



IEEE Standard for Third-Generation Video Coding

IEEE Computer Society

Developed by the
Data Compression Standard Committee

IEEE Std 1857.10™-2021

IEEE Standard for Third-Generation Video Coding

Developed by the

Data Compression Standard Committee
of the
IEEE Computer Society

Approved 9 November 2021

IEEE SA Standards Board

Abstract: A set of high-efficiency video coding tools, including intra prediction, inter prediction, transform, entropy coding and loop filter, and corresponding decoding procedure, is defined in this standard. The target applications and services include, but are not limited to, digital television broadcasting, IPTV, video surveillance, video conference, Internet video streaming, and other video/audio-enabled services and applications such as user-generated multimedia content, digital storage media, and communication.

Keywords: coding unit, decoding, encoding, entropy coding, IEEE 1857.10™, inter prediction, intra prediction, loop filter, prediction block, prediction unit, quantization, slice, transform, transform block, transform coefficients

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

Copyright © 2022 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 8 March 2022. 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 978-1-5044-8140-3 STD25072
Print: ISBN 978-1-5044-8141-0 STDPD25072

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 the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. 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 does not warrant or represent the accuracy or completeness of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to results and workmanlike effort. 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 his or her 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 development 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.

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 Standards Coordinating Committees 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](#). An IEEE Account is needed to access the application.

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

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.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US 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, IEEE does not 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](#). 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](#). Search for standard number and year of approval to access the web page of the published standard. Errata links are located 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](#).

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. 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.

Participants

At the time this IEEE standard was submitted to the IEEE SA Standards Board for approval, the Audio Video Coding Working Group had the following membership:

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

Xiaoqiang Cao	Zhongliang Li	Shuhui Wang
Fangdong Chen	Fan Liang	Yingbin Wang
Huanbang Chen	Ruling Liao	Zhenyu Wang
Jie Chen	Heyuan Lin	Yujing Wei
Wei Chen	Jucui Lin	Xi Xie
Kui Fan	Shan Liu	Zhihuang Xie
Xiaopeng Fan	Tao Lin	Xiaoyu Xiu
Cheng Fang	Shiqiang Long	Guisen Xu
Ruidong Fang	Xiaomu Lu	Weiwei Xu
Tianliang Fu	Haichuan Ma	Xiaozhong Xu
Wen Gao	Siwei Ma	Haitao Yang
Xiaoding Gao	Xiang Ma	Lu Yu
Zhewei Guo	benben Niu	Jiaqi Zhang
Ming He	Xiao Ouyang	Li Zhang
Yun He	Dongping Pan	Weimin Zhang
Ye Hu	Yinji Piao	Xue Zhang
Tiejun Huang	Yucheng Sun	Yuhuai Zhang
Chuanmin Jia	Dong Wang	Haiwu Zhao
Yunrui Jian	Fan Wang	Liping Zhao
Dong Jiang	Ji Wang	Riyang Zhao
Meng Lei	Li Wang	Yin Zhao
Ge Li	Liqiang Wang	Jiancheng Zheng
Junru Li	Mengyang Wang	Jianhua Zheng
Xinwei Li	Ronggang Wang	Xiaozhen Zheng
Yiming Li	Shanshe Wang	Kailun Zhou

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

Oscar Au	Dong Liu	Shiqi Wang
Eric Chai	Qian Liu	Yaowei Wang
Wei Gao	Haopeng Lu	Lisa Ward
Xu Han	Siwei Ma	Jianchao Wei
Yun He	Luntian Mou	Feng Wu
Werner Hoelzl	Rajesh Murthy	Yiqun Xu
Bo Jiang	Joanna Olszewska	Lu Yu
Piotr Karocki	R. K. Rannow	Yu Yuan
Alexis Kokkonos	Bo Sun	Oren Yuen
Ting Li	Yonghong Tian	Jiaqi Zhang
Weiping Li	Ronggang Wang	Xinfeng Zhang
Fan Liang		Xiaozhen Zheng

When the IEEE SA Standards Board approved this standard on 9 November 2021, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Edward A. Addy
Doug Edwards
Ramy Ahmed Fathy
J. Travis Griffith
Thomas Koshy
Joseph L. Koepfinger*
David J. Law

Howard Li
Daozhuang Lin
Kevin Lu
Daleep C. Mohla
Chenhui Niu
Damir Novosel
Annette Reilly
Dorothy Stanley

Mehmet Ulema
Lei Wang
F. Keith Waters
Karl Weber
Sha Wei
Howard Wolfman
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1857.10-2021, IEEE Standard for Third-Generation Video Coding.

This standard provides a series of high-efficiency coding tools for the compression of the video data to save the storage space or bandwidth for transmission. The target applications and services include, but are not limited to, digital television broadcasting, video surveillance, video conference, Internet video streaming, and other video/audio-enabled services and applications such as user-generated video content, digital video storage, and communication.

Contents

1. Overview	13
1.1 Scope	13
1.2 Purpose	13
1.3 Word usage	13
2. Normative references	14
3. Definitions, acronyms, and abbreviations	14
3.1 Definitions	14
3.2 Acronyms and abbreviations	20
4. Conventions	21
4.1 Overview	21
4.2 Arithmetic operators	21
4.3 Logical operators	22
4.4 Relational operators	22
4.5 Bit-wise operators	23
4.6 Assignment operators	23
4.7 Mathematical functions	23
4.8 Structure relation	25
4.9 Description method of bitstream syntax, parsing process, and decoding process	25
4.9.1 Description method of bitstream syntax	25
4.9.2 Functions	27
4.9.3 Descriptors	29
4.9.4 Reserved, forbidden, and marker	29
5. Structure of coded bitstream	30
5.1 Video sequence	30
5.1.1 Overview	30
5.1.2 Progressive and interlaced sequences	30
5.1.3 Sequence header	30
5.2 Picture	31
5.2.1 Overview	31
5.2.2 Picture format	31
5.2.3 Picture types	32
5.2.4 Picture reordering	33
5.2.5 Reference picture and reference picture list	34
5.3 Patch	34
5.4 Largest coding unit, coding unit tree, and coding unit	35
6. Bitstream syntax and semantics	35
6.1 Syntax description	35
6.1.1 Start code	35
6.1.2 Syntax of video sequence	36
6.1.3 Definition of picture	49
6.1.4 Definition of patch	58
6.1.5 Definition of coding unit tree	60
6.1.6 Definition of coding unit	63
6.1.7 Definition of transform block	74
6.1.8 Definition of adaptive leveling filter parameter	80
6.1.9 Definition of enhanced sample adaptive offset parameter	80

6.1.10 Definition of cross-component parameter.....	81
6.1.11 Definition of intra string copy prediction.....	82
6.1.12 Definition of isc-evs-ubvs-coding-unit	84
6.1.13 Definition of evs_or_unmatched_pixel_string_coding()	87
6.2 Semantics description	94
6.2.1 Video extension	94
6.2.2 Semantics description of video sequence	95
6.2.3 Picture	117
6.2.4 Patch.....	127
6.2.5 Coding unit tree.....	130
6.2.6 Coding unit	131
6.2.7 Transform block.....	141
6.2.8 Adaptive leveling filter parameters	142
6.2.9 Enhance sample adaptive offset param	143
6.2.10 Cross-component sample adaptive sample csao_ param.....	144
6.2.11 Intra string copy prediction	146
7. Parsing process	149
7.1 kth-order Exp-Golomb codes	149
7.2 Parsing process of ue(v) and se(v)	149
7.3 Parsing process of ae(v).....	150
7.3.1 Overview.....	150
7.3.2 Initialization	150
7.3.3 Parsing of binary symbol string	151
7.3.4 Inverse binarization method.....	169
8. Decoding process.....	184
8.1 Sequence decoding	184
8.2 Picture decoding	185
8.2.1 Overview.....	185
8.2.2 Picture header decoding	185
8.2.3 Derive information of reference picture list.....	186
8.2.4 Update decoded picture buffer	187
8.2.5 Construct reference picture list	188
8.2.6 Output decoded picture	188
8.2.7 Determine weighted quantization matrix	188
8.3 Patch decoding.....	191
8.4 LCU decoding.....	191
8.5 CU decoding.....	192
8.5.1 Overview.....	192
8.5.2 Determine quantization parameter	193
8.5.3 CU type and related information.....	194
8.5.4 Neighboring block	202
8.5.5 Determine the mode for splitting coding unit into transform blocks	202
8.5.6 Intra prediction mode	204
8.5.7 Motion information	221
8.6 Transform block decoding	271
8.6.1 Overview.....	271
8.6.2 Dequantization	271
8.6.3 Inverse transform	273
8.7 Intra prediction	279
8.7.1 Ordinary intra prediction.....	279
8.7.2 Intra block copy prediction	315
8.7.3 Intra string copy prediction	317

8.8 Inter prediction	321
8.8.1 Overview	321
8.8.2 Derivation of prediction sample	321
8.8.3 Prediction sample combination	350
8.8.4 Prediction sample leveling filter	362
8.9 Prediction compensation	370
8.9.1 Overview	370
8.9.2 Obtain the prediction sample matrix	370
8.9.3 Obtain the compensated sample matrix	371
8.9.4 Obtain the frequently occurring position chroma sample	371
8.10 Deblocking filter	372
8.10.1 Overview	372
8.10.2 Determination of the necessity of deblocking filter	372
8.10.3 Determine the deblocking filter adjustment parameter	373
8.10.4 Derivation process of boundary filtering strength	373
8.10.5 Boundary filtering process for luma component $B_s = 4$	377
8.10.6 Boundary filtering process for luma component $B_s = 3$	377
8.10.7 Boundary filtering process for luma component $B_s = 2$	377
8.10.8 Boundary filtering process for luma component $B_s = 1$	378
8.10.9 Boundary filtering process for luma component $B_s = 0$	378
8.10.10 Boundary filtering process for chroma component $B_s > 0$	378
8.10.11 Deblocking filtering adjustment process	378
8.11 Sample adaptive offset	379
8.11.1 Basic sample adaptive offset	379
8.11.2 Enhanced sample adaptive offset	383
8.11.3 Cross-component sample adaptive offset	385
8.12 Adaptive leveling filter	389
8.12.1 Overview	389
8.12.2 Decoding of adaptive leveling filter coefficients	389
8.12.3 Derivation of adaptive leveling filter unit	392
8.12.4 Determination of adaptive leveling filter coefficient index of luma component adaptive leveling filter unit	393
8.12.5 Process of adaptive leveling filtering	395
Annex A (normative) Pseudo-start code	398
Annex B (normative) Profiles and levels	399
B.1 Overview	399
B.2 Profiles	399
B.3 Levels	404
B.3.1 Levels defined in this standard	404
B.3.2 Profile-independent level limits	406
Annex C (normative) Bitstream buffer verifier	414
C.1 Overview	414
C.2 Conventions	414
C.2.1 Convention 1	414
C.2.2 Convention 2	414
C.2.3 Convention 3	414
C.3 Basic operation	415
C.3.1 Data input	415
C.3.2 Data removed from the buffer	417
C.4 Check time interval of BBV buffer	419
Annex D (normative) Weighted quantization matrix	420

Annex E (informative) Reference implementation method for reconstructing video signal of high dynamic range and constant luminance system	421
E.1 Overview	421
E.2 Syntax and semantics	421
E.2.1 Syntax description	421
E.2.2 Semantics description	421
E.3 Reconstruction process of high dynamic range signals	422
Annex F (informative) Optical-electro transfer function of high dynamic range scene luma fidelity	424
F.1 Overview	424
F.2 Terms and definitions	424
F.3 Electro-optic transfer function	424
F.4 Optical-electro transfer function	425
Annex G (normative) Scan table	426
Annex H (informative) Alternative implementation of AEC decoding procedures	429
Annex I (normative) Inverse transform matrix	431
I.1 DCT2 inverse transform matrix	431
I.1.1 4×4 DCT2 inverse transform matrix	431
I.1.2 8×8 DCT2 inverse transform matrix	431
I.1.3 16×16 DCT2 inverse transform matrix	431
I.1.4 32×32 DCT2 inverse transform matrix	432
I.1.5 64×64 DCT2 inverse transform matrix	433
I.2 DCT8 inverse transform matrix	438
I.2.1 4×4 DCT8 inverse transform matrix	438
I.2.2 8×8 DCT8 inverse transform matrix	438
I.2.3 16×16 DCT8 inverse transform matrix	439
I.3 DST7 inverse transform matrix	439
I.3.1 4×4 DST7 inverse transform matrix	439
I.3.2 8×8 DST7 inverse transform matrix	439
I.3.3 16×16 DST7 inverse transform matrix	439
I.3.4 32×32 DST7 inverse transform matrix	440
Annex J (normative) Array used in system coding	442
J.1 prefix_ctx	442
J.2 min_in_group	442
J.3 group_index	442
Annex K (informative) Bibliography	443

IEEE Standard for Third-Generation Video Coding

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 the compression of the video data with significant coding efficiency improvement over IEEE Std 1857.4™-2018.¹ The target applications and services include, but are not limited to, digital television broadcasting, video surveillance, video conference, Internet video streaming, and other video/audio-enabled services and applications such as user-generated video content, digital video storage, 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*).^{2, 3}

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 Information on references can be found in Clause 2.

² The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

³ The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

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.

CIE S 015, Lighting of Outdoor Workplaces.⁴

IEEE Std 1857.4TM-2018, IEEE Standard for Second-Generation IEEE 1857 Video Coding.^{5, 6}

ISO 11664-1/CIE S 014-1, Colorimetry E S 014-1 Outdoor Workplaces.sc Observers.⁷

ISO 11664-3/CIE S 014-3, Colorimetry—Part 3: CIE Tristimulus Values.

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.⁸

affine control point motion vectors: Reference motion vectors used to calculate affine prediction subblock motion vector, containing two or three motion vectors.

affine motion information: A six-tuple for inter prediction of affine prediction unit, which is composed of affine type (four parameters or six parameters), reference prediction mode, L0 motion vector, L1 motion vector, L0 reference index, and L1 reference index.

affine prediction unit: It is composed of multiple subblocks. Each subblock consists of a 4×4 or 8×8 luma prediction block and corresponding chroma prediction blocks. The motion vectors of the subblocks may be different.

affine prediction unit with four parameters: Affine prediction unit in which the number of motion vectors in the affine control point motion vector group is two.

affine prediction unit with six parameters: Affine prediction unit in which the number of motion vectors in the affine control point motion vector group is three.

affine sample refinement information: The information used to improve the prediction samples of the affine prediction unit, which consists of the affine sample refinement flag, the affine sample level refinement flag, the affine sample vertical refinement flag, the first deviation matrix, the second deviation matrix, the third deviation matrix, and the fourth deviation matrix.

⁴ CIE publications are available from the International Commission on Illumination (<https://cie.co.at/>).

⁵ The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

⁶ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<https://standards.ieee.org/>).

⁷ ISO publications are available from the International Organization for Standardization (<https://www.iso.org/>) and the American National Standards Institute (<https://www.ansi.org/>).

⁸ *IEEE Standards Dictionary Online* 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.

B picture: A picture that may be decoded in inter prediction by using a library picture and the previous and future reference pictures as reference pictures in display order.

bin: Symbols that make up bin string, including “0” and “1”.

bin string: An ordered string of 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 binary bits. The left-most bit is the most significant bit (MSB); the right-most bit is the least significant bit (LSB).

bitstream: The binary data stream formed by the coded pictures.

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 pictures.

block: An $M \times N$ (M -column by N -row) sample matrix or an $M \times N$ transform coefficient matrix.

block scan: A specific sequential order of quantization coefficients.

block vector: Block vector is a two-dimensional vector used for intra block copy prediction, and its value is the coordinate offset between the current block and the reference block, where both the current block and the reference block are in the current picture.

byte: A bit string of 8 bits.

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

chroma: A sample matrix or single sample of one of the two color difference signals C_r and C_b .

coding block: An $M \times M$ sample block, which is partitioned from the largest coding block.

coded picture: A coded representation of a picture.

coding unit: A coding unit is partitioned from the largest coding unit, including a luma coding block or corresponding chroma coding blocks, or both of them, which depends on its component model.

compensated sample: Sample obtained via prediction compensation.

component: A single matrix of the three sample matrices (one luma matrix and two chroma matrices) of the picture or a single sample in the matrices.

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

decoded picture buffer: A buffer holding decoded pictures for reference, output reordering and output timing.

NOTE—Include reference picture buffer.⁹

⁹ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

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

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

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

equal value string: A string with all of its sample values equal to each other

flag: It is a binary-state variable.

field: It is composed of alternate rows in the three sample matrices that make up the frame.

filtered sample: Sample obtained via deblocking filtering.

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

NOTE—The aim of specifying forbidden values is usually to avoid pseudo-start code in the bitstream.

frame: The representation of the spatial information of the video signal, consisting of a luma sample matrix (Y) and two chroma sample matrices (Cb and Cr).

frequently occurring position: Sample positions with the same sample value and frequently occurring in the same picture. The sample value is used as a predictor of other sample values.

historical intra copy information list: Used for intra block copy prediction or intra string copy prediction and composed of intra copy information.

historical motion information list: Used for inter prediction, it consists of the motion information of prediction unit.

historical point prediction information list: Used for nonordinary string submode and composed of point prediction information lists of prediction units coded by nonordinary string submode.

I picture: A picture that is decoded using only intra prediction.

incompletely matched sample string: A string with at least one unmatched sample.

inter coding: Coding of coding units or pictures that uses inter prediction.

inter prediction: The process of generating a current picture sample prediction value by using a previously decoded picture.

intra block copy prediction: A mode in which the decoded samples are copied to the current sample region in the same decoded picture as the current sample prediction value. The region of the copied sample is the same as the current sample region.

intra coding: Coding of coding units or pictures that uses intra prediction.

NOTE—The coding unit using ordinary intra prediction is called an “ordinary intra coding unit”; the coding unit using intra block copy prediction is called an “intra block copy coding unit”; the coding unit using intra string copy prediction is called an “intra string copy coding unit”.

intra copy information: The information used for intra block copy prediction or intra string copy prediction. It is composed of displacement vector, position, size information, and number of repetitions.

intra mode frequency list: List used for intra prediction, which consists of the frequency of the intra prediction mode of the prediction unit.

intra prediction: The process of using previously decoded samples in the same decoded picture to generate the predicted value of the current sample.

NOTE—Intra prediction includes ordinary intra prediction, intra block copy prediction, and intra string copy prediction.

intra string copy prediction: A mode in which decoded samples are copied to the current sample region in the same decoded picture as the predicted value of the current sample. The two-dimensional shape and number of samples of the region where the samples are copied are the same as those of the current sample region.

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

largest coding block: A $K \times K$ block of samples. Largest coding block is partitioned from one of the three sample matrices (one luma matrix and two chroma matrices) of the picture.

largest coding unit: Contain a $L \times L$ luma sample block and corresponding chroma sample blocks. Largest coding unit is partitioned from picture.

layer: Hierarchical structure in a bitstream. Higher layers contain lower layers. The coding layers starting from high to low are as follows: sequence, picture, patch, largest coding unit, coding unit, and coding block.

level: It specifies a set of limits on the syntax elements and the values that may be taken by the syntax elements at a certain profile.

library picture: A picture in the library stream, which can be used as a reference picture when decoding a sequence bitstream.

library stream: Bitstream containing library picture.

library stream decoder: Decoder for decoding library stream.

luma: Y represents the sample matrix or single sample of the luma signal.

motion information: The quintuple used for inter prediction, which consists of prediction reference mode, L0 motion vector, L1 motion vector, L0 reference index, and L1 reference index.

motion vector: A two-dimensional vector used for inter prediction, pointing from the current picture to a reference picture, the value of which is the coordinate offset between the current block and the reference block. The motion vectors pointing to the reference pictures in the reference picture list 0 and the reference picture list 1 are referred to as L0 motion vector and L1 motion vector, respectively.

nonordinary string submode: An intra string copy prediction submode that uses equal value string, unit basis vector string, and unmatched sample string for prediction.

offset sample: Sample obtained via sample adaptive offset.

ordinary string: A string predicted by an arbitrary valid string vector.

ordinary string submode: An intra string copy prediction submode that uses ordinary string and incompletely matched sample string for prediction.

output order: The order in which the decoded pictures are output, which is the same as the display order.

P picture: A picture that may be decoded by using only previously displayed reference pictures in inter prediction.

parse: This process reads a bitstream and produces syntax elements.

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

partition type: The type of organization of subsets obtained via partition.

patch: Adjacent number of largest coding units ordered consecutively in the raster scan.

picture: A picture can be a frame or a field.

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

prediction: An embodiment of the prediction process.

prediction block: An $M \times N$ sample block using the same prediction process. Prediction block is partitioned from coding unit.

prediction compensation: Sum of sample residue decoded from syntax elements and corresponding prediction value.

prediction partition type: The type in which one coding unit is partitioned into intra prediction blocks or inter prediction units.

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

prediction unit: A prediction unit has one luma prediction block and corresponding chroma prediction blocks, or either of them, which is partitioned from coding unit.

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.

point prediction: In intra string copy prediction, all reference samples of a current string are the same sample of a frequently occurring position and the predictors of all samples in the current string are the sample value in the frequently occurring position.

point prediction information list: Used to store all point vectors of a coding unit coded by nonordinary string submodule.

point vector: A two-dimensional vector specifies the coordinates of a frequently occurring position.

quantization coefficient: The value of transform coefficient before dequantization.

quantization parameter: A parameter 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 bitstream.

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 and third rows, and so on. Rows are scanned from left to right.

reconstructed sample: Sample for composing decoded picture, which is decoded from bitstream by decoder.

reference index: The number of the reference picture in the reference picture list. The number in the reference picture list 0 is referred to as the L0 reference index, and the number in the reference picture list 1 is referred to as the L1 reference index.

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

reference picture buffer: A buffer used to save decoded pictures and used for prediction.

reference picture list: A list of reference pictures of the current picture, which may include reference picture list 0 and reference picture list 1.

reference picture list 0: Picture list 0 composed of the reference pictures of the current picture, which may include the previous and future pictures in display order.

reference picture list 1: Picture list 1 composed of the reference pictures of the current picture, which may include the previous and future pictures in display order.

reserved: The term, when used to specify some values of a particular syntax element, is for future extensions of IEEE Std 1857.10™. These values shall not be used in bitstreams conforming to IEEE Std 1857.10.

residual: The difference between the reconstructed value and the prediction value of a sample or data element.

RL picture: A P picture or B picture that may be decoded in inter prediction by using only library picture as reference picture.

run: The number of consecutive identical data elements in the decoding process. Run refers to the number of the coefficients with a value of 0 preceding to a nonzero coefficient in block scan (in the block scan order).

sample: A basic element that forms a picture.

sample value: Amplitude value of a sample.

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

sequence stream: A stream that can be decoded with reference to a library picture provided by the information other than the stream.

sequence stream decoder: Decoder for decoding sequence stream. The decoder should contain library picture index output port and library picture input port.

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.

string vector: String vector is a two-dimensional vector used for intra string copy prediction mode, and its value is the coordinate offset between the current string and the reference string, where both the current string and the reference string are in the current picture.

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

syntax element: The result of parsing the data unit in the bitstream.

transform coefficient: A scalar quantity on the transform domain.

traverse scan: The traverse scan maps a two-dimensional rectangle to a one-dimensional string. The entry of the one-dimensional string starts from the starting row of the two-dimensional rectangle, and the entire two-dimensional rectangle is scanned progressively. The scanning direction of the starting row is from left to right, the scanning direction of adjacent rows of the starting row is from right to left, and any two adjacent rows have opposite scanning directions.

unit basis vector string: A string with string vector equal to $(0, -1)$.

unmatched sample: A sample without reference sample in intra string copy prediction.

unmatched sample string: A string with only unmatched sample(s).

width height ratio: The ratio of the horizontal distance between the columns of luma samples in a picture to the vertical distance between the rows. It is expressed as $h \times v$, where h is the number of the samples in horizontal direction and v is the number of the samples in vertical direction.

3.2 Acronyms and abbreviations

ABVR	adaptive block vector resolution
AEC	advanced entropy code
AIR	adaptive intra refresh
ALF	adaptive leveling filter
AMVR	adaptive motion vector resolution
ASR	affine sample refinement
AWP	angular weighted prediction
BBV	bitstream buffer verifier
BGC	bidirectional gradient correction
BIO	bidirectional optical flow
CB	coding block
CBR	constant bit rate
CBVP	class-based block vector prediction
CCSAO	cross-component sample adaptive offset
CPLF	cross patch loop filter
CU	coding unit

CUT	coding unit tree
DMVR	decoder-side motion vector refinement
DPB	decoded picture buffer
DT	derived tree
EALF	enhanced adaptive leveling filter
EIPM	extended intra prediction mode
ESAO	enhanced sample adaptive offset
EVS	equal value string
FIMC	frequency-based intra mode coding
HMVP	history-based motion vector prediction
IBC	intra block copy
IIP	improved intra prediction
ISC	intra string copy
IST	implicit selected transform
LCB	largest coding block
LCU	largest coding unit
LSB	least significant bit
MIPF	multiple intra prediction filter
MSB	most significant bit
MVAP	motion vector angle prediction
OBMC	overlapped block motion compensation
OS	ordinary string
PB	prediction block
PBT	position-based transform
PMC	prediction from multiple cross-component
PU	prediction unit
ROI	region of interest
SAO	sample adaptive offset
SAWP	spatial angular weighted prediction
SBT	subblock transform
SRCC	scan region-based coefficient coding
ST	secondary transform
TB	transform block
TSCPM	two-step cross-component prediction mode
UBVS	unit basis vector string
UMVE	ultimate motion vector expression

4. Conventions

4.1 Overview

The mathematical operators and priority levels used in this standard are similar to those used in the C programming language. However, integer division and arithmetic shift operations are specifically defined. Unless otherwise stated, numbering and counting conventions generally begin from 0.

4.2 Arithmetic operators

Arithmetic operators are defined in Table 1.

Table 1—Definitions of arithmetic operators

Arithmetic operator	Definition
+	Additive operation.
−	Subtraction (as a binary 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, $-7/4$ and $7/-4$ are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is 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 cumulative sum of $f(i)$ with the independent variable i taking all integer values from a up to and including b .
$a \% b$	Modular arithmetic. Remainder of a divided by b , where both a and b are positive integers.
$\lceil \cdot \rceil$	Round up.

4.3 Logical operators

Logical operators are defined in Table 2.

Table 2—Definitions of logical operators

Logical operator	Definition
$a \&\& b$	Boolean logical “and” of a and b .
$a \parallel b$	Boolean logical “or” of a and b .
!	Boolean logical “not”.
$express ? a : b$	If the result of the expression $express$ is true or not 0, then a is used for assignment; otherwise, use b for assignment.

4.4 Relational operators

Relational operators are defined in Table 3.

Table 3—Definitions of relational operators

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

4.5 Bit-wise operators

Bit-wise operators are defined in Table 4.

Table 4—Definitions of bit-wise operators

Bit-wise operator	Definition
&	Bit-wise “and”.
	Bit-wise “or”.
~	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 .
$a \ll b$	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 .

4.6 Assignment operators

Assignment operators are defined in Table 5.

Table 5—Definitions of 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, e.g., $x += 3$ is equivalent to $x = x + 3$, and $x += (-3)$ is equivalent to $x = x + (-3)$.
-=	Decrement by amount specified, e.g., $x -= 3$ is equivalent to $x = x - 3$, and $x -= (-3)$ is equivalent to $x = x - (-3)$.

4.7 Mathematical functions

The definitions of mathematical functions are shown in Equation (1) through Equation (13).

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

where

x independent variable x .

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

where

x independent variable x .

$$\text{Floor}(x) = \lfloor x \rfloor \quad (3)$$

where

x independent variable x .

$$\text{Clip1}(x) = \text{Clip3}(0, 2^{\text{BitDepth}} - 1, x) \quad (4)$$

where

x independent variable x .
 BitDepth precision of coded sample.

$$\text{Clip3}(i, j, x) = \begin{cases} i & ; \quad x < i \\ j & ; \quad x > j \\ x & ; \quad \text{others} \end{cases} \quad (5)$$

where

x independent variable x .
 i lower bound.
 j upper bound.

$$\text{Median}(x, y, z) = x + y + z - \text{Min}(x, \text{Min}(y, z)) - \text{Max}(x, \text{Max}(y, z)) \quad (6)$$

where

x independent variable x .
 y independent variable y .
 z independent variable z .

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases} \quad (7)$$

where

x independent variable x .
 y independent variable y .

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases} \quad (8)$$

where

x independent variable x .
 y independent variable y .

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x \geq 0 \\ -1 & ; \quad x < 0 \end{cases} \quad (9)$$

where

x independent variable x .

$$\text{Log}(x) = \log_2 x \quad (10)$$

where

x independent variable x .

$$\text{Ln}(x) = \log_e x \quad (11)$$

where

x independent variable x .

e base of the natural logarithm, with a value of 2.718281828...

$$\text{Rounding}(x, s) = \text{Sign}(x) \times ((\text{Abs}(x) + (1 \ll (s - 1))) \gg s) \quad s \geq 1 \quad (12)$$

where

x independent variable x .

s independent variable s .

$$\text{Exchange}(x, y) = (\text{Swap the value of } x \text{ and } y) \quad (13)$$

where

x independent variable x .

s independent variable s .

4.8 Structure relation

The structure relation is defined in Table 6.

Table 6—Structure relation

Structure relation	Definition
->	For example, $a \rightarrow b$ means that a is a structure and b is a member variable of a .

4.9 Description method of bitstream syntax, parsing process, and decoding process

4.9.1 Description method of bitstream syntax

The description method of bitstream syntax is similar to C language. Syntax elements in the bitstream are represented in bold type. Each syntax element is described by its name (all lowercase letters with underscored characters), syntax, and semantics. 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. Such variables appear in the syntax tables, or text, named by a mixture of lowercase letter and uppercase letter without underscoring. Variables starting with an uppercase letter are used for the decoding of the current and related syntax structures. Variables starting with an uppercase letter may also be used for the decoding of subsequent syntax structures. Variables starting with a lowercase letter are only used within the subclause from which they are derived.

The relationship between the mnemonic symbols of the syntax element value, the mnemonic symbols of the variable value and their values is described in the text. In some cases, the two are equivalent. A mnemonic symbol is represented by one or more groups of letters separated by underscores, each group starts with an uppercase letter and may contain more uppercase letters.

When the length of the bit string is an integral multiple of 4, it can be represented by a hexadecimal notation. The prefix of hexadecimal notation is “0x”, for example, “0x1a” represents a bit string “0001 1010”.

In the conditional statement, 0 means FALSE, and nonzero means TRUE.

The syntax tables describe all the supersets of the bitstream syntax that conform to this standard, and the additional syntax restrictions are described in the relevant clauses.

Table 7 shows an example of pseudo-code describing syntax. It always means to read one data element from the bitstream whenever syntax elements are there.

Table 7—Pseudo-code of syntax description

Pseudo-code	Descriptor
/* A statement is a descriptor of a syntax element, or indicates the existence, type, and value of a syntax element, two examples are given below. */	
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	
...	
}	
/*A “while” structure tests whether the condition is TRUE. If TRUE, the statement is executed repeatedly until the condition is not TRUE. */	
while (condition)	
statement	
/*A “do ... while” structure executes the statement once, and then tests the condition. It repeatedly executes the statement while the condition remains true. */	
do	
statement	
while (condition)	

Pseudo-code	Descriptor
/*An “if ... else” structure tests the condition first. If it is true, the primary statement is executed. Otherwise, the alternative statement is executed. If the alternative statement is unnecessary to be executed, the “else” part and corresponding alternative statement can be omitted. */	
if (condition)	
primary statement	
else	
alternative statement	
/*A “for” structure executes the initial statement first, then tests the condition. If it is TRUE, the primary and subsequent statements are executed repeatedly until the condition is not TRUE.*/	
for (initial statement; condition; subsequent statement)	
primary statement	
/*A “break” statement is used in do-while, while and for loop body, which can make the current loop body immediately terminate the loop. */	
break	

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

4.9.2 Functions

4.9.2.1 Overview

Functions in this clause are used to describe syntax. These functions assume that there is a bitstream pointer in decoder. The pointer indicates the position of the next bit to be read from the bitstream. A function name and the parameters in the left and right parentheses make up a function; however, a function may also exist without parameters.

4.9.2.2 byte_aligned()

If the current position of bitstream is byte-aligned, it returns TRUE. Otherwise, it returns FALSE.

4.9.2.3 next_bits(n)

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, it returns 0.

4.9.2.4 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 the current bitstream pointer position. The bitstream pointer is not changed. If the current position of bitstream is byte-aligned, byte_aligned_next_bits(n) returns next n bits in the bitstream with MSB first, without changing the bitstream pointer. If the remaining number of bits is less than n, it returns 0.

4.9.2.5 next_start_code()

next_start_code() seeks the next start code in bitstream, with the bitstream pointer pointing to the first bit of the start code prefix. The definition of the function is shown in Table 8.

Table 8—Definition of next_start_code function

Definition	Descriptor
next_start_code() × {	
stuffing_bit	“1”
while (! byte_aligned())	
stuffing_bit	“0”
while (next_bits(24) != “0000 0000 0000 0000 0000 0001”)	
stuffing_byte	“00000000”
}	

stuffing_byte shall be present in the bitstream after the picture header and before the start code of the first patch.

4.9.2.6 is_end_of_patch()

is_end_of_patch() checks whether the patch is end in the bitstream; if the patch is end in the bitstream, it returns TRUE; otherwise it returns FALSE. This function does not modify the bitstream pointer. The definition of the function is shown in Table 9.

Table 9—Definition of is_end_of_patch function

Definition	Descriptor
is_end_of_patch() {	
if (byte_aligned()) {	
if (next_bits(32) == 0x80000001)	
return TRUE /* End of patch */	
}	
else {	
if ((byte_aligned_next_bits(24) == 0x000001) && is_stuffing_pattern())	
return TRUE /* End of patch */	
}	
return FALSE;	
}	

4.9.2.7 is_stuffing_pattern()

is_stuffing_pattern() checks whether the remaining bits of the current byte or the next byte at byte aligned position are stuffing bits at the end of patch. If yes, it returns TRUE; otherwise it returns FALSE. This function does not modify the bitstream pointer. The definition of the function is shown in Table 10.

Table 10—Definition of is_stuffing_pattern function

Definition	Descriptor
is_stuffing_pattern() {	
if (next_bits(8-n) == (1<<(7-n))) /* n = 0 to 7, is the position offset of the bitstream pointer at current byte; if n is 0, the bitstream pointer points to the most significant bit of current byte*/	
return TRUE	
Else	
return FALSE;	
}	

4.9.2.8 read_bits(n)

read_bits(n) returns the next n bits in the bitstream with MSB first 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.

The function is also used to describe the parsing process and decoding process.

4.9.3 Descriptors

The descriptors specify the parsing process of syntax elements, as shown in Table 11.

Table 11—Descriptors

Descriptor	Description
ae(v)	Advanced entropy-coded syntax element. The parsing process is specified in 7.3.
b(8)	A byte with arbitrary value. The parsing process for this descriptor is specified by the returned value of read_bits(8).
f(n)	Consecutive n bits taking designated value. 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”, the number of bits is determined by other syntax elements. The parsing process is specified by the return value of read_bits(n), the returned value is interpreted as two’s complement with MSB first.
r(n)	Consecutive 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. The parsing process is specified in 7.2.
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), the returned value is interpreted as binary system with MSB first.
ue(v)	Unsigned integer Exp-Golomb-coded syntax element. The parsing process is specified in 7.2.

4.9.4 Reserved, forbidden, and marker

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

The term “reserved” specifies some particular syntax element values for future extensions of this standard. These values shall not be used in bitstreams conforming to this standard.

The term “forbidden” specifies some particular syntax element values that shall not be used in bitstreams conforming to this standard.

“marker_bit” specifies a bit with value 1.

“reserved_bits” in bitstream indicate that some syntax elements are reserved for future extensions 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” starting from any byte aligned position.

5. Structure of coded bitstream

5.1 Video sequence

5.1.1 Overview

The video sequence is the highest syntax structure of bitstream. The video sequence begins with the first sequence header, and the sequence end code or video edit code indicates the end of a video sequence. The sequence header between the first sequence header of the video sequence and the sequence end code or video edit code that appears first is the repetitive sequence header. Each sequence header is followed by one or more coded pictures, and each picture should have a picture header before it. The coded pictures are ordered accordingly in the bitstream. The order of these pictures in the bitstream is the same as the one in decoding, but it can be different from display order.

5.1.2 Progressive and interlaced sequences

This standard supports coding of both progressive and interlaced sequences.

A frame in sequences consists of three sample matrices: a luma sample matrix (Y) and two chroma sample matrices (Cb and Cr). The value of the sample matrix element is an integer. The relationship between Y, Cb, and Cr components and the primary (analog) Red, Green, and Blue signals (E'R, E'G, and E'B), including the chromaticity of these primaries and the transfer characteristics of these primaries, and so on, may be specified in the bitstream. This information does not affect the decoding process.

A field is composed of alternating rows in the three sample matrices that make up the frame. That is, the first, third, fifth row and so on of the frame sample matrix form a top field. The second, fourth, sixth row and so on form a bottom field.

A decoder outputs a series of pictures. There is a frame time interval between two frames. For interlaced sequence, there is a field time interval between two fields of each frame. For progressive sequence, the time interval between two fields of each frame is 0.

5.1.3 Sequence header

A video sequence header begins with the video sequence start code followed by a string of coded picture data.

The sequence header can appear repeatedly in a bitstream, which is called a “repetitive sequence header”. The primary purpose of using a repetitive sequence header is to support random access to the video sequence.

The first decoded picture after a sequence header should be I picture or RL picture.

The sequence header corresponding to the current picture is the nearest sequence header decoded before the current picture.

If the first decoded picture of the current picture after the sequence header is I picture and the current picture is displayed after this I picture, the reference picture of the current picture should be included in this I picture and the picture should be displayed after this I picture.

If the first decoded picture of the current picture after the sequence header is RL picture and the current picture is displayed after this RL picture, the reference picture of the current picture should be included in this RL picture, the picture should be displayed after this RL picture, and the library picture should be referenced by this RL picture.

When editing a bitstream or making a random access, all the data before a repetitive sequence header can be discarded, and the resulting new bitstream should also conform to this standard.

5.2 Picture

5.2.1 Overview

A picture can be a frame or a field, with the coded data starting with a picture start code and ending with a sequence start code, sequence end code, or next picture start code.

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

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

5.2.2 Picture format

5.2.2.1 4:0:0 format

In this format, there is only a Y matrix.

The location of luma sample is shown in Figure 1.

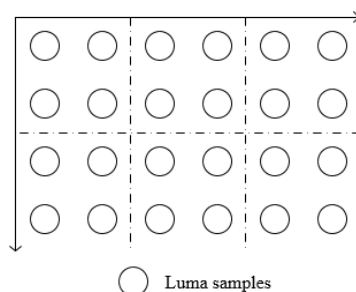


Figure 1—Location of luma sample in 4:0:0 format

5.2.2.2 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 rows and samples. In addition, if the coded data of two fields of the picture appear in sequential order, the number of rows of the Y matrix should be divided exactly by 4. The location of luma sample and chroma sample is shown in Figure 2.

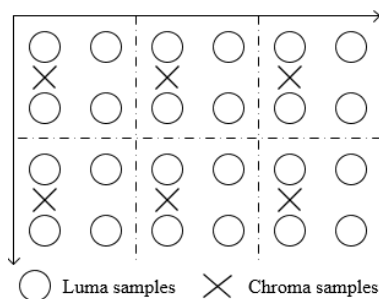


Figure 2—Location of luma and chroma samples in 4:2:0 format

5.2.2.3 4:2:2 format

In this format, the Cb and Cr matrices shall be half the size of the Y matrix in horizontal dimension and the same size of the Y matrix in vertical dimension. The number of samples per row of the Y matrix should be an even number. If the coded data of two fields of the picture appear in sequential order, the number of rows of the Y matrix should also be an even number.

The location of luma sample and chroma sample is shown in Figure 3.

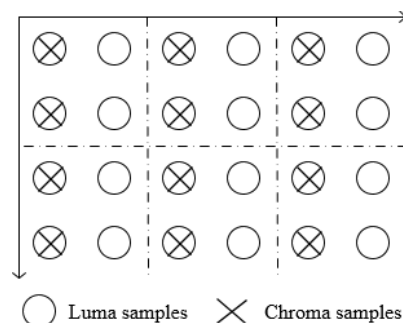


Figure 3—Location of luma and chroma samples in 4:2:2 format

5.2.2.4 4:4:4 format

In this format, the Cb and Cr matrices shall be the same size of the Y matrix in both horizontal and vertical dimensions. If the coded data of two fields of the picture appear in sequential order, the number of rows of the Y matrix should be an even number.

The location of luma sample and chroma sample is shown in Figure 4.

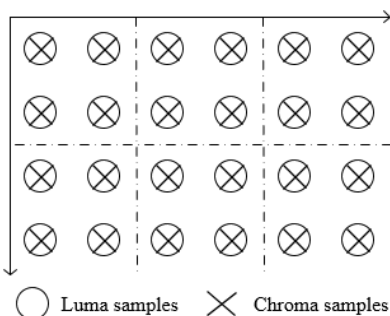


Figure 4—Location of luma and chroma samples in 4:4:4 format

5.2.3 Picture types

This standard specifies the following three types of decoded pictures:

- a) I picture
- b) P picture
- c) B picture

5.2.4 Picture reordering

When the video sequence does not contain a B picture, the decoding order is the same as the display order. When the video sequence contains a B picture, the decoding order is different from the display order. Picture reordering shall be performed before the decoded picture is output and displayed.

The first decoded picture after a sequence header shall be an I picture or RL picture. The decoding order of the pictures in the code stream that are displayed after the I picture or RL picture shall be after the I picture or RL picture.

The following examples describe the picture reordering.

Example 1: There are three B pictures between I/RL and P pictures and three B pictures between two successive P pictures. According to the decoding order, the picture 4P is predicted with the picture 0I/RL, the picture 2B is predicted with the pictures 4P and 0I/RL, the picture 1B is predicted with the pictures 2B and 0I/RL, and the picture 3B is predicted with the pictures 4P and 2B. The decoding order is 0I/RL, 4P, 2B, 1B, 3B; the display order is 0I/RL, 1B, 2B, 3B, 4P.

Sort by decoding order:

Decoding order	0	1	2	3	4	5	6	7	8	9	10	11	12
Type	I/RL	P	B	B	B	P	B	B	B	P	B	B	B
Display order	0	4	2	1	3	8	6	5	7	12	10	9	11

Sort by display order:

Display order	0	1	2	3	4	5	6	7	8	9	10	11	12
Type	I/RL	B	B	B	P	B	B	B	P	B	B	B	P
Decoding order	0	3	2	4	1	7	6	8	5	11	10	12	9

Example 2: There are seven B pictures between I/RL and P pictures and seven B pictures between two successive P pictures. According to the decoding order, the picture 8P is predicted with the picture 0I/RL, the picture 4B is predicted with the pictures 8P and 0I/RL, the picture 2B is predicted with the pictures 4B and 0I/RL, the picture 1B is predicted with the pictures 2B and 0I/RL, the picture 3B is predicted with the pictures 4B and 2B, the picture 6B is predicted with the pictures 8P and 4B, the picture 5B is predicted with the pictures 6B and 4B, and the picture 7B is predicted with the pictures 8P and 6B. The decoding order is 0I/RL, 8P, 4B, 2B, 1B, 3B, 6B, 5B, 7B; the display order is 0I/RL, 1B, 2B, 3B, 4B, 5B, 6B, 7B, 8P.

Sort by decoding order:

Decoding order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Type	I/RL	P	B	B	B	B	B	B	B	P	B	B	B	B	B	B	B
Display order	0	8	4	2	1	3	6	5	7	16	12	10	9	11	14	13	15

Sort by display order:

Display order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Type	I/RL	B	B	B	B	B	B	B	P	B	B	B	B	B	B	B	P
Decoding order	0	4	3	5	2	7	6	8	1	12	11	13	10	15	14	16	9

Example 3: There are two B pictures between I/RL and P pictures and two B pictures between two successive P pictures. According to the decoding order, the picture 3P is predicted with the picture 0I/RL and the pictures 1B and 2B are predicted with the pictures 3P and 0I/RL. The decoding order is 0I/RL, 3P, 1B, 2B; the display order is 0I/RL, 1B, 2B, 3P.

Sort by decoding order:

Decoding order	0	1	2	3	4	5	6	7	8	9	10	11	12
Type	I/RL	P	B	B	P	B	B	I	B	B	P	B	B
Display order	0	3	1	2	6	4	5	9	7	8	12	10	11

Sort by display order:

Display order	0	1	2	3	4	5	6	7	8	9	10	11	12
Type	I/RL	B	B	P	B	B	I	B	B	P	B	B	P
Decoding order	0	2	3	1	5	6	4	8	9	7	11	12	10

Example 4: There are 16 B pictures after the I/RL picture.

Sort by decoding order:

Decoding order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Type	I/RL	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
Display order	0	16	8	4	2	1	3	6	5	7	12	10	9	11	14	13	15

Sort by display order:

Display order	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Type	I/RL	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
Decoding order	0	5	4	6	3	8	7	9	2	12	11	13	10	15	14	16	1

5.2.5 Reference picture and reference picture list

The P picture may have a plurality of (previous) reference pictures that are displayed before the current picture, and these reference pictures constitute the reference picture list 0 of the P picture.

The B picture may have a plurality of (previous) reference pictures that are displayed before the current picture and a plurality of (future) reference pictures that are displayed after the current picture; these reference pictures constitute the reference picture list 0 and reference picture list 1 of the B picture.

The reference pixel indicated by the motion vector can exceed the boundaries of the reference picture. In this case, for the integer samples exceeding the boundaries of the reference picture, use the integer samples within the picture closest to the location indicated by the integer reference sample to extend the boundaries.

The field boundary extension method is the same as the reference picture boundary extension method.

5.3 Patch

A patch is a rectangular area in a picture, which contains the parts of several largest coding units that are within the picture. Patches shall not overlap with each other. See Figure 5 for the patch structure.

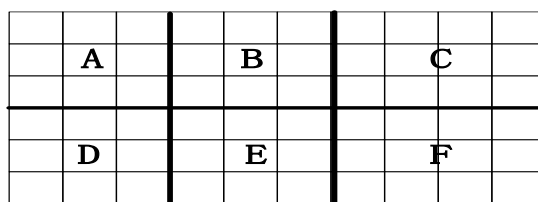


Figure 5—Patch structure

5.4 Largest coding unit, coding unit tree, and coding unit

A picture is partitioned into LCU. LCUs shall not overlap with each other. The upper left sample of an LCU should not exceed the picture boundary, and the lower right sample of an LCU may exceed the picture boundary.

An LCU is partitioned into one or more CUs, which is determined by CUT. A CU is partitioned into one or more TBs.

6. Bitstream syntax and semantics

6.1 Syntax description

6.1.1 Start code

The start code is a special bit string. In bitstreams conforming to this standard, the same bit string should not occur anywhere except in start codes.

A start code consists of a start code prefix and a start code value. The start code prefix is the bit string “0000 0000 0000 0000 0000 0001”. All start codes shall be byte aligned.

The start code value is an 8 bit integer that identifies the type of start code. See Table 12.

Table 12—Start code value

Start code type	Start code value (Hexadecimal)
patch_start_code	00~7F
reserved	80~8E
patch_end_code	8F
reserved	90~AF
video_sequence_start_code	B0
video_sequence_end_code	B1
user_data_start_code	B2
intra_picture_start_code	B3
reserved	B4
extension_start_code	B5
inter_picture_start_code	B6
video_edit_code	B7
reserved	B8
System start code	B9~FF

When assigning some values, certain syntax elements may contain the same bit string as the start code prefix. These are called “pseudo-start codes”. Encoders and decoders conforming to this standard shall apply the methods defined in Annex A to deal with pseudo-start codes.

6.1.2 Syntax of video sequence

6.1.2.1 Definition of video sequence

The definition of a video sequence is shown in Table 13.

Table 13—Definition of video sequence

Definition of video sequence	Descriptor
video_sequence() {	
do {	
sequence_header()	
extension_and_user_data(0)	
do {	
if (next_bits(32) == intra_picture_start_code)	
intra_picture_header()	
else	
inter_picture_header()	
extension_and_user_data(1)	
picture_data()	
} while ((next_bits(32) == inter_picture_start_code) (next_bits(32) == intra_picture_start_code))	
} while ((next_bits(32) != video_sequence_end_code) && (next_bits(32) != video_edit_code))	
if (next_bits(32) == video_sequence_end_code)	
video_sequence_end_code	f(32)
if (next_bits(32) == video_edit_code)	
video_edit_code	f(32)
}	

6.1.2.2 Definition of sequence header

The definition of a sequence header is shown in Table 14.

Table 14—Definition of sequence header

Definition of sequence header	Descriptor
sequence_header() {	
video_sequence_start_code	f(32)
profile_id	u(8)
level_id	u(8)
progressive_sequence	u(1)
field_coded_sequence	u(1)
library_stream_flag	u(1)
if (! LibraryStreamFlag) {	
library_picture_enable_flag	u(1)
if (LibraryPictureEnableFlag) {	

Definition of sequence header	Descriptor
duplicate_sequence_header_flag	u(1)
}	
}	
marker_bit	f(1)
horizontal_size	u(14)
marker_bit	f(1)
vertical_size	u(14)
chroma_format	u(2)
sample_precision	u(3)
if (profile_id == 0x22 profile_id == 0x32) {	
encoding_precision	u(3)
}	
marker_bit	f(1)
aspect_ratio	u(4)
frame_rate_code	u(4)
marker_bit	f(1)
bit_rate_lower	u(18)
marker_bit	f(1)
bit_rate_upper	u(12)
low_delay	u(1)
temporal_id_enable_flag	u(1)
marker_bit	f(1)
bbv_buffer_size	u(18)
marker_bit	f(1)
max_dpb_minus1	u(4)
rpl1_index_exist_flag	u(1)
rpl1_same_as_rpl0_flag	u(1)
marker_bit	f(1)
num_ref_pic_list_set[0]	ue(v)
for (j = 0; j < NumRefPicListSet[0]; j++) {	
reference_picture_list_set(0, j)	
}	
if (! Rpl1SameAsRpl0Flag) {	
num_ref_pic_list_set[1]	ue(v)
for (j = 0; j < NumRefPicListSet[1]; j++) {	
reference_picture_list_set(1, j)	
}	
}	
num_ref_default_active_minus1[0]	ue(v)
num_ref_default_active_minus1[1]	ue(v)
log2_lcu_size_minus2	u(3)
log2_min_cu_size_minus2	u(2)
log2_max_part_ratio_minus2	u(2)
max_split_times_minus6	u(3)

Definition of sequence header	Descriptor
log2_min_qt_size_minus2	u(3)
log2_max_bt_size_minus2	u(3)
log2_max_eqt_size_minus3	u(2)
marker_bit	f(1)
weight_quant_enable_flag	u(1)
if (WeightQuantEnableFlag)	
{	
load_seq_weight_quant_data_flag	u(1)
if (load_seq_weight_quant_data_flag == "1") {	
weight_quant_matrix()	
}	
}	
st_enable_flag	u(1)
sao_enable_flag	u(1)
alf_enable_flag	u(1)
affine_enable_flag	u(1)
smvd_enable_flag	u(1)
ipcm_enable_flag	u(1)
amvr_enable_flag	u(1)
num_of_hmvp_cand	u(4)
umve_enable_flag	u(1)
if ((NumOfHmvpCand != 0) && AmvrEnableFlag) {	
emvr_enable_flag	u(1)
}	
intra_pf_enable_flag	u(1)
tscpm_enable_flag	u(1)
marker_bit	f(1)
dt_enable_flag	u(1)
if (DtEnableFlag) {	
log2_max_dt_size_minus4	u(2)
}	
pbt_enable_flag	u(1)
EipmEnableFlag = 0	
MipfEnableFlag = 0	
Intra PfChromaEnableFlag = 0	
UmveEnhancementEnableFlag = 0	
AffineUmveEnableFlag = 0	
SbTmvpEnableFlag = 0	
SrccEnableFlag = 0	
EnhancedStEnableFlag = 0	
EnhancedTscpmEnableFlag = 0	
MaecEnableFlag = 0	
if (profile_id == 0x30 profile_id == 0x32) {	
EipmEnableFlag = 1	

Definition of sequence header	Descriptor
pmc_enable_flag	u(1)
if (TscpmEnableFlag) {	
EnhancedTscpmEnableFlag = 1	
}	
iip_enable_flag	u(1)
sawp_enable_flag	u(1)
MipfEnableFlag = 1	
if (Intra PfEnableFlag) {	
Intra PfChromaEnableFlag = 1	
}	
if (UmveEnableFlag) {	
UmveEnhancementEnableFlag = 1	
}	
if (AffineEnableFlag) {	
AffineUmveEnableFlag = 1	
}	
if (AffineEnableFlag) {	
asr_enable_flag	u(1)
}	
awp_enable_flag	u(1)
SbTmvpEnableFlag = 1	
etmvp_mvap_enable_flag	u(1)
dmvr_enable_flag	u(1)
bio_enable_flag	u(1)
bgc_enable_flag	u(1)
inter_pf_enable_flag	u(1)
inter_pc_enable_flag	u(1)
obmc_enable_flag	u(1)
if (StEnableFlag) {	
EnhancedStEnableFlag = 1	
}	
sbt_enable_flag	u(1)
ist_enable_flag	u(1)
SrccEnableFlag = 1	
MaecEnableFlag = 1	
esao_enable_flag	u(1)
ccsao_enable_flag	u(1)
if (EsaoEnableFlag) {	
SaoEnableFlag = 0	
}	
if (AlfEnableFlag) {	
ealf_enable_flag	u(1)
}	
ibc_enable_flag	u(1)

Definition of sequence header	Descriptor
marker_bit	f(1)
isc_enable_flag	u(1)
if (IbcEnableFlag IscEnableFlag) {	
num_of_intra_hmvp_cand	u(4)
}	
fimc_enable_flag	u(1)
nn_tools_set_hook	u(8)
if (NnFilterEnableFlag)	
{	
num_of_nn_filter_minus1	ue(v)
}	
marker_bit	f(1)
}	
if (low_delay == “0”)	
output_reorder_delay	u(5)
cross_patch_loop_filter_enable_flag	u(1)
ref_colocated_patch_flag	u(1)
stable_patch_flag	u(1)
if (stable_patch_flag == “1”) {	
uniform_patch_flag	u(1)
if (uniform_patch_flag == “1”) {	
marker_bit	f(1)
patch_width_minus1	ue(v)
patch_height_minus1	ue(v)
}	
}	
reserved_bits	r(2)
next_start_code()	
}	

6.1.2.3 Definition of reference picture list configuration set

The definition of a reference picture list configuration set is shown in Table 15.

Table 15—Definition of reference picture list configuration set

Definition of reference picture list configuration set	Descriptor
reference_picture_list_set(list, rpls) {	
if (LibraryPictureEnableFlag) {	
reference_to_library_enable_flag	u(1)
}	
num_of_ref_pic[list][rpls]	ue(v)
for (i = 0; i < NumOfRefPic[list][rpls]; i++) {	
if (ReferenceToLibraryEnableFlag) {	
library_index_flag[list][rpls][i]	u(1)
}	

Definition of reference picture list configuration set	Descriptor
if (LibraryIndexFlag[list][rpls][i]) {	
referenced_library_picture_index [list][rpls][i]	ue(v)
}	
else {	
abs_delta_doi [list][rpls][i]	ue(v)
if (abs_delta_doi[list][rpls][i] > 0) {	
sign_delta_doi [list][rpls][i]	u(1)
}	
}	
}	
}	

6.1.2.4 Definition of weighted quantization matrix

The definition of a weighted quantization matrix is shown in Table 16.

Table 16—Definition of weighted quantization matrix

Definition of weighted quantization matrix	Descriptor
weight_quant_matrix() {	
for (sizeId = 0; sizeId < 2; sizeId++) {	
WQMSize = 1 << (sizeId+2)	
for (i = 0; i < WQMSize; i++) {	
for (j = 0; j < WQMSize; j++) {	
weight_quant_coeff	ue(v)
if (sizeId == 0)	
WeightQuantMatrix _{4x4} [i][j] = WeightQuantCoeff	
else	
WeightQuantMatrix _{8x8} [i][j] = WeightQuantCoeff	
}	
}	
}	
}	

6.1.2.5 Extension and user data syntax

The definition of extension and user data, definition of extension data, and definition of user data are shown in Table 17, Table 18, and Table 19, respectively.

Table 17—Definition of extension and user data

Definition of extension and user data	Descriptor
extension_and_user_data(i) {	
while ((next_bits(32) == extension_start_code) (next_bits(32) == user_data_start_code)) {	
if (next_bits(32) == extension_start_code)	
extension_data(i)	
if (next_bits(32) == user_data_start_code)	

Definition of extension and user data	Descriptor
user_data()	
}	
}	

Table 18—Definition of extension data

Definition of extension data	Descriptor
extension_data(i) {	
while (next_bits(32) == extension_start_code) {	
extension_start_code	f(32)
if (i == 0) { /* after sequence header */	
if (next_bits(4) == "0010") /* sequence display extension */	
sequence_display_extension()	
else if (next_bits(4) == "0011") /* temporal scalability extension */	
temporal_scalability_extension()	
else if (next_bits(4) == "0100") /* copyright extension */	
copyright_extension()	
else if (next_bits(4) == "1010") /* mastering display and content metadata extension */	
mastering_display_and_content_metadata_extension()	
else if (next_bits(4) == "1011") /* camera parameters extension */	
camera_parameters_extension()	
else if (next_bits(4) == "1101") /* reference to library picture extension */	
cross_random_access_point_reference_extension()	
else	
while (next_bits(24) != "0000 0000 0000 0000 0000 0001")	
reserved_extension_data_byte	u(8)
}	
}	
else { /* after picture header */	
if (next_bits(4) == "0100") /* copyright extension */	
copyright_extension()	
else if (next_bits(4) == "0101") /* high dynamic range picture metadata extension */	
hdr_dynamic_metadata_extension()	
else if (next_bits(4) == "0111") /* picture display extension */	
picture_display_extension()	
else if (next_bits(4) == "1011") /* camera parameters extension */	
camera_parameters_extension()	
else if (next_bits(4) == "1100") /* ROI parameters extension */	
roi_parameters_extension()	
else if (next_bits(4) == "1110") /* adaptive intra refresh parameters */	
air_parameters_extension()	
else {	
while (next_bits(24) != "0000 0000 0000 0000 0000 0001")	

Definition of extension data	Descriptor
reserved_extension_data_byte	u(8)
}	
}	
}	
}	

Table 19—Definition of user data

Definition of user data	Descriptor
user_data() {	
user_data_start_code	f(32)
while (next_bits(24) != “0000 0000 0000 0000 0000 0001”) {	
user_data	b(8)
}	
}	

6.1.2.6 Definition of sequence display extension

The definition of a sequence display extension is shown in Table 20.

Table 20—Definition of sequence display extension

Definition of sequence display extension	Descriptor
sequence_display_extension() {	
extension_id	f(4)
video_format	u(3)
sample_range	u(1)
color_description	u(1)
if (color_description) {	
color_primaries	u(8)
transfer_characteristics	u(8)
matrix_coefficients	u(8)
}	
display_horizontal_size	u(14)
marker_bit	f(1)
display_vertical_size	u(14)
td_mode_flag	u(1)
if (td_mode_flag == “1”) {	
td_packing_mode	u(8)
view_reverse_flag	u(1)
}	
next_start_code()	
}	

6.1.2.7 Definition of temporal scalability extension

The definition of a temporal scalability extension is shown in Table 21.

Table 21—Definition of temporal scalability extension

Definition of temporal scalability extension	Descriptor
temporal_scalability_extension() {	
extension_id	f(4)
num_of_temporal_level_minus1	u(3)
for (i = 0; i < num_of_temporal_level_minus1; i++) {	
temporal_frame_rate_code[i]	u(4)
temporal_bit_rate_lower[i]	u(18)
marker_bit	f(1)
temporal_bit_rate_upper[i]	u(12)
}	
next_start_code()	
}	

6.1.2.8 Definition of copyright extension

The definition of a copyright extension is shown in Table 22.

Table 22—Definition of copyright extension

Definition of copyright extension	Descriptor
copyright_extension() {	
extension_id	f(4)
copyright_flag	u(1)
copyright_id	u(8)
original_or_copy	u(1)
reserved_bits	r(7)
marker_bit	f(1)
copyright_number_1	u(20)
marker_bit	f(1)
copyright_number_2	u(22)
marker_bit	f(1)
copyright_number_3	u(22)
next_start_code()	
}	

6.1.2.9 Definition of high dynamic range picture extension

The definition of a high dynamic range picture metadata extension is shown in Table 23.

Table 23—Definition of high dynamic range picture extension

Definition of high dynamic range picture metadata extension	Descriptor
hdr_dynamic_metadata_extension() {	
extension_id	f(4)
hdr_dynamic_metadata_type	u(4)
while (next_bits(24) != “0000 0000 0000 0000 0000 0001”) {	
extension_data_byte	u(8)
}	
next_start_code()	
}	

6.1.2.10 Definition of mastering display and content metadata extension

The definition of a mastering display and content metadata extension is shown in Table 24.

Table 24—Definition of mastering display and content metadata extension

Definition of mastering display and content metadata extension	Descriptor
mastering_display_and_content_metadata_extension() {	
extension_id	f(4)
for (c = 0; c<3; c++) {	
display primaries_x[c]	u(16)
marker_bit	f(1)
display primaries_y[c]	u(16)
marker_bit	f(1)
}	
white_point_x	u(16)
marker_bit	f(1)
white_point_y	u(16)
marker_bit	f(1)
max_display_mastering_luminance	u(16)
marker_bit	f(1)
min_display_mastering_luminance	u(16)
marker_bit	f(1)
max_content_light_level	u(16)
marker_bit	f(1)
max_picture_average_light_level	u(16)
marker_bit	f(1)
reserved_bits	r(16)
next_start_code()	
}	

6.1.2.11 Definition of camera parameters extension

The definition of a camera parameters extension is shown in Table 25.

Table 25—Definition of camera parameters extension

Definition of camera parameters extension	Descriptor
camera_parameters_extension() {	
extension_id	f(4)
reserved_bits	r(1)
camera_id	u(7)
marker_bit	f(1)
height_of_image_device	u(22)
marker_bit	f(1)
focal_length	u(22)
marker_bit	f(1)
f_number	u(22)
marker_bit	f(1)
vertical_angle_of_view	u(22)
marker_bit	f(1)
camera_position_x_upper	i(16)
marker_bit	f(1)
camera_position_x_lower	i(16)
marker_bit	f(1)
camera_position_y_upper	i(16)
marker_bit	f(1)
camera_position_y_lower	i(16)
marker_bit	f(1)
camera_position_z_upper	i(16)
marker_bit	f(1)
camera_position_z_lower	i(16)
marker_bit	f(1)
camera_direction_x	i(22)
marker_bit	f(1)
camera_direction_y	i(22)
marker_bit	f(1)
camera_direction_z	i(22)
marker_bit	f(1)
image_plane_vertical_x	i(22)
marker_bit	f(1)
image_plane_vertical_y	i(22)
marker_bit	f(1)
image_plane_vertical_z	i(22)
marker_bit	f(1)
reserved_bits	r(16)
next_start_code()	
}	

6.1.2.12 Definition of ROI parameters extension

The definition of an ROI parameters extension is shown in Table 26.

Table 26—Definition of ROI parameters extension

Definition of ROI parameters extension	Descriptor
roi_parameters_extension() {	
extension_id	f(4)
current_picture_roi_num	u(8)
roiIndex = 0	
if (PictureType != 0) {	
prev_picture_roi_num	u(8)
for (i = 0; i < PrevPictureROINum; i++) {	
roi_skip_run	ue(v)
if (roi_skip_run != "0") {	
for (j = 0; j < roi_skip_run; j++) {	
skip_roi_mode[i+j]	u(1)
if (j % 22 == 0) {	
marker_bit	f(1)
}	
if (skip_roi_mode == "1") {	
ROIInfo[roiIndex]->asisx = PrevROIInfo[i+j] ->asisx	
ROIInfo[roiIndex]->asisy = PrevROIInfo[i+j] ->asisy	
ROIInfo[roiIndex]->width = PrevROIInfo[i+j] -> width	
ROIInfo[roiIndex]->height = PrevROIInfo[i+j] -> height	
roiIndex++	
}	
}	
i += j	
marker_bit	f(1)
}	
else {	
roi_axisx_delta	se(v)
marker_bit	f(1)
roi_axisy_delta	se(v)
marker_bit	f(1)
roi_width_delta	se(v)
marker_bit	f(1)
roi_height_delta	se(v)
marker_bit	f(1)
ROIInfo[roiIndex]->asisx = PrevROIInfo[i+j] ->asisx + ROIAxisxDelta	
ROIInfo[roiIndex]->asisy = PrevROIInfo[i+j] ->asisy + ROIAxisyDelta	
ROIInfo[roiIndex]->width = PrevROIInfo[i+j] -> width + ROIWidthDelta	
ROIInfo[roiIndex]->height = PrevROIInfo[i+j] -> height + ROIHeightDelta	
roiIndex++	
}	
}	
}	
for (i = roiIndex; i < current_picture_roi_num; i++) {	

Definition of ROI parameters extension	Descriptor
roi_axisx	u(6)
marker_bit	f(1)
roi_axisy	u(6)
marker_bit	f(1)
roi_width	u(6)
marker_bit	f(1)
roi_height	u(6)
marker_bit	f(1)
ROIInfo[roiIndex]->asisx = roi_axisx	
ROIInfo[roiIndex]->asisy = roi_axisy	
ROIInfo[roiIndex]->width = roi_width	
ROIInfo[roiIndex]->height = roi_height	
roiIndex++	
}	
for (i = 0; i < roiIndex; i++) {	
PrevROIInfo[i]->asisx = ROIInfo[i]->asisx	
PrevROIInfo[i]->asisy = ROIInfo[i]->asisy	
PrevROIInfo[i]->width = ROIInfo[i]->width	
PrevROIInfo[i]->height = ROIInfo[i]->height	
}	
next_start_code()	
}	

6.1.2.13 Definition of reference to library picture extension

The definition of a reference to library picture extension is shown in Table 27.

Table 27—Definition of reference to library picture extension

Definition of reference to library picture extension	Descriptor
cross_random_access_point_reference_extension() {	
extension_id	f(4)
crr_lib_number	u(3)
marker_bit	f(1)
i = 0	
while (i < crr_lib_number) {	
crr_lib_pid[i]	u(9)
i++	
if (i%2 == 0) {	
marker_bit	f(1)
}	
}	
next_start_code()	
}	

6.1.2.14 Definition of adaptive intra refresh parameter extension

The definition of an adaptive intra refresh parameters extension is shown in Table 28.

Table 28—Definition of adaptive intra refresh parameters extension

Definition of adaptive intra refresh parameters extension	Descriptor
air_parameters_extension() {	
extension_id	f(4)
air_bound_x	u(10)
air_bound_y	u(10)
next_start_code()	
}	

6.1.3 Definition of picture

6.1.3.1 Definition of intra picture header

The definition of an intra picture header is shown in Table 29.

Table 29—Definition of intra picture header

Definition of intra picture header	Descriptor
intra_picture_header() {	
intra_picture_start_code	f(32)
bbv_delay	u(32)
time_code_flag	u(1)
if (time_code_flag == “1”)	
time_code	u(24)
decode_order_index	u(8)
if (LibraryStreamFlag)	
library_picture_index	ue(v)
if (temporal_id_enable_flag == “1”)	
temporal_id	u(3)
if (low_delay == “0”)	
picture_output_delay	ue(v)
if (low_delay == “1”)	
bbv_check_times	ue(v)
progressive_frame	u(1)
if (progressive_frame == “0”)	
picture_structure	u(1)
top_field_first	u(1)
repeat_first_field	u(1)
if (field_coded_sequence == “1”) {	
top_field_picture_flag	u(1)
reserved_bits	r(1)
}	
ref_pic_list_set_flag[0]	u(1)
if (RefPicListSetFlag[0]) {	