_sign_flag	u(1)
levelVal[i] = 1 − 2 * trailing_ones_sign_flag	
} else {	
level_prefix	ce(v)
levelCode = (level_prefix << suffixLength)	
if(suffixLength > 0 || level_prefix >= 14) {	
level_suffix	u(v)
levelCode += level_suffix	
}	
if(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)	
levelCode += 2	
if(levelCode % 2 == 0)	
levelVal[i] = (levelCode + 2) >> 1	
else	
levelVal[i] = (−levelCode − 1) >> 1	
if(suffixLength == 0)	
suffixLength = 1	
if(Abs(levelVal[i]) > (3 << (suffixLength − 1)) && suffixLength < 6)	
suffixLength++	
}	
if(TotalCoeff(coeff_token) <endIdx − startIdx + 1) {	
total_zeros	ce(v)
zerosLeft = total_zeros	
} else	
zerosLeft = 0	
for(i = 0; i < TotalCoeff(coeff_token) − 1; i++) {	
if(zerosLeft > 0) {	
run_before	ce(v)
runVal[i] = run_before	
} else	
runVal[i] = 0	
zerosLeft = zerosLeft − runVal[i]	
}	
runVal[TotalCoeff(coeff_token) − 1] = zerosLeft	
coeffNum = −1	
for(i = TotalCoeff(coeff_token) − 1; i >= 0; i− −) {	
coeffNum += runVal[i] + 1	

© ISO/IEC 2015 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 14496-29:2015