



IEEE Standard for Advanced Audio and Video Coding

STANDARDS

IEEE Computer Society

Developed by the
Standards Activities Board

IEEE Std 1857™-2023
(Revision of IEEE Std 1857-2013)



IEEE Standard for Advanced Audio and Video Coding

Developed by the

Standards Activities Board
of the
IEEE Computer Society

Approved 5 June 2023

IEEE SA Standards Board

THIS IS A REDLINE DOCUMENT PRODUCED BY TECHSTREET, AN ASME COMPANY.

This document is intended to provide users with an indication of changes from one edition to the next. It includes a full-text version of the new document, plus an indication of changes from the previous version.

Redlines are designed to save time and improve efficiencies by using the latest software technology to find and highlight document changes. More professionals are using valuable new technologies like redlines, to help improve outcomes in a fast paced global business world.

Because it may not be technically possible to capture all changes accurately, it is recommended that users consult previous editions as appropriate. In all cases, only the current base version of this publication is to be considered the official document.

This Redline is produced solely by Techstreet. IEEE has not prepared this Redline and is not responsible for its accuracy.

Redline Processing Notes:

1. ~~Red Text~~ - Red strikethrough text denotes deletions.
2. Blue Text - Blue underlined text denotes modifications and additions.

Abstract: This standard defines a set of tools for efficient video coding, including directional intra prediction, variable block size inter prediction and context adaptive binary arithmetic coding and the corresponding decoding procedure.

The target applications and services include but not limited TV over Internet, User Generated Multimedia Content, IP-based video conference, IP-based surveillance and other video/audio enabled services and applications such as digital television broadcasting, digital storage media, and communication.

Keywords: block, coefficients, decoding, encoding, entropy coding, field, frame, IEEE 1857, image, inter prediction, intra prediction, IPTV, macroblock, picture, quantization, slice, transform

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2023 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 25 October 2023. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-9803-6 STD26230
Print: ISBN 978-1-5044-9804-3 STDPD26230

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus balloting process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#).¹ An IEEE account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.²

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

¹ Available at: <https://development.standards.ieee.org/myproject-web/public/view.html#landing>.

² Available at: <https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html>.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).⁵

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

Participants

At the time this IEEE standard was completed, the Audio Video Coding Working Group had the following membership:

Siwei Ma, *Chair*
Cliff Reader, *Vice Chair*

Xiaopeng Fan
Yun He
Tiejun Huang
Xiangyang Ji
Guoping Li
Fan Liang
Jian Lou

Yan Lu
Qiang Wang
Ronggang Wang
Zhenyu Wang
Feng Wu
Lianhuan Xiong

Lu Yu
Zhihua Zeng
Li Zhang
Xianguo Zhang
Haiwu Zhao
Jianhua Zheng
Xiaozhen Zheng

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Oscar Au
Bill Brown
James Carroll
Eric Chai
Pin Chang
Xuesong Gao
Werner Hoelzl
Cheng Huang
Piotr Karocki
Li Li
Fan Liang

Changyu Liu
Dong Liu
Jinghui Lu
Xiaoxu Luan
Siwei Ma
M. Sabarimalai Manikandan
Luntian Mou
Rajesh Murthy
Bansi Patel
Dev Paul

Benjamin Rolfé
Eugene Stoudenmire
Walter Struppler
Bo Sun
Yonghong Tian
John Vergis
Yaowei Wang
Feng Wu
Lu Yu
Yu Yuan
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 5 June 2023, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past President*
Konstantinos Karachalios, *Secretary*

Sara R. Biyabani
Doug Edwards
Ramy Ahmed Fathy
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick

Joseph S. Levy
Howard Li
Gui Lin
Johnny Daozhuang Lin
Kevin W. Lu
Daleep C. Mohla
Andrew Myles

Paul Nikolich
Annette D. Reilly
Robby Robson
Lei Wang
F. Keith Waters
Karl Weber
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not normative part of IEEE Std 1857-2023, IEEE Standard for Advanced Audio and Video Coding.

This standard provides regular high quality and efficient coding tool sets for compression, decompression, and packaging of the video data to save the storage space or bandwidth for transmission over the internet.

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Word usage	1
1.4 Applications	2
1.5 Profiles and levels	2
1.6 Overview of the design characteristics	2
1.7 Predictive coding	3
1.8 Picture partitioned into macroblocks and smaller partitions	3
1.9 Transform and quantization	3
2. Normative references	3
3. Definitions, acronyms, and abbreviations	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	9
4. Operators and mathematical functions	9
4.1 Description	9
4.2 Arithmetic operators	9
4.3 Logical operators	10
4.4 Relational operators	10
4.5 Bit-wise Operators	10
4.6 Assignment Operators	11
4.7 Mathematical Functions	11
4.8 Structure Relation	12
5. Description method of bitstream syntax, parsing process and decoding process	12
5.1 Description style	12
5.2 Functions	14
5.3 Descriptors	15
5.4 Reserved, forbidden ,and marker	15
6. Structure of Coded Bitstream	16
6.1 Sequence	16
6.2 Picture	17
6.3 Slices	21
6.4 Macroblock	23
6.5 8×8 Block	24
6.6 4×4 Block	25
7. Bitstream Syntax and Semantics	25
7.1 Syntax Description	25
7.2 Semantics Description	44
8. Parsing Process	67
8.1 k th -Order Exp-Golomb Codes	67
8.2 ue(v), se(v), me(v) and me4(v)	68
8.3 ce(v) and ce4(v)	71
8.4 ae(v)	73
9. Decoding Process	87

9.1 High Level Syntax Structures	87
9.2 Picture Header Decoding.....	87
9.3 Slice decoding.....	91
9.4 Macroblock decoding	91
9.5 Block Decoding	116
9.6 Quantization Coefficients Prediction	121
9.7 Inverse Quantization.....	124
9.8 Inverse Transform.....	128
9.9 Intra Prediction	129
9.10 Inter Prediction	135
9.11 Reconstruction.....	148
9.12 Loop Filter	149
Annex A	154
Annex B.....	155
B.1 Description.....	155
B.2 Profiles	155
B.3 Levels.....	158
Annex C.....	163
Annex D	169

IEEE Standard for Advanced Audio and Video Coding

~~IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.~~

~~This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.~~

1. Overview

1.1 Scope

This standard defines a set of tools for efficient video coding and the corresponding decoding procedure, including intra prediction, inter prediction, transform, quantization, and entropy coding.

1.2 Purpose

This standard provides efficient coding tool sets for compression, decompression, and packaging of the video data to save ~~the~~ storage space and ~~the~~ bandwidth for transmission over the Internet.

The target applications and services include but are not limited to Internet Protocol television (IPTV), IP-based video conference, IP-based surveillance, user-generated multimedia content, and other video/audio-enabled services and applications such as digital television broadcasting, digital storage media, and communication.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall equals is required to*).^{1,2}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should equals is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may equals is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can equals is able to*).

1.4 Applications

This standard is designed to cover a broad range of applications, including but not limited to, the following:

CATV	cable TV on optical networks, copper, etc.
DBS	direct broadcast satellite video services
DSL	digital subscriber line video services
DTTB	digital terrestrial television broadcasting
IPTV	Internet Protocol television
ISM	interactive storage media (optical disks, etc.)

¹ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

² The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact. 11

MMM	multimedia mailing
MSPN	multimedia services on packet networks
RTC	real-time conversational services (videoconferencing, videophone, etc.)
RVS	remote video surveillance

1.5 Profiles and levels

This standard can serve a wide range of bit rates, resolutions, and qualities. Considering the capability of interoperation, a limited number of subsets of the syntax are stipulated by means of “profiles” and “levels.”

A “profile” is a subset of the syntax elements, semantics, and algorithmic features of this standard.

A “level” is a specified set of limits on the syntax elements and the values that may be taken by the syntax elements of a certain profile.

1.6 Overview of the design characteristics

In this standard, a number of techniques may be used to achieve highly efficient compression, including intra coding, inter coding, transform, quantization, and entropy coding. Intra coding uses various spatial prediction modes to exploit the spatial statistical dependencies in the source signal of one picture. Inter coding uses motion vectors for block-based inter prediction to exploit the temporal statistical dependencies between the different pictures. The prediction residual is then further compressed through using a transform to remove spatial correlation inside the transform block before it is quantized. Quantization is an irreversible process that typically discards less important visual information and provides a close approximation to the source samples. Finally, the intra prediction modes or motion vectors are transmitted with the quantization coefficients and encoded by entropy coding.

1.7 Predictive coding

Intra coding need not refer to other pictures⁷ and intra coded pictures can provide random access points to the coded sequence.

Inter coding need to refer ~~refers~~ to prior coded pictures, and the order of the decoding process can be different from the order of the source picture capture process or the output order for display. In inter coding, motion vector precision can be up to 1/4 or 1/8 pixel, and motion vectors are coded by predictive coding.

1.8 Picture partitioned into macroblocks and smaller partitions

In this standard, the basic processing unit of the video decoding process is a macroblock, which consists of a 16×16 block of luma samples and corresponding blocks of chroma samples. A macroblock can be further partitioned into sub-macroblocks for prediction. The minimum size of a sub-macroblock is 4×4.

1.9 Transform and quantization

The transform block-size is 8×8 or 4×4. Transform coefficients are scalar quantized.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

Recommendation ITU-R BT.709, Parameter values for the HDTV standards for production and international programme exchange.¹

¹ ITU publications are available from the International Telecommunication Union (<http://www.itu.int/>).

Recommendation ITU-R BT.470-2, Television systems, System M, ~~B, G~~.

Recommendation ITU-R BT.470-2, Television systems, System B, G.

SMPTE 170M, Television—Composite Analog Video Signal—NTSC for Studio Applications.²

SMPTE 240M (1987), Television—1125-Line High-Definition Production Systems—Signal Parameters.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.³

background predictive forward inter-decoded picture (S-picture): A picture that may be decoded using intra prediction from the decoded samples within the same slice or inter prediction from the previously-decoded reference pictures, using at most one motion vector and forward reference index to predict the sample values of each block.

NOTE—The reference picture of an S-picture should be a G-picture.⁴

backward prediction: Prediction by using a future reference picture (in display order).

² ~~SMPTE publications are available from the Society of Motion Picture & Television Engineers (<http://www.smpte.org/>).~~

³ *IEEE Standards Dictionary Online* ~~subscription~~ is available at: <http://dictionary.ieee.org>. An IEEE account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page. ~~http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.~~

⁴ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

bidirectional inter decoded picture (B-picture): A picture that may be decoded by using intra prediction from the decoded samples within the same slice or inter prediction from the previously decoded reference pictures, using at most two motion vectors and reference indices to predict the sample values of each block.

bidirectional prediction: A combination of forward prediction and backward prediction.

bin: A symbol that can take one of the two possible values of 0 and 1.

bin string: An ordered string of a finite number of bins. The left-most bit is the most significant bit (MSB); the right-most bit is the least significant bit (LSB).

bit string: An ordered string of a finite number of bits. The left-most bit is the ~~most significant bit~~ (MSB); the right-most bit is the ~~least significant bit~~ (LSB).

bitstream: A sequence of bits that forms the representation of coded pictures and associated data forming one or more coded video sequences.

bitstream buffer: A buffer to store the bitstream.

bitstream order: Arrangement order of the coded pictures in a bitstream, which is the same as the decoding order of the picture.

block: An $M \times N$ (M -column by N -row) array of samples, or an $M \times N$ array of transform coefficients.

block scan: A specific sequential order of quantization coefficients.

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 the first bit.

byte-aligned: A bit in a bitstream is byte-aligned when its position is an integer multiple of 8 bits from the first bit in the bitstream.

chroma: An adjective specifying that a sample array or single sample is representing one of the two color difference signals related to the primary colors.

NOTE—The symbols used for a chroma array or sample_c are Cb and Cr.

coded picture: A coded representation of a picture.

component: An array or single sample from one of the three arrays (one luma and two chroma arrays) that make up a field or frame. In a different context, component is a type of video information that is transmitted or stored as two or more separate signals.

core picture: A kind of picture at most uses only one reference picture ~~which~~ ~~that~~ is correctly decoded after the nearest random access point and also at most uses one ~~reference~~ ~~reference~~ picture. In the bitstream, the first P-picture or S-picture after such picture shall use only one reference picture.

decode processing: The decode processing includes ~~the~~ parsing process and decoding process.

decoded picture: A reconstructed picture that is derived from the bitstream.

decoded picture buffer: A buffer holding decoded pictures for reference, output reordering, or output delay.

decoder: An embodiment of a decode processing.

decoding order: The order in which the coded pictures are decoded according to the prediction relations among pictures.

decoding process: The process that produces decoded pictures from syntax elements.

dequantization: The process by ~~which~~ scaling quantization coefficients gets transform coefficients.

display order: The order to display decoded pictures. A picture without display order shall not be output.

encoder: An embodiment of an encoding process.

encoding process: The process that produces a bitstream conforms to [IEEE Std 1857](#) ~~the defined~~ ~~syntax description~~.

field: An assembly of ~~alternate~~ ~~alternative~~ rows of a frame. A frame is composed of two fields, a top field_z and a bottom field.

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

forbidden: ~~The~~ ~~This~~ term ~~forbidden~~ specifies some values of syntax elements.

NOTE 1—These values ~~will~~ ~~shall~~ not be used in bitstreams conforming to this standard.

NOTE 2—The aim of specifying forbidden values is usually to avoid ~~a~~ pseudo start code prefix in the bitstream.

forward inter-~~z~~decoded picture (P-picture): A picture that may be decoded by using intra prediction from the decoded samples within the same slice or inter prediction from previously-decoded reference pictures, using at most one motion vector and forward reference index to predict the sample values of each block.

forward prediction: Prediction by using a previous reference pictures in display order.

frame: A frame contains an array of luma samples (Y) and two corresponding arrays of chroma samples (Cb and Cr).

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

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

inter-view prediction: Prediction by using a reference picture in reference view.

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

intra-~~z~~decoded background picture (G-picture): A picture that is decoded by using prediction only from the decoded samples within the same slice.

NOTE—If a G-picture is coded as two fields, the first field should be intra decoded.

intra-decoded picture (I-picture): A picture that is decoded using prediction only from the decoded samples within the same slice.

NOTE—If I-picture is coded as two fields, ~~then~~ the first field should be intra decoded.

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

inverse transform: A part of the decoding process by which a set of transform coefficients ~~are~~ ~~is~~ converted into spatial-domain values.

layer: One of a set of syntactical structures in a non-branching hierarchical relation. Higher layers contain lower layers. The coding layers starting from high to low are as follows: sequence, picture, slice, macroblock, and block layers.

level: It specifies a set of limits on the syntax elements and the values that may be taken by the syntax elements of a certain profile. In a different context, level is the value of a transform coefficient prior to inverse quantization.

luma: An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colors.

NOTE—The symbol used for luma is Y.

macroblock: A 16×16 block of luma samples and two corresponding blocks of chroma samples.

macroblock address: A macroblock address is the index of a macroblock. It starts from the top-left macroblock in a picture and continues in raster scan order. The initial value is zero.

macroblock location: The two-dimensional coordinates of a macroblock in a picture denoted by (x, y) . If coded data of two fields of current picture is appear in alternate ~~alternating~~ order, 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. If coded data of two fields of a current picture appears ~~appear~~ in sequential order, for the top left macroblock of the top field, (x, y) is equal to $(0, 0)$. x is incremented by 1 for each macroblock column of the top field from left to right. y is incremented by 1 for each macroblock row of the top field from top to bottom. For the top left macroblock of the bottom field (x, y) is equal to $(0, (H+31)>>5)$, where H is the number of samples in vertical direction. x is incremented by 1 for each macroblock column of the bottom field ~~field~~ from left to right. y is incremented by 1 for each macroblock row of the bottom field from top to bottom.

macroblock row: A continuous group of macroblock with the same vertical positions from the left boundary to the right boundary in one coded picture. The height is 16 luma samples.

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.

output order: The order in which the decoded pictures are output from the decoded picture buffer, which is ~~the~~ same as the display order of decoded pictures.

output reordering delay: The delay between the instance of decoding a coded picture from a bitstream and the instance of outputting the decoded picture. It is due to the difference between the display order and the decoding order of a picture.

output processing: It specifies how to derive output frames or fields from the decoded pictures.

parsing process: The process specified in this standard reads a bitstream and produces syntax elements.

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

picture reordering: A process during which the decoded pictures are reordered if there is a difference between their display order and decoding order.

prediction: An embodiment of the prediction process.

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

prediction value: The previously decoded sample value or the combination of the previously decoded data elements used by the decoding process of the subsequent sample values or data elements.

profile: A specified subset of the syntax elements, semantics, and algorithmic features [of this standard](#). Profile sorted into group and category.

quantization coefficient: The value of ~~the~~ transform coefficient before dequantization.

quantization parameter: A variable used by the decoding process for dequantization of quantization coefficients.

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

random access point: A point in a bitstream to start random access.

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) are scanned from left to right.

reference index: An index into a list of reference pictures (or fields of reference pictures) in ~~a~~ decoded picture buffer.

reference picture: A picture contains samples that may be used for inter prediction in the decoding process of subsequent pictures in decoding order.

reference view: A picture contains samples that may be used for inter-view prediction in the decoding process of subsequent views in decoding order.

reserved: ~~The~~ ~~This~~ term reserved, when used in the clauses specifying ~~to specify~~ some values of a particular syntax element, is for future extensions of the publication ~~syntax~~.

NOTE—These values should not be used in bitstreams conforming to this standard.

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

run: A number of consecutive data elements represented in the decoding process. In one context, run refers to 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 or a field scan. In other contexts, run refers to a number of skipped macroblocks.

sample: A basic element that forms a picture.

sample aspect ratio: It specifies, for assisting the display process, which is not specified in this 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. ~~The~~ Sample aspect ratio is expressed as $h:v$, where h is horizontal width and v is vertical height (in arbitrary units of spatial distance).

sample value: A value containing ~~the~~ sign and magnitude of a sample.

sequence: The highest hierarchical structure of coded bitstream, which comprises one or more consecutive coded pictures.

skipped macroblock: A macroblock for which no data is ~~are~~ coded other than an indication that the macroblock is to be decoded as “skipped”. This indication may be common to several macroblocks.

slice: An integer number of macroblocks ~~macroblock~~ rows ordered consecutively in the raster scan.

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

slice set: A set of all slices, which has the same slice index, in a picture.

source: Term used to describe an uncompressed video material or some certain properties of this material.

start code: A unique codeword of 32 bits embedded in the bitstream.

NOTE—Start code has many usages and can be used to identify the start of a syntax structure in bitstream.

stuffing bits: A bit string inserted into the bitstream during encoding. It is disposed of before the decoding process.

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

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.

variable-length coding: A reversible procedure of entropy coding that assigns shorter codewords expected to be more frequent symbols and longer codewords to symbols expected to be less frequent.

view sequence: A sequence which ~~that~~ has a view identifier. It is one of the multi-view sequences.

X-profile decoder: A decoder capable of decoding the bitstreams conforming to a certain profile.

3.2 Acronyms and abbreviations

BBV bitstream buffer verifier

CBR constant bit rate

LSB least significant bit

MB macroblock

MSB most significant bit

4. Operators and mathematical functions

4.1 Description

The mathematical operators used in this [standard 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.

4.2 Arithmetic operators

The following arithmetic operators are defined [in](#) ~~as~~ Table 1.

Table 1 — Arithmetic operators

Arithmetic operator	Definition
+	Addition
-	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
×	Multiplication
a^b	Exponentiation. Specifies a to the power of b . 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 .
$\div$	Used to denote division in mathematical equations where no truncation and rounding are intended.
$\frac{a}{b}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=a}^b f(i)$	The summation of $f(i)$ with i taking all integer values from a up to and including b .
$a \% b$	Modulus. Remainder of a divided by b , defined only for integers a and b with $a \geq 0$ and $b > 0$.

[\(New\)](#)

Arithmetic operator	Definition
+	Addition
-	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
×	Multiplication
a^b	Exponentiation. Specifies a to the power of b . 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 and rounding are intended.
$\frac{a}{b}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=a}^b f(i)$	The summation of $f(i)$ with i taking all integer values from a up to and including b .
$a \% b$	Modulus. Remainder of a divided by b , defined only for integers a and b with $a \geq 0$ and $b > 0$.

(Deleted)

4.3 Logical operators

The following logical operators are defined in Table 2.

Table 2 — Logical operators

Logical operator	Definition
$a \&\& b$	Boolean logical “and” of a and b
$a b$	Boolean logical “or” of a and b
!	Boolean logical “not”

4.4 Relational operators

The following relational operators are defined in Table 3.

Table 3 — Relational operators

Relational operator	Definition
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

(New)

Relational operator	Definition
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

(Deleted)

4.5 Bit-wise operators

The following bit-wise operators are defined in [as](#) Table 4.

Table 4 — Bit-wise operators

Bit-wise operator	Definition
&	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 “not”.
a >> b	Arithmetic right shift of a two’s complement integer representation of a by b binary digits. This function is defined only for positive integer values of b. Bits shifted into the most significant bit (MSB) as a result of the right shift shall have a value equal to the MSB of a prior to the shift operation.
a << b	Arithmetic left shift of a two’s complement integer representation of a by b binary digits. This function is defined only for positive integer values of b. Bits shifted into the least significant bit (LSB) as a result of the left shift have a value equal to 0.

(New)

Bit-wise operator	Definition
$\&$	Bit-wise “and.” When operating on integer arguments, it 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, it 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.
$\sim$	Bit-wise “not.”
$a \gg b$	Arithmetic right shift of a two’s complement integer representation of a by b binary digits. This function is defined only for positive integer values of b . Bits shifted into the MSB as a result of the right shift shall have a value equal to the MSB of a prior to the shift operation.
$a \ll b$	Arithmetic left shift of a two’s complement integer representation of a by b binary digits. This function is defined only for positive integer values of b . Bits shifted into the LSB as a result of the left shift have a value equal to 0.

(Deleted)

4.6 Assignment operators

The following arithmetic operators are defined in as Table 5.

Table 5 — Assignment operators

Assignment operator	Definition
$=$	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 $x = x - (-3)$.

(New)

Assignment operator	Definition
=	Assignment operator.
++	Increment, i.e., $x++$ is equivalent to $x = x + 1$; when used in an array index, it 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, it 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 $x = x - (-3)$.

(Deleted)

4.7 Mathematical functions

The following mathematical functions are defined as follows in Equation (1) through Equation (8):

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

$$\text{Ceil}(x) = \lceil x \rceil \quad (2)$$

$$\text{Clip1}(x) = \text{Clip3}(0, 2^n - 1, x) \quad (3)$$

$$\text{Clip3}(i, j, x) = \begin{cases} i & ; \quad x < i \\ j & ; \quad x > j \\ x & ; \quad \text{others} \end{cases} \quad (4)$$

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

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases} \quad (6)$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases} \quad (7)$$

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x \geq 0 \\ -1 & ; \quad x < 0 \end{cases} \quad (8)$$

4.8 Structure relation

The structure relation is defined in Table 6.

Table 6 — Structure relation

Structure relation	Definition
$\rightarrow$	Structure dereference, i.e., $a \rightarrow b$ means reference member b of structure a .

5. Description method of bitstream syntax, parsing process, and decoding process

5.1 Description style

The description style of the syntax is similar to C. Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lowercase letters with underscore characters) 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 uppercase letter ~~letters~~ and without any underscore characters. Variables starting with an uppercase letter are derived for the decoding of the current syntax structure and all depending on syntax structures. Variables starting with an uppercase letter may be used in the decoding process for later syntax structures mentioning the originating syntax structure of the variable. Variables starting with a lowercase letter are only used within the subclause from which they are derived.

The association of values and names is specified in the text. In some cases, “mnemonic” names for syntax element values or variable values are used interchangeably with their numerical values. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an uppercase letter and may contain more uppercase letters.

Hexadecimal notation, indicated by prefixing the hexadecimal number by “0x” ~~‘0x’~~, may be used when the number of bits is an integer multiple of four 4. For example, “0x1a” ~~‘0x1a’~~ represents a bit-string “ 0001 1010” ~~1010’~~.

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

The following table below shows an example of pseudo-code. It always means to read one data element from the bitstream whenever syntax elements are there.

	Descriptor
/* A statement can be a syntax element with associated descriptor or can be an expression used to specify its existence, type, and value, as in the following examples */	
syntax element	ue(v)
conditioning statement	
/* A group of statements enclosed in brackets is a compound statement and is treated functionally as a single statement. */	
{	
statement	
statement	
...	
}	
/* A "while" structure specifies that the statement is to be evaluated repeatedly while the condition remains true. */	
while (condition)	
statement	
/* A "do ... while" structure executes the statement once, and then tests the condition. It repeatedly evaluates the statement while the condition remains true. */	
do	
statement	
while (condition)	
/* An "if ... else" structure tests the condition first. If it is true, the primary statement is evaluated. Otherwise, the alternative statement is evaluated. If the alternative statement is unnecessary to be evaluated, the "else" and corresponding alternative statement can be omitted. */	
if (condition)	
primary statement	
else	
alternative statement	
/* A "for" structure evaluates the initial statement at the beginning then tests the condition. If it is true, the primary and subsequent statements are evaluated until the condition becomes false. */	
for (initial statement; condition; subsequent statement)	
primary statement	

The parsing process and decoding process are described in pseudo-code similar to C programming language.

5.2 Functions

[Functions below](#) ~~The functions that follow~~ are used to describe syntax. These functions assume that there is a bitstream pointer in ~~the~~ decoder. The pointer indicates the position of the next bit to be read from the bitstream. A function name and the parameters in the parentheses make up a function; however, a function may also exist without parameters.

byte_aligned(): If the current position of bitstream is byte-aligned, ~~then~~ it returns true. Otherwise, it returns false.

next_bits(n): [next_bits\(n\)](#) provides the next n bits in the bitstream with MSB first, without changing the bitstream pointer. If the remaining number of bits within the byte stream is less than n, ~~then~~ it returns 0.

byte_aligned_next_bits(n): If the current position in bitstream is not byte-aligned, [byte_aligned_next_bits\(-n-\)](#) returns n bits with MSB first starting from the next byte after current bitstream pointer position. The bitstream pointer is not changed. If the remaining number of bits within the byte stream is less than n, ~~then~~ it returns 0. If the current position of bitstream is byte-aligned, [byte_aligned_next_bits\(-n-\)](#) returns ~~the~~ next n bits in the bitstream with MSB first, without changing the bitstream pointer. If the remaining number of bits is less than n, ~~then~~ it returns 0.

next_start_code(): [next_start_code\(\)](#) seeks the next start code prefix in bitstream. The function is detailed in the following table:-

next_start_code() {	Descriptor
stuffing_bit	'1'
while (! byte_aligned())	
stuffing_bit	'0'
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
stuffing_byte	'0000 0000'
}	

[stuffing_byte](#) shall be present in ~~the~~ bitstream after ~~the~~ picture header and before the start code of the first slice.

is_end_of_slice(): [is_end_of_slice\(\)](#) checks whether the slice is end in the bitstream. The function is detailed in the following table: 

<code>is_end_of_slice() {</code>	Descriptor
<code>if (byte_aligned()) {</code>	
<code>if (next_bits(32) == 0x80000001</code>	
<code>return TRUE; // End of slice</code>	
<code>}</code>	
<code>else {</code>	
<code>if (byte_aligned_next_bits(24) == 0x000001 && is_stuffing_pattern())</code>	
<code>return TRUE; // End of slice</code>	
<code>}</code>	
<code>return FALSE;</code>	
<code>}</code>	

is_stuffing_pattern(): [is_stuffing_pattern\(\)](#) checks whether the remaining bits of current byte or the next byte at byte aligned position are stuffing bits. The function is detailed in the following table: 

<code>is_stuffing_pattern() {</code>	Descriptor
<code>if (next_bits(8-n) == (1<<(7-n))</code>	
<code>// n: 0~7, is the position offset of the bitstream pointer at current byte;</code>	
<code>// if n is 0, the bitstream pointer points to the most significant bit of current byte</code>	
<code>return TRUE;</code>	

<code>else</code>	
<code>return FALSE;</code>	
<code>}</code>	

read_bits(n): [read_bits\(n\)](#) reads the next n bits in the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) returns 0 and does not advance the bitstream pointer.

5.3 Descriptors

The ~~following~~ descriptors [below](#) specify the parsing process of syntax elements: 

ce(v): Advanced entropy-coded syntax element with the first bit on the left. The parsing process is specified in 8.4.

b(8): A byte with arbitrary value (8 bits). The parsing process for this descriptor is specified by the return value of `read_bits(8)`.

ce(v): Variable-length₇ entropy-coded syntax element with the first bit on the left. The parsing process is specified in 8.3.1.

ce4(v): Variable-length₇ entropy-coded syntax element with the first bit on the left. The parsing process is specified in 8.3.2.

f(n): A bit string with n bits. The parsing process is specified by the returned value of `read_bits(n)`.

i(n): Signed integer with n bits. In syntax table, if n is 'v', ~~then~~ the number of bits is determined by other syntax elements. The parsing process is specified by the return value of `read_bits(n)`, interpreted as two's complement representation with MSB first.

me(v): Mapped Exp-Golomb-coded syntax element with the first bit on the left. The parsing process is specified in 8.2.

me4(v): Mapped Exp-Golomb-coded syntax element with the first bit on the left. The parsing process is specified in 8.2.

r(n): n bits '0'. The parsing process is specified by the returned value of `read_bits(n)`.

se(v): Signed integer Exp-Golomb-coded syntax element with the first bit on the left. The parsing process is specified in 8.2.

u(n): Unsigned integer with n bits. In syntax table, if n is 'v', the number of bits is determined by other syntax elements. The parsing process is specified by the returned value of `read_bits(n)`, interpreted as two's complement representation with MSB first.

ue(v): Unsigned integer Exp-Golomb-coded syntax element with the first bit on the left. The parsing process is specified in 8.2.

5.4 Reserved, forbidden, and marker

In the bitstream syntax defined by this standard, the value of some syntax elements is marked as ‘reserved’ or ‘forbidden’.

The term ‘reserved’, when used in the clauses specifying some values of a particular syntax element, is for future uses. These values shall not be used in bitstreams conforming to this standard, but may be used in future extensions of this standard.

The term ‘forbidden’ specifies some values of syntax elements that shall not be used in bitstreams conforming to this standard.

marker_bit specifies a bit with value 1.

reserved_bits specify that some particular syntax elements are used for future extension of this standard. The decode processing shall ignore these bits. In the series of consecutive reserved_bits bytes, there shall not be a string of more than 21 consecutive ‘0’ s.

6. Structure of coded bitstream

This subclause ~~clause~~ specifies the structure of coded bitstream, the dependency on its layers, and the order of processing.

6.1 Sequence

6.1.1 Description

Sequence is the highest syntax structure of the coded bitstream. It starts with a sequence header, followed by one or more coded pictures. Each coded picture starts with a picture header and is ordered accordingly in the bitstream. The order of these pictures in the bitstream is the same as one in decoding, but it can be different from display order. video_sequence_end_code indicates the end of a video sequence.

6.1.2 Progressive and interlaced sequences

This standard supports coding of both progressive and interlaced sequences.

A frame in sequences consists of three matrices of integer samples: a luminance matrix (Y), and two chrominance matrices (Cb and Cr). The ~~relationship~~ between Y , Cb , and Cr components and the primary (analogue) Red, Green, and Blue signals ($E'R$, $E'G$ and $E'B$), the chromaticity of these primaries and the transfer characteristics of these primaries may be specified in the bitstream. This information does not affect the decoding process.

A field consists of every other line of samples in the three rectangular matrices of integers representing a frame. That is, the first, third, fifth line and so on form a top field. The rest form a bottom field.

A decoder outputs a series of frames or fields. An interval time exists between two successive frames for progressive sequence, i.e., one frame duration. For interlaced sequences, an interval time of two successive fields is referred to as one field duration. For progressive sequence, the interval time between the top and the bottom field is 0.

6.1.3 Sequence header

A video sequence header commences with a `video_sequence_start_code` followed by a series of syntax elements.

The initial sequence header can be allowed to appear repeatedly in the bitstream—which is called the repeated sequence header—and it makes random access into the potential sequence.

The first picture after a sequence header can be I-picture, G-picture, S-picture, or P-picture using inter-view prediction. The first P-picture after a sequence header should use pictures after the sequence header for forward prediction. When editing a bitstream or making a random access, all the data before a repeated sequence header can be discarded, and the resulting bitstream should also conform to this standard.

6.2 Picture

6.2.1 Description

One picture is one frame, with the encoded data commencing with a picture_header_code. It terminates with a sequence header, video_sequence_end_code₇ or next picture header.

In the bitstream, coded data of two fields with interlaced scan pattern can appear in sequential order or in alternate ~~alternative~~ order. The decoding order and display order of the two fields are determined in the picture header.

The decode processing of coded pictures includes parsing process and decoding process.

6.2.2 Picture format

6.2.2.1 4:0:0 format

In this format, there is only one Y matrix, as shown in Figure 1.

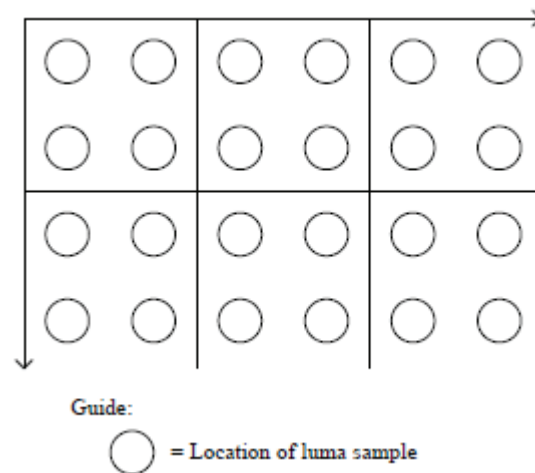


Figure 1— Vertical and horizontal locations of 4:0:0 luma and chroma samples in a picture

6.2.2.2 4:2:0 format

In this format, the Cb and Cr matrices shall be half the size of the Y matrix in both horizontal and vertical dimensions, and the Y matrix shall have an even number of lines and samples, as shown in Figure 2.

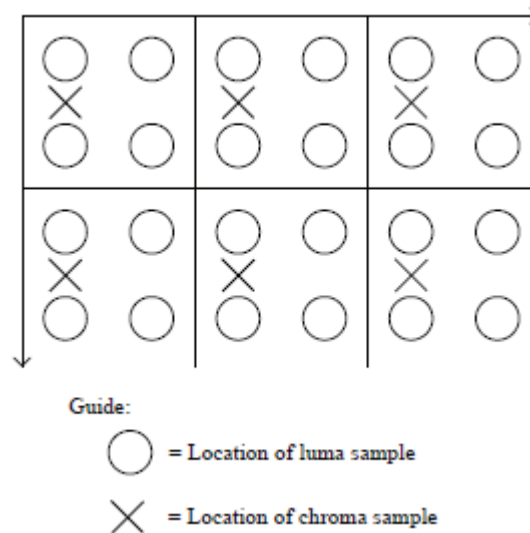


Figure 2— Vertical and horizontal locations of 4:2:0 luma and chroma samples in a picture

6.2.2.3 4:2:2 format

In 4:2:2 sampling, each of the two chroma arrays has the same height but half the width of the luma array, as shown in Figure 3.

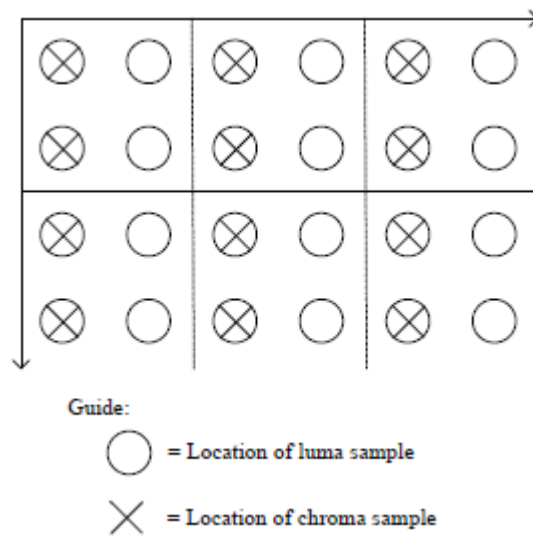


Figure 3— Vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture

6.2.2.4 4:4:4 format

In 4:4:4 sampling, each of the two chroma arrays has the same height and width as the luma array, as shown in Figure 4.

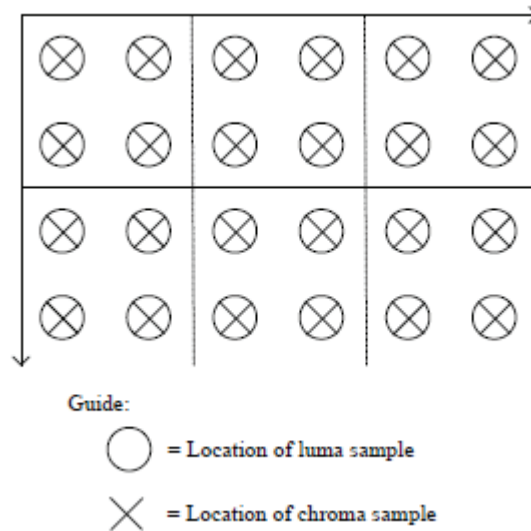


Figure 4— Vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture

6.2.3 Picture types

This standard specifies five types of decoded pictures, *as follows*:

- Intra-decoded picture (I-picture).
- Intra-decoded background picture (G-picture).
- Forward inter-decoded picture (P-picture).
- Background predictive forward inter decoded picture (S-picture).
- Bi-directional inter-decoded picture (B-picture).

6.2.4 Frame reordering

When the bitstream does not contain a B-picture, the decoding order of pictures is the same as the display order. When a B-picture is present, the decoding order is different from the display order. The decoded pictures should be reordered according to the following rules:

- If the current decoded picture is a B-picture, ~~then~~ output the current decoded picture directly.
- If the current decoded picture is an I-, a G-, an S-, or a P-picture, ~~then~~ output the previous decoded I-, S- or P-picture if it existed; output the previous decoded G-picture if it existed and it should be output; no picture is outputted if it did not.
- When all pictures have been decoded, if there are still decoded pictures except non-output G-picture in the buffer, ~~then~~ output it.

Example 1: —There are two B-pictures between successive I and P pictures, and also two B-pictures between successive P-pictures. Picture ‘1I’ is used by picture ‘4P’ for prediction. Pictures of ‘4P’ and ‘1I’ are both used by pictures of ‘2B’ and ‘3B’ for predictions. Therefore, the decoding order of pictures is ‘1I’, ‘4P’, ‘2B’, ‘3B’ and the display order is ‘1I’, ‘2B’, ‘3B’, ‘4P’.

Input order at the encoder:

1I 2B 3B 4P 5B 6B 7P 8B 9B 10I 11B 12B 13P

Decoding order:

1I 4P 2B 3B 7P 5B 6B 10I 8B 9B 13P 11B 12B

Output order at the decoder (display order):

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

Example 2: — There are two B-pictures between the successive output G-picture and S- or P-pictures, and also two B-pictures between the successive S- and P pictures. Picture ‘1G’ (an output G-picture) is used by picture ‘4P’ for prediction. Both pictures of ‘4P’ and ‘1G’ are used by pictures of ‘2B’ and ‘3B’ for prediction. Picture ‘1G’ is used by picture ‘10S’ for prediction. Therefore, the decoding order of pictures is ‘1G’, ‘4P’, ‘2B’, ‘3B’, and the display order is ‘1G’, ‘2B’, ‘3B’, ‘4P’.

Input order at the encoder:

1G	2B	3B	4P	5B	6B	7P	8B	9B	10S	11B	12B	13P
----	----	----	----	----	----	----	----	----	-----	-----	-----	-----

Decoding order:

1G	4P	2B	3B	7P	5B	6B	10S	8B	9B	13P	11B	12B
----	----	----	----	----	----	----	-----	----	----	-----	-----	-----

Output order at the decoder (display order):

1	2	3	4	5	6	7	8	9	10	11	12	13
G	B	B	P	B	B	P	B	B	S	B	B	P

Example 3: — There are two B-pictures between the successive pictures of I and P pictures, and also two B-pictures between the successive pictures of S or P. A non-output G-picture is inserted between I- and P-pictures by the encoder. Picture ‘1I’ is used by picture ‘4P’ for prediction. Pictures ‘4P’ and ‘1G’ are both used by pictures ‘2B’ and ‘3B’ for predictions. Picture ‘1G’ (the non-output G-picture) is used by picture ‘10S’ for prediction. Therefore, the decoding order of pictures is ‘1I’, ‘1G’, ‘4P’, ‘2B’, ‘3B’, and the display order is ‘1I’, ‘2B’, ‘3B’, and ‘4P’.

Input order at the encoder:

1I 2B 3B 4P 5B 6B 7P 8B 9B 10S 11B 12B 13P

Decoding order:

1I 1G 4P 2B 3B 7P 5B 6B 10S 8B 9B 13P 11B 12B

Output order at the decoder (display order):

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	S	B	B	P

6.2.5 Reference picture

P- and B-picture can have a maximum of two reference pictures⁷, respectively. P-picture without using interview prediction can use the two prior reference pictures for forward prediction. P-picture using inter-view prediction can use one prior reference picture and one inter-view reference picture for forward prediction.

Within coding of a frame, except B-picture, a previous decoded field can also be used as reference for the next field. B-picture can use the two reference pictures for forward and backward prediction. S-picture can have a maximum of one reference picture, which shall be the latest decoded G-picture in the same view with the S-picture.

A motion vector can exceed the boundaries of a reference frame. In this case, the nearest integer pixel in the frame shall be used to extend the boundary to get integer pixels out of exceed the boundaries. The number of pixels shall not exceed the boundaries of a reference picture by 16 in both ~~the~~ vertical and horizontal [direction](#) ~~directions~~ for luma array. As for chroma array:

- In 4:0:0 sampling, the number of pixels used for reference extension shall not exceed 16 in both ~~the~~ vertical and horizontal [direction](#) ~~directions~~.

- In 4:2:0 sampling, the number of pixels used for reference extension shall not exceed 8 in both ~~the~~ vertical and horizontal direction ~~directions~~.
- In 4:2:2 sampling, the number of pixels used for reference extension shall not exceed 8 in ~~the~~ horizontal direction and 16 in ~~the~~ vertical direction.
- In 4:4:4 sampling, the number of pixels used for reference extension shall not exceed 16 in both ~~the~~ vertical and horizontal direction ~~directions~~.

This limitation also applies for field case.

6.3 Slices

A slice consists of an integer number of macroblocks ordered consecutively in the raster scan order. Macroblock lines within a slice shall not overlap. Slices shall not overlap with each other either. ~~A~~ Slice set consists of all slices of a picture with the same slice index.

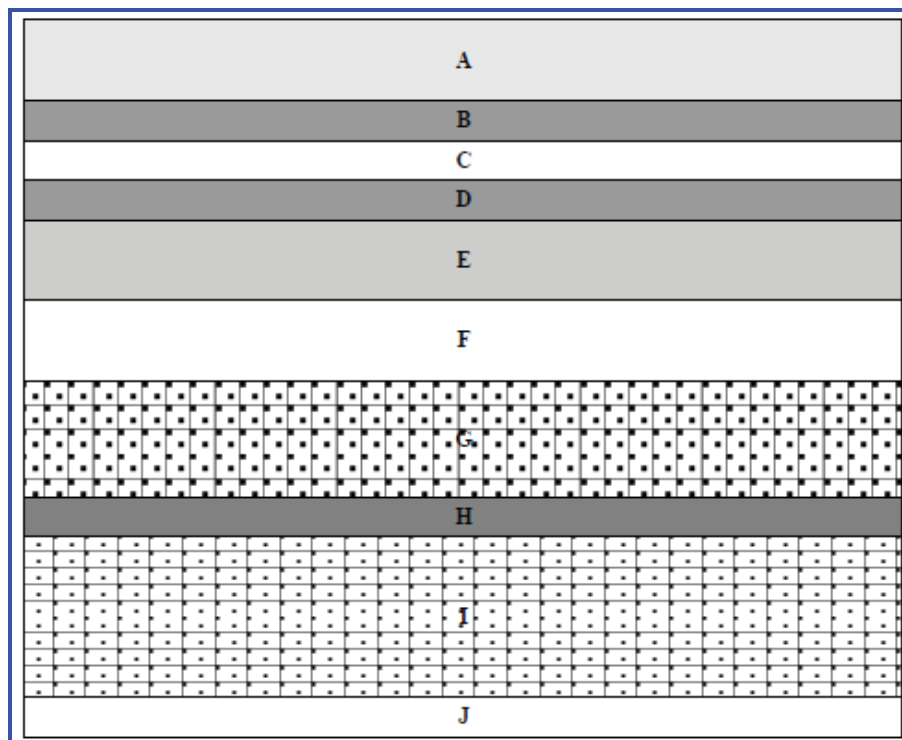
If the bitstreams of two fields with interlaced scan pattern appears ~~appear~~ in alternate ~~alternating~~ order, ~~then~~ the decoding process for macroblocks within a slice set shall not use the data of other slice sets in the same picture. If ~~a~~ slice set is disabled, ~~then~~ the decoding process for macroblocks within a slice shall not use the data of other slices in the same picture. If ~~a~~-slice set is enabled, ~~then~~ the decoding process for macroblocks within a slice can use the data of other slices in the same slice set.

If the bitstreams of two fields with ~~an~~ interlaced scan pattern appear in sequential order, ~~then~~ the data of the two fields shall belong to different slices and different slice sets, and the decoding process for macroblocks within a slice set shall not use the data of other slice sets in the same picture. If ~~the~~ slice set is disabled, ~~then~~ the decoding process for macroblocks within a slice shall not use the data of other slices in the same field. If ~~the~~ slice set is enabled, ~~then~~ the decoding process for macroblocks within a slice can use the data of other slices in the same slice set.

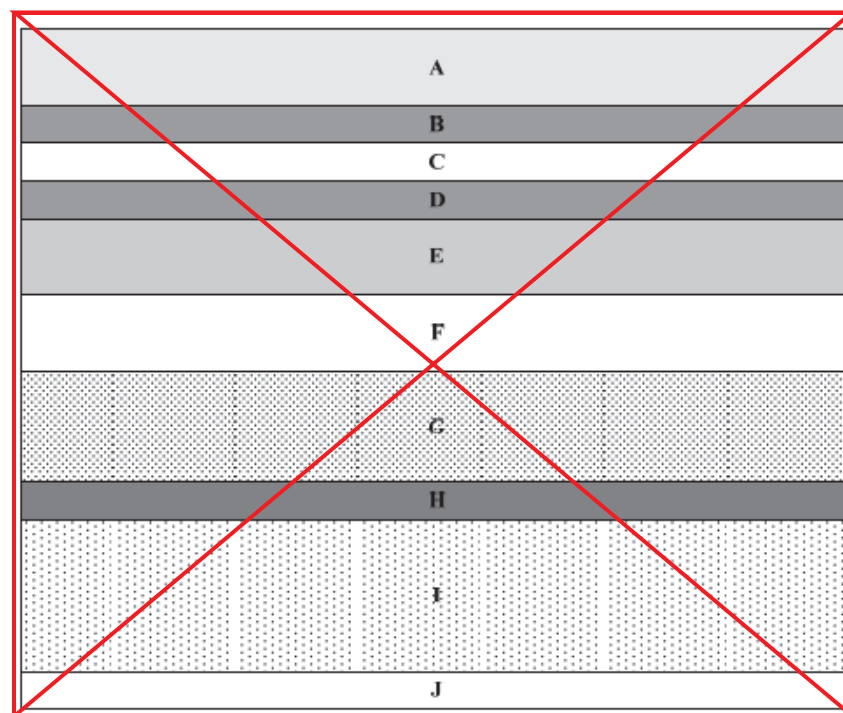
If the sequence is ~~in~~ top-bottom mode, and the bitstreams of two fields with ~~an~~ interlaced scan pattern ~~appears~~ appear in ~~alternate~~ alternating order, ~~then~~ the data of different views shall belong to different slices and different slice sets. The decoding process for macroblocks within a slice set shall not use the data of other slice sets in the same picture of the same view sequence. If ~~a~~-slice set is disabled, ~~then~~ the decoding process of macroblocks within a slice shall not use the data of other slices in the same picture of the same view sequence. If ~~a~~-slice set is enabled, ~~then~~ the decoding process for macroblocks within a slice can use the data of other slices in the same slice set.

If the sequence is ~~in~~ top-bottom mode, and the bitstreams of two fields with ~~an~~ interlaced scan pattern ~~appears~~ appear in sequential order, ~~then~~ the data of the two fields shall belong to different slices and different slice sets, and the data of different views shall belong to different slices and different slice sets. The decoding process for macroblocks within a slice set shall not use the data of other slice sets in the same field of the same view sequence. If ~~a~~ slice set is disabled, ~~then~~ the decoding process for macroblocks within a slice shall not use the data of other slices in the same field of the same view sequence. If ~~a~~ slice set is enabled, ~~then~~ the decoding process for macroblocks within a slice can use the data of other slices in the same slice set.

See Figure 5 for an example of ~~the~~ slice structure. The limitation of boundary extension of a picture also applies to ~~the~~ slice case.

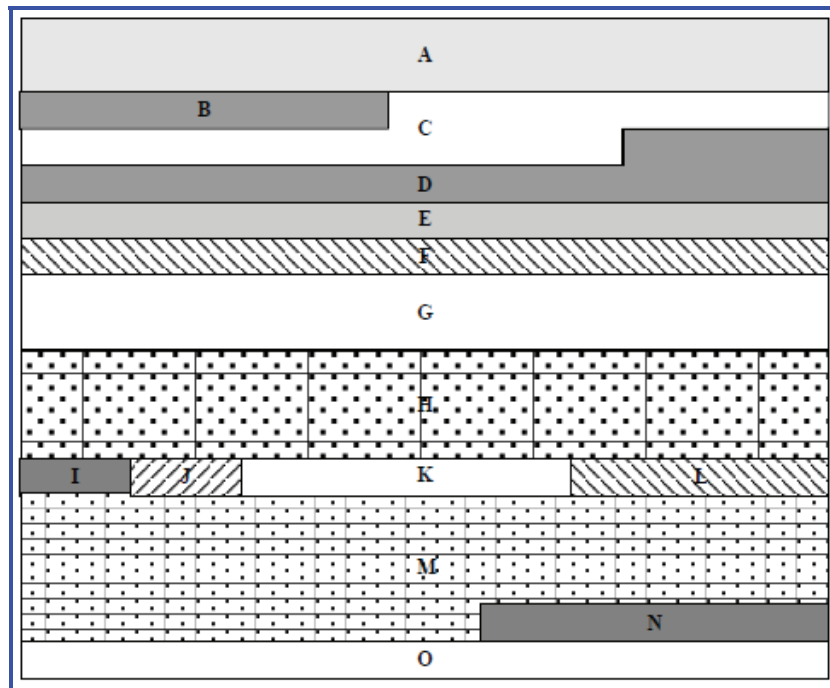


(New)

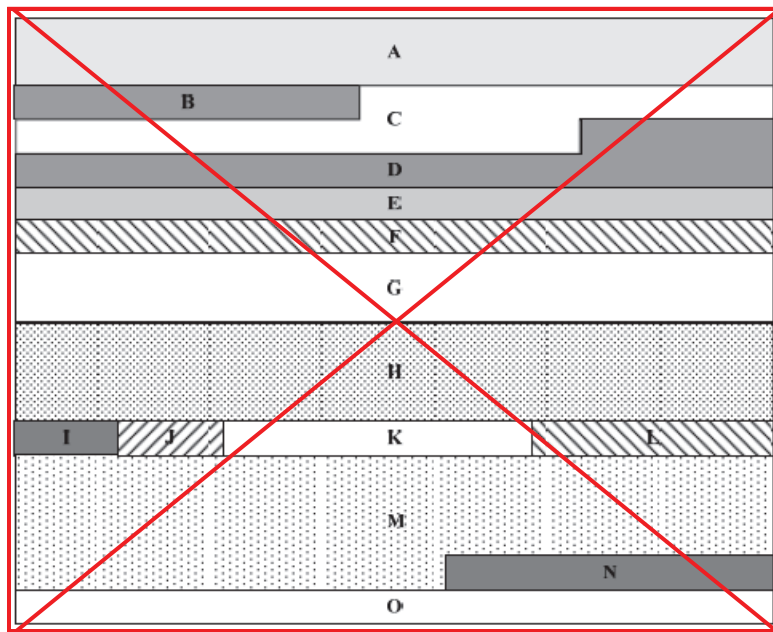


(Deleted)

(a) Slice consists of an integer number of macroblock lines



(New)



~~(Deleted)~~

(b) Slice consists of an integer number of macroblocks

Figure 5—Slice structure