**COMMITTEE DRAFT AMENDMENT****© ISO/IEC 2022 – All rights reserved****Text of ISO/IEC 23002-7:202x/CDAM 1****63****Part 7: Versatile supplemental enhancement information messages for coded video bitstreams, AMENDMENT 1: Additional SEI messages****Information technology — MPEG video technologies****Élément introductif — Élément central — Partie 7: Titre de la partie****Information technology — MPEG video technologies — Part 7: Versatile supplemental enhancement information messages for coded video bitstreams, AMENDMENT 1: Additional SEI messages****E****2022-08-13****(30) Committee****ISO/IEC****ISO/IEC J****202x****1****Amendment****International Standard****202x****141****ISO/IEC 23002‑****ISO/IEC 23002‑7****ISO/IEC 23002-7:202x/CDAM 1****JISC****Coding of audio, picture, multimedia and hypermedia information****Information technology****5****29****1** **2****見出し 2****見出し 1****0****2****STD Version 2.1c2****30** **4** **ISO/IEC JTC 1/SC 29 /WG 5 N 141**

Date: **2022-08-13**

**Text of ISO/IEC 23002-7:202x/CDAM 1**

ISO/IEC JTC 1/SC 29/WG 5

Secretariat:  JISC

**Information technology — MPEG video technologies — Part 7: Versatile supplemental enhancement information messages for coded video bitstreams, AMENDMENT 1: Additional SEI messages**

*Élément introductif — Élément central — Partie 7: Titre de la partie*

|  |
| --- |
| **Warning**  This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.  Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation |

**Copyright notice**

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

[Indicate the full address, telephone number, fax number, telex number, and electronic mail address, as appropriate, of the Copyright Manger of the ISO member body responsible for the secretariat of the TC or SC within the framework of which the working document has been prepared.]

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Foreword

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

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

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

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see [www.iso.org/iso/foreword.html](https://www.iso.org/foreword-supplementary-information.html).

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*, in collaboration with ITU-T. A technically aligned twin text is published as ITU-T H.274.

A list of all parts in the ISO/IEC 23002 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html).

Information technology — MPEG video technologies — Part 7: Versatile supplemental enhancement information messages for coded video bitstreams, AMENDMENT 1: Additional SEI messages

*Clause 2*

Add the following references:

ISO/IEC 15938-17, Information technology – Multimedia content description interface – Part 17: Compression of neural networks for multimedia content description and analysis.

IETF Standard 66, Uniform Resource Identifiers (URI): Generic Syntax, <http://tools.ietf.org/html/rfc3986>

IETF RFC 4151, The 'tag' URI Scheme, October 2005, <http://tools.ietf.org/html/rfc4151>

*Subclause 5.8*

Add the following function definitions in alphabetical order and adjust the intermediate and subsequent formula indices accordingly:

Reflect( y, z ) = (11)

Wrap( y,z ) = (17)

*Subclause 8.1*

Replace Table 4 with the following:

|  |  |
| --- | --- |
| **SEI message** | **Persistence scope** |
| Filler payload | The PU containing the SEI message |
| User data registered by Rec. ITU-T T.35 | Unspecified |
| User data unregistered | Unspecified |
| Film grain characteristics | Specified by the syntax of the SEI message |
| Frame packing arrangement | Specified by the syntax of the SEI message |
| Display orientation | Specified by the syntax of the SEI message |
| Referenced parameter sets | The CLVS containing the SEI message |
| Decoded picture hash | The PU containing the SEI message |
| Mastering display colour volume | The CLVS containing the SEI message |
| Colour transform information | Specified by the syntax of the SEI message |
| Content light level information | The CLVS containing the SEI message |
| DRAP indication | The picture associated with the SEI message |
| Alternative transfer characteristics | The CLVS containing the SEI message |
| Ambient viewing environment | The CLVS containing the SEI message |
| Content colour volume | Specified by the syntax of the SEI message |
| Equirectangular projection | Specified by the syntax of the SEI message |
| Generalized cubemap projection | Specified by the syntax of the SEI message |
| Sphere rotation | Specified by the syntax of the SEI message |
| Region-wise packing | Specified by the syntax of the SEI message |
| Omnidirectional viewport | Specified by the syntax of the SEI message |
| Alpha channel information | Specified by the syntax of the SEI message |
| Frame-field information | The PU containing the SEI message |
| Depth representation information | Specified by the semantics of the SEI message |
| Multiview acquisition information | The CVS containing the SEI message |
| Multiview view position | The CVS containing the SEI message |
| Annotated regions | Specified by the syntax of the SEI message |
| Sample aspect ratio information | Specified by the syntax of the SEI message |
| Scalability dimension information | The CVS containing the SEI message |
| Extended DRAP indication | The picture associated with the SEI message |
| Shutter interval information | The CVS containing the SEI message |
| Neural-network post-filter characteristics | The CVS containing the SEI message |
| Neural-network post-filter activation | The PU containing the SEI message |
| Phase indication | Specified by the semantics of the SEI message |

*New subclauses 8.27 to 8.30*

Add subclauses 8.27 to 8.30 as follows:

* 1. **Shutter interval information SEI message**

**8.27.1 Shutter interval information SEI message syntax**

|  |  |
| --- | --- |
| shutter\_interval\_info( payloadSize ) { | **Descriptor** |
| **sii\_time\_scale** | u(32) |
| **fixed\_shutter\_interval\_within\_clvs\_flag** | u(1) |
| if( fixed\_shutter\_interval\_within\_clvs\_flag ) |  |
| **sii\_num\_units\_in\_shutter\_interval** | u(32) |
| else { |  |
| **sii\_max\_sub\_layers\_minus1** | u(3) |
| for( i = 0; i <= sii\_max\_sub\_layers\_minus1; i++ ) |  |
| **sub\_layer\_num\_units\_in\_shutter\_interval**[ i ] | u(32) |
| } |  |
| } |  |

**8.27.2 Shutter interval information SEI message syntax**

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

**sii\_time\_scale** specifies the number of time units that pass in one second. The value of sii\_time\_scale shall not be equal to 0. For example, a time coordinate system that measures time using a 27 MHz clock has an sii\_time\_scale of 27 000 000.

**fixed\_shutter\_interval\_within\_clvs\_flag** equal to 1 specifies that the indicated shutter interval is the same for all temporal sublayers in the CLVS. fixed\_shutter\_interval\_within\_clvs\_flagequal to 0 specifies that the indicated shutter interval may not be the same for all temporal sublayers in the CLVS.

**sii\_num\_units\_in\_shutter\_interval**, when fixed\_shutter\_interval\_within\_clvs\_flag is equal to 1, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable shutterInterval, in units of seconds, is equal to the quotient of sii\_num\_units\_in\_shutter\_interval divided by sii\_time\_scale. For example, to represent a shutter interval equal to 0.04 seconds, sii\_time\_scale may be equal to 27 000 000 and sii\_num\_units\_in\_shutter\_interval may be equal to 1 080 000.

**sii\_max\_sub\_layers\_minus1** plus 1 specifies the maximum number of temporal sublayers that may be present in each CLVS referring to the SPS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2022-03 Annex D when sii\_max\_sub\_layers\_minus1 is equal to 1 and fixed\_shutter\_interval\_within\_clvs\_flag is equal to 0.

**sub\_layer\_num\_units\_in\_shutter\_interval**[ i ], when present, specifies the number of time units of a clock operating at the frequency sii\_time\_scale Hz that corresponds to the shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS. The shutter interval for each picture with temporal sublayer identifier equal to i in the CLVS, denoted by the variable subLayerShutterInterval[ i ], in units of seconds, is equal to the quotient of sub\_layer\_num\_units\_in\_shutter\_interval[ i ] divided by sii\_time\_scale.

The variable subLayerShutterInterval[ i ], corresponding to the indicated shutter interval of each picture with temporal sublayer identifier equal to i in the CLVS, is thus derived as follows:

if( fixed\_shutter\_interval\_within\_clvs\_flag )  
 subLayerShutterInterval[ i ] = sii\_num\_units\_in\_shutter\_interval ÷ sii\_time\_scale (75)  
else  
 subLayerShutterInterval[ i ] = sub\_layer\_num\_units\_in\_shutter\_interval[ i ] ÷ sii\_time\_scale

* 1. **Neural-network post-filter characteristics SEI message**

**8.28.1 Neural-network post-filter characteristics SEI message syntax**

|  |  |
| --- | --- |
| nn\_post\_filter\_characteristics( payloadSize ) { | **Descriptor** |
| **nnpfc\_id** | ue(v) |
| **nnpfc\_mode\_idc** | ue(v) |
| **nnpfc\_purpose\_and\_formatting\_flag** | u(1) |
| if( nnpfc\_purpose\_and\_formatting\_flag ) { |  |
| **nnpfc\_purpose** | ue(v) |
| if( nnpfc\_purpose = = 2 | | nnpfc\_purpose = = 4 ) |  |
| **nnpfc\_out\_sub\_c\_flag** | u(1) |
|  |  |
| if( nnpfc\_purpose = = 3 | | nnpfc\_purpose = = 4 ) { |  |
| **nnpfc\_pic\_width\_in\_luma\_samples** | ue(v) |
| **nnpfc\_pic\_height\_in\_luma\_samples** | ue(v) |
| } |  |
| /\* input and output formatting \*/ |  |
| **nnpfc\_component\_last\_flag** | u(1) |
| **nnpfc\_inp\_format\_flag** | u(1) |
| if( nnpfc\_inp\_format\_flag = = 1 ) |  |
| **nnpfc\_inp\_tensor\_bitdepth\_minus8** | ue(v) |
| **nnpfc\_inp\_order\_idc** | ue(v) |
| **nnpfc\_auxiliary\_inp\_idc** | ue(v) |
| **nnpfc\_separate\_colour\_description\_present\_flag** | u(1) |
| if( nnpfc\_separate\_colour\_description\_present\_flag ) { |  |
| **nnpfc\_colour\_primaries** | u(8) |
| **nnpfc\_transfer\_characteristics** | u(8) |
| **nnpfc\_matrix\_coeffs** | u(8) |
| } |  |
| **nnpfc\_out\_format\_flag** | u(1) |
| if( nnpfc\_out\_format\_flag = = 1 ) |  |
| **nnpfc\_out\_tensor\_bitdepth\_minus8** | ue(v) |
| **nnpfc\_out\_order\_idc** | ue(v) |
| **nnpfc\_constant\_patch\_size\_flag** | u(1) |
| **nnpfc\_patch\_width\_minus1** | ue(v) |
| **nnpfc\_patch\_height\_minus1** | ue(v) |
| **nnpfc\_overlap** | ue(v) |
| **nnpfc\_padding\_type** | ue(v) |
| if( nnpfc\_padding\_type = = 4 ){ |  |
| **nnpfc\_luma\_padding\_val** | ue(v) |
| **nnpfc\_cb\_padding\_val** | ue(v) |
| **nnpfc\_cr\_padding\_val** | ue(v) |
| } |  |
| **nnpfc\_complexity\_idc** | ue(v) |
| if( nnpfc\_complexity\_idc > 0 ) |  |
| nnpfc\_complexity\_element( nnpfc\_complexity\_idc ) |  |
| if( nnpfc\_mode\_idc = = 2 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **nnpfc\_reserved\_zero\_bit** | u(1) |
| **nnpfc\_uri\_tag**[ i ] | st(v) |
| **nnpfc\_uri**[ i ] | st(v) |
| } |  |
| } |  |
| /\* filter specified or updated by ISO/IEC 15938-17 bitstream \*/ |  |
| if( nnpfc\_mode\_idc = = 1 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **nnpfc\_reserved\_zero\_bit** | u(1) |
| for( i = 0; more\_data\_in\_payload( ); i++ ) |  |
| **nnpfc\_payload\_byte**[ i ] | b(8) |
| } |  |
| } |  |

|  |  |
| --- | --- |
| nnpfc\_complexity\_element( nnpfc\_complexity\_idc ) { | **Descriptor** |
| if( nnpfc\_complexity\_idc = = 1 ) { |  |
| **nnpfc\_parameter\_type\_idc** | u(2) |
| if (nnpfc\_parameter\_type\_idc ! = 2) |  |
| **nnpfc\_log2\_parameter\_bit\_length\_minus3** | u(2) |
| **nnpfc\_num\_parameters\_idc** | u(6) |
| **nnpfc\_num\_kmac\_operations\_idc** | ue(v) |
| } |  |
| } |  |

* + 1. **Neural-network post-filter characteristics SEI message semantics**

This SEI message specifies a neural network that may be used as a post-processing filter. The use of specified post-processing filters for specific pictures is indicated with neural-network post-filter activation SEI messages.

Use of this SEI message requires the definition of the following variables:

– Cropped decoded output picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

– Luma sample array CroppedYPic and chroma sample arrays CroppedCbPic and CroppedCrPic, when present, of the cropped decoded output picture for vertical coordinates y and horizontal coordinates x, where the top-left corner of the sample array has coordinates y equal to 0 and x equal to 0.

– Bit depth BitDepthY for the luma sample array of the cropped decoded output picture.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the cropped decoded output picture.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– When nnpfc\_auxiliary\_inp\_idc is equal to 1, a quantization strength value StrengthControlVal.

When this SEI message specifies a neural network that may be used as a post-processing filter, the semantics specify the derivation of the luma sample array FilteredYPic[ x ][ y ] and chroma sample arrays FilteredCbPic[ x ][ y ] and FilteredCrPic[ x ][ y ], as indicated by the value of nnpfc\_out\_order\_idc, that contain the output of the post-processing filter.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**nnpfc\_id** contains an identifying number that may be used to identify a post-processing filter. The value of nnpfc\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of nnpfc\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of nnpfc\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore it.

**nnpfc\_mode\_idc** equal to 0 specifies that the post-processing filter associated with the nnpfc\_id value is determined by external means not specified in this document.

nnpfc\_mode\_idc equal to 1 specifies that the post-processing filter associated with the nnpfc\_id value is a neural network represented by the ISO/IEC 15938-17 bitstream contained in this SEI message.

nnpfc\_mode\_idc equal to 2 specifies that the post-processing filter associated with the nnpfc\_id value is a neural network identified by a specified tag Uniform Resource Identifier (URI) (nnpfc\_uri\_tag[ i ]) and neural network information URI (nnpfc\_uri[ i ]).

The value of nnpfc\_mode\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_mode\_idc greater than 2 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_mode\_idc.

**nnpfc\_purpose\_and\_formatting\_flag** equal to 0 specifies that no syntax elements related to the filter purpose, input formatting, output formatting, and complexity are present. nnpfc\_purpose\_and\_formatting\_flag equal to 1 specifies that syntax elements related to the filter purpose, input formatting, output formatting, and complexity are present.

When nnpfc\_mode\_idc is equal to 1 and the current CLVS does not contain a preceding neural-network post-filter characteristics SEI message, in decoding order, that has the value of nnpfc\_id equal to the value of nnpfc\_id in this SEI message, nnpfc\_purpose\_and\_formatting\_flag shall be equal to 1.

When the current CLVS contains a preceding neural-network post-filter characteristics SEI message, in decoding order, that has the same value of nnpfc\_id equal to the value of nnpfc\_id in this SEI message, at least one of the following conditions shall apply:

– This SEI message has nnpfc\_mode\_idc equal to 1 and nnpfc\_purpose\_and\_formatting\_flag equal to 0 in order to provide a neural network update.

– This SEI message has the same content as the preceding neural-network post-filter characteristics SEI message.

When this SEI message is the first neural-network post-filter characteristics SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS, it specifies a base post-processing filter that pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS. When this SEI message is not the first neural-network post-filter characteristics SEI message, in decoding order, that has a particular nnpfc\_id value within the current CLVS, this SEI message pertains to the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS or the next neural-network post-filter characteristics SEI message having that particular nnpfc\_id value, in output order, within the current CLVS.

**nnpfc\_purpose** indicates the purpose of post-processing filter as specified in Table 20. The value of nnpfc\_purpose shall be in the range of 0 to 232 − 2, inclusive. Values of nnpfc\_purpose that do not appear in Table 20 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_purpose.

**Table 20 – Definition of nnpfc\_purpose**

|  |  |
| --- | --- |
| **Value** | **Interpretation** |
| 0 | Unknown or unspecified |
| 1 | Visual quality improvement |
| 2 | Chroma upsampling from the 4:2:0 chroma format to the 4:2:2 or 4:4:4 chroma format, or from the 4:2:2 chroma format to the 4:4:4 chroma format |
| 3 | Increasing the width or height of the cropped decoded output picture without changing the chroma format |
| 4 | Increasing the width or height of the cropped decoded output picture and upsampling the chroma format |

NOTE 1 – When a reserved value of nnpfc\_purpose is taken into use in the future by ITU-T | ISO/IEC, the syntax of this SEI message could be extended with syntax elements whose presence is conditioned by nnpfc\_purpose being equal to that value.

When SubWidthC is equal to 1 and SubHeightC is equal to 1, nnpfc\_purpose shall not be equal to 2 or 4.

**nnpfc\_out\_sub\_c\_flag** equal to 1 specifies that outSubWidthC is equal to 1 and outSubHeightC is equal to 1. nnpfc\_out\_sub\_c\_flag equal to 0 specifies that outSubWidthC is equal to 2 and outSubHeightC is equal to 1. When nnpfc\_out\_sub\_c\_flag is not present, outSubWidthC is inferred to be equal to SubWidthC and outSubHeightC is inferred to be equal to SubHeightC.If SubWidthC is equal to 2 and SubHeightC is equal to 1, nnpfc\_out\_sub\_c\_flag shall not be equal to 0.

**nnpfc\_pic\_width\_in\_luma\_samples** and **nnpfc\_pic\_height\_in\_luma\_samples** specify the width and height, respectively, of the luma sample array of the picture resulting by applying the post-processing filter identified by nnpfc\_id to a cropped decoded output picture. When nnpfc\_pic\_width\_in\_luma\_samples and nnpfc\_pic\_height\_in\_luma\_samples are not present, they are inferred to be equal to CroppedWidth and CroppedHeight, respectively.

**nnpfc\_component\_last\_flag** equal to 0 specifies that the second dimension in the input tensor inputTensor to the post-processing filter and the output tensor outputTensor resulting from the post-processing filter is used for the channel. nnpfc\_component\_last\_flag equal to 1 specifies that the last dimension in the input tensor inputTensor to the post-processing filter and the output tensor outputTensor resulting from the post-processing filter is used for the channel.

NOTE 2 – The first dimension in the input tensor and in the output tensor is used for the batch index, which is a practice in some neural network frameworks. While the semantics of this SEI message use batch size equal to 1, it is up to the post-processing implementation to determine the batch size used as input to the neural network inference.

NOTE 3 – A colour component is an example of a channel.

**nnpfc\_inp\_format\_flag** indicates the method of converting a sample value of the cropped decoded output picture to an input value to the post-processing filter. When nnpfc\_inp\_format\_flag is equal to 0, the input values to the post-processing filter are real numbers and the functions InpY( ) and InpC( ) are specified as follows:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) − 1 ) (76)

InpC( x )= x ÷ ( ( 1  <<  BitDepthC ) − 1 ) (77)

When nnpfc\_inp\_format\_flag is equal to 1, the input values to the post-processing filter are unsigned integer numbers and the functions InpY( ) and InpC( ) are specified as follows:

shiftY = BitDepthY − inpTensorBitDepth  
if( inpTensorBitDepth >= BitDepthY)  
 InpY( x ) = x  <<  ( inpTensorBitDepth − BitDepthY ) (78)  
else  
 InpY( x ) = Clip3(0, ( 1  <<  inpTensorBitDepth ) − 1, ( x + ( 1  <<  ( shiftY − 1 ) ) )  >>  shiftY )

shiftC = BitDepthC − inpTensorBitDepth  
if( inpTensorBitDepth >= BitDepthC )  
 InpC( x ) = x  <<  ( inpTensorBitDepth − BitDepthC ) (79)  
else  
 InpC( x ) = Clip3(0, ( 1  <<  inpTensorBitDepth ) − 1, ( x + ( 1  <<  ( shiftC − 1 ) ) )  >>  shiftC )

The variable inpTensorBitDepth is derived from the syntax element nnpfc\_inp\_tensor\_bitdepth\_minus8 as specified below.

**nnpfc\_inp\_tensor\_bitdepth\_minus8** plus 8 specifies the bit depth of luma sample values in the input integer tensor. The value of inpTensorBitDepth is derived as follows:

inpTensorBitDepth = nnpfc\_inp\_tensor\_bitdepth\_minus8 + 8(80)

It is a requirement of bitstream conformance that the value of nnpfc\_inp\_tensor\_bitdepth\_minus8 shall be in the range of 0 to 24, inclusive.

**nnpfc\_auxiliary\_inp\_idc** not equal to 0 specifies auxiliary input data is present in the input tensor of the neural-network post-filter. nnpfc\_auxiliary\_inp\_idc equal to 0 indicates that auxiliary input data is not present in the input tensor. nnpfc\_auxiliary\_inp\_idc equal to 1 specifies that auxiliary input data is derived as specified in Table 23. The value of nnpfc\_auxiliary\_inp\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_auxiliary\_inp\_idc greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_auxiliary\_inp\_idc.

**nnpfc\_separate\_colour\_description\_present\_flag** equal to 1 indicates that a distinct combination of colour primaries, transfer characteristics, and matrix coefficients for the picture resulting from the post-processing filter is specified in the SEI message syntax structure. nnfpc\_separate\_colour\_description\_present\_flag equal to 0 indicates that the combination of colour primaries, transfer characteristics, and matrix coefficients for the picture resulting from the post-processing filter is the same as indicated in VUI parameters for the CLVS.

**nnpfc\_colour\_primaries** has the same semantics as specified in subclause 7.3 for the vui\_colour\_primaries syntax element, except as follows:

– nnpfc\_colour\_primaries specifies the colour primaries of the picture resulting from applying the neural-network post-filter specified in the SEI message, rather than the colour primaries used for the CLVS.

– When nnpfc\_colour\_primaries is not present in the neural-network post-filter characteristics SEI message, the value of nnpfc\_colour\_primaries is inferred to be equal to vui\_colour\_primaries.

**nnpfc\_transfer\_characteristics** has the same semantics as specified in subclause 7.3 for the vui\_transfer\_characteristics syntax element, except as follows:

– nnpfc\_transfer\_characteristics specifies the transfer characteristics of the picture resulting from applying the neural-network post-filter specified in the SEI message, rather than the transfer characteristics used for the CLVS.

– When nnpfc\_transfer\_characteristics is not present in the neural-network post-filter characteristics SEI message, the value of nnpfc\_transfer\_characteristics is inferred to be equal to vui\_transfer\_characteristics.

**nnpfc\_matrix\_coeffs** has the same semantics as specified in subclause 7.3 for the vui\_matrix\_coeffs syntax element, except as follows:

– nnpfc\_matrix\_coeffs specifies the matrix coefficients of the picture resulting from applying the neural-network post-filter specified in the SEI message, rather than the matrix coefficients used for the CLVS.

– When nnpfc\_matrix\_coeffs is not present in the neural-network post-filter characteristics SEI message, the value of nnpfc\_matrix\_coeffs is inferred to be equal to vui\_matrix\_coeffs.

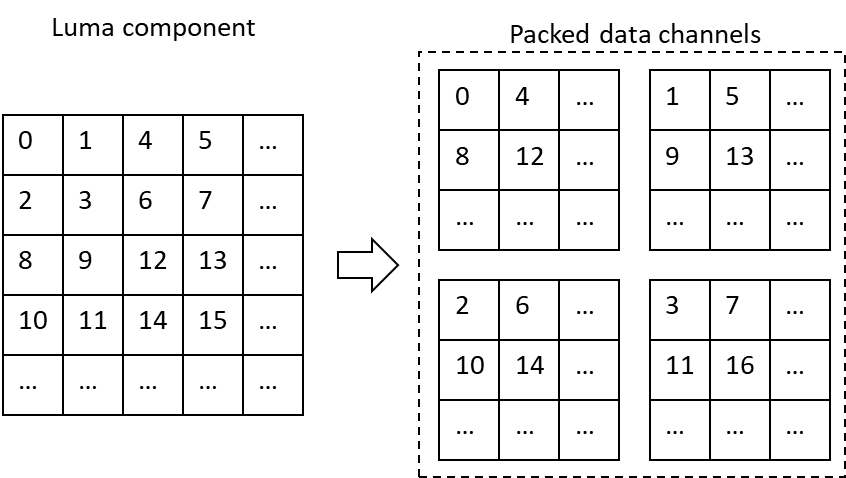
– The values allowed for nnpfc\_matrix\_coeffs are not constrained by the chroma format of the decoded video pictures that is indicated by the value of ChromaFormatIdc for the semantics of the VUI parameters.

– When nnpfc\_matrix\_coeffs is equal to 0, nnpfc\_out\_order\_idc shall not be equal to 1 or 3.

**nnpfc\_inp\_order\_idc** indicates the method of ordering the sample arrays of a cropped decoded output picture as the input to the post-processing filter. Table 21 contains an informative description of nnpfc\_inp\_order\_idc values. The semantics of nnpfc\_inp\_order\_idc in the range of 0 to 3, inclusive, are specified in Table 23, which specifies a process for deriving the input tensors inputTensor for different values of nnpfc\_inp\_order\_idc and a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors. When the chroma format of the cropped decoded output picture is not 4:2:0, nnpfc\_inp\_order\_idc shall not be equal to 3. The value of nnpfc\_inp\_order\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_inp\_order\_idc greater than 3 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_inp\_order\_idc.

**Table 21 – Informative description of nnpfc\_inp\_order\_idc values**

|  |  |
| --- | --- |
| **nnpfc\_inp\_ order\_idc** | **Description** |
| 0 | If nnpfc\_auxiliary\_inp\_idc is equal to 0, one luma matrix is present in the input tensor, thus the number of channels is 1. Otherwise, nnpfc\_auxiliary\_inp\_idc is not equal to 0 and one luma matrix and one auxiliary input matrix are present, thus the number of channels is 2. |
| 1 | If nnpfc\_auxiliary\_inp\_idc is equal to 0, two chroma matrices are present in the input tensor, thus the number of channels is 2. Otherwise, nnpfc\_auxiliary\_inp\_idc is not equal to 0 and two chroma matrices and one auxiliary input matrix are present, thus the number of channels is 3. |
| 2 | If nnpfc\_auxiliary\_inp\_idc is equal to 0, one luma and two chroma matrices are present in the input tensor, thus the number of channels is 3. Otherwise, nnpfc\_auxiliary\_inp\_idc is not equal to 0 and one luma matrix, two chroma matrices and one auxiliary input matrix are present, thus the number of channels is 4. |
| 3 | If nnpfc\_auxiliary\_inp\_idc is equal to 0, four luma matrices and two chroma matrices are present in the input tensor, thus the number of channels is 6. Otherwise, nnpfc\_auxiliary\_inp\_idc is not equal to 0 and four luma matrices, two chroma matrices, and one auxiliary input matrix are present in the input tensor, thus the number of channels is 7. The luma channels are derived in an interleaved manner as illustrated in Figure 12. This nnpfc\_inp\_order\_idc can only be used when the chroma format is 4:2:0. |
| 4..255 | Reserved |



**Figure 12 – Illustration of luma data channels of nnpfc\_inp\_order\_idc equal to 3 (informative)**

A patch is a rectangular array of samples from a component (e.g., a luma or chroma component) of a picture.

**nnpfc\_constant\_patch\_size\_flag** equal to 0 specifies that the post-processing filter accepts any patch size that is a positive integer multiple of the patch size indicated by nnpfc\_patch\_width\_minus1 and nnpfc\_patch\_height\_minus1 as input. When nnpfc\_constant\_patch\_size\_flag is equal to 0 the patch size width shall be less than or equal to CroppedWidth. When nnpfc\_constant\_patch\_size\_flag is equal to 0 the patch size height shall be less than or equal to CroppedHeight. nnpfc\_constant\_patch\_size\_flag equal to 1 specifies that the post-processing filter accepts exactly the patch size indicated by nnpfc\_patch\_width\_minus1 and nnpfc\_patch\_height\_minus1 as input.

**nnpfc\_patch\_width\_minus1** + 1, when nnpfc\_constant\_patch\_size\_flag equal to 1, specifies the horizontal sample counts of the patch size required for the input to the post-processing filter. When nnpfc\_constant\_patch\_size\_flag is equal to 0, any positive integer multiple of ( nnpfc\_patch\_width\_minus1 + 1 ) may be used as the horizontal sample counts of the patch size used for the input to the post-processing filter. The value of nnpfc\_patch\_width\_minus1 shall be in the range of 0 to Min( 32766, CroppedWidth − 1 ), inclusive.

**nnpfc\_patch\_height\_minus1** + 1, when nnpfc\_constant\_patch\_size\_flag equal to 1, specifies the vertical sample counts of the patch size required for the input to the post-processing filter. When nnpfc\_constant\_patch\_size\_flag is equal to 0, any positive integer multiple of ( nnpfc\_patch\_height\_minus1 + 1 ) may be used as the vertical sample counts of the patch size used for the input to the post-processing filter. The value of nnpfc\_patch\_height\_minus1 shall be in the range of 0 to Min( 32766, CroppedHeight − 1 ), inclusive.

**nnpfc\_overlap** specifies the overlapping horizontal and vertical sample counts of adjacent input tensors of the post-processing filter. The value of nnpfc\_overlap shall be in the range of 0 to 16383, inclusive.

The variables inpPatchWidth, inpPatchHeight, outPatchWidth, outPatchHeight, horCScaling, verCScaling, outPatchCWidth, outPatchCHeight, and overlapSize are derived as follows:

inpPatchWidth = nnpfc\_patch\_width\_minus1 + 1  
inpPatchHeight = nnpfc\_patch\_height\_minus1 + 1  
outPatchWidth = ( nnpfc\_pic\_width\_in\_luma\_samples \* inpPatchWidth ) / CroppedWidth  
outPatchHeight = ( nnpfc\_pic\_height\_in\_luma\_samples \* inpPatchHeight ) / CroppedHeight  
horCScaling = SubWidthC / outSubWidthC  
verCScaling = SubHeightC / outSubHeightC (81)  
outPatchCWidth = outPatchWidth \* horCScaling  
outPatchCHeight = outPatchHeight \* verCScaling  
overlapSize = nnpfc\_overlap

It is a requirement of bitstream conformance that outPatchWidth \* CroppedWidth shall be equal to nnpfc\_pic\_width\_in\_luma\_samples \* inpPatchWidth and outPatchHeight \* CroppedHeight shall be equal to nnpfc\_pic\_height\_in\_luma\_samples \* inpPatchHeight.

**nnpfc\_padding\_type** specifies the process of padding when referencing sample locations outside the boundaries of the cropped decoded output picture as described in Table 22. The value of nnpfc\_padding\_type shall be in the range of 0 to 15, inclusive.

**Table 22 – Informative description of nnpfc\_padding\_type values**

|  |  |
| --- | --- |
| **nnpfc\_padding\_type** | **Description** |
| 0 | zero padding |
| 1 | replication padding |
| 2 | reflection padding |
| 3 | wrap-around padding |
| 4 | fixed padding |
| 5..15 | Reserved |

**nnpfc\_luma\_padding\_val** specifies the luma value to be used for padding when nnpfc\_padding\_type is equal to 4.

**nnpfc\_cb\_padding\_val** specifies the Cb value to be used for padding when nnpfc\_padding\_type is equal to 4.

**nnpfc\_cr\_padding\_val** specifies the Cr value to be used for padding when nnpfc\_padding\_type is equal to 4.

The function InpSampleVal( y, x, picHeight, picWidth, croppedPic ) with inputs being a vertical sample location y, a horizontal sample location x, a picture height picHeight, a picture width picWidth, and sample array croppedPic returns the value of sampleVal derived as follows:

if( nnpfc\_padding\_type = = 0 )  
 if( y < 0 | | x < 0 | | y >= picHeight | | x >= picWidth )  
 sampleVal = 0  
 else  
 sampleVal = croppedPic[ x ][ y ] (82)  
else if( nnpfc\_padding\_type = = 1 )  
 sampleVal = croppedPic[ Clip3( 0, picWidth − 1, x ) ][ Clip3( 0, picHeight − 1, y ) ]  
else if( nnpfc\_padding\_type = = 2 )   
 sampleVal = croppedPic[ Reflect( picWidth − 1, x ) ][ Reflect( picHeight − 1, y ) ]  
else if( nnpfc\_padding\_type = = 3 )   
 if( y >= 0 && y < picHeight ) sampleVal = croppedPic[ Wrap( picWidth − 1, x ) ][ y ]   
else if( nnpfc\_padding\_type = = 4 )   
 if( y < 0 | | x < 0 | | y >= picHeight | | x >= picWidth )  
 sampleVal[ 0 ] = nnpfc\_luma\_padding\_val sampleVal[ 1 ] = nnpfc\_cb\_padding\_val  
 sampleVal[ 2 ] = nnpfc\_cr\_padding\_val  
 else  
 sampleVal = croppedPic[ x ][ y ]

**Table 23 – Process for deriving the input tensors inputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors**

|  |  |
| --- | --- |
| **nnpfc\_inp\_ order\_idc** | **Process DeriveInputTensors( ) for deriving input tensors** |
| 0 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  inpVal = InpY( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight,  CroppedWidth, CroppedYPic ) )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpVal  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpVal  if( nnpfc\_auxiliary\_inp\_idc = = 1 )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = 2(StrengthControlVal − 42) / 6  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = 2(StrengthControlVal − 42) / 6  } |
| 1 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  inpCbVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight / SubHeightC,  CroppedWidth / SubWidthC, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( cTop + yP, cLeft + xP, CroppedHeight / SubHeightC,  CroppedWidth / SubWidthC, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpCrVal  }  if( nnpfc\_auxiliary\_inp\_idc = = 1 )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 2 ][ yP + overlapSize ][ xP + overlapSize ] = 2(StrengthControlVal − 42) / 6  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 2 ] = 2(StrengthControlVal − 42) / 6  } |
| 2 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  yY = cTop + yP  xY = cLeft + xP  yC = yY / SubHeightC  xC = xY / SubWidthC   inpYVal = InpY( InpSampleVal( yY, xY, CroppedHeight,  CroppedWidth, CroppedYPic ) )  inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,  CroppedWidth / SubWidthC, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / SubHeightC,  CroppedWidth / SubWidthC, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpYVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 2 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpYVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 2 ] = inpCrVal  }  if( nnpfc\_auxiliary\_inp\_idc = = 1 )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 3 ][ yP + overlapSize ][ xP + overlapSize ] = 2(StrengthControlVal − 42) / 6  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 3 ] = 2(StrengthControlVal − 42) / 6  } |
| 3 | for( yP = −overlapSize; yP < inpPatchHeight + overlapSize; yP++)  for( xP = −overlapSize; xP < inpPatchWidth + overlapSize; xP++ ) {  yTL = cTop + yP \* 2  xTL = cLeft + xP \* 2  yBR = yTL + 1  xBR = xTL + 1  yC = cTop / 2 + yP  xC = cLeft / 2 + xP  inpTLVal = InpY( InpSampleVal( yTL, xTL, CroppedHeight,  CroppedWidth, CroppedYPic ) )  inpTRVal = InpY( InpSampleVal( yTL, xBR, CroppedHeight,  CroppedWidth, CroppedYPic ) )  inpBLVal = InpY( InpSampleVal( yBR, xTL, CroppedHeight,  CroppedWidth, CroppedYPic ) )  inpBRVal = InpY( InpSampleVal( yBR, xBR, CroppedHeight,  CroppedWidth, CroppedYPic ) )  inpCbVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,  CroppedWidth / 2, CroppedCbPic ) )  inpCrVal = InpC( InpSampleVal( yC, xC, CroppedHeight / 2,  CroppedWidth / 2, CroppedCrPic ) )  if( nnpfc\_component\_last\_flag = = 0 ) {  inputTensor[ 0 ][ 0 ][ yP + overlapSize ][ xP + overlapSize ] = inpTLVal  inputTensor[ 0 ][ 1 ][ yP + overlapSize ][ xP + overlapSize ] = inpTRVal  inputTensor[ 0 ][ 2 ][ yP + overlapSize ][ xP + overlapSize ] = inpBLVal  inputTensor[ 0 ][ 3 ][ yP + overlapSize ][ xP + overlapSize ] = inpBRVal  inputTensor[ 0 ][ 4 ][ yP + overlapSize ][ xP + overlapSize ] = inpCbVal  inputTensor[ 0 ][ 5 ][ yP + overlapSize ][ xP + overlapSize ] = inpCrVal  inputTensor[ 0 ][ 6 ][ yP + overlapSize ][ xP + overlapSize ] = 2(StrengthControlVal − 42) / 6  } else {  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 0 ] = inpTLVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 1 ] = inpTRVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 2 ] = inpBLVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 3 ] = inpBRVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 4 ] = inpCbVal  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 5 ] = inpCrVal }  if( nnpfc\_auxiliary\_inp\_idc = = 1 )  if( nnpfc\_component\_last\_flag = = 0 )  inputTensor[ 0 ][ 6 ][ yP + overlapSize ][ xP + overlapSize ] = 2(StrengthControlVal − 42) / 6  else  inputTensor[ 0 ][ yP + overlapSize ][ xP + overlapSize ][ 6 ] = 2(StrengthControlVal − 42) / 6  } |
| 4..255 | Reserved |

**nnpfc\_complexity\_idc** greater than 0 specifies that one or more syntax elements that indicate the complexity of the post-processing filter associated with the nnpfc\_id may be present. nnpfc\_complexity\_idc equal to 0 specifies that no syntax elements that indicates the complexity of the post-processing filter associated with the nnpfc\_id are present. The value nnpfc\_complexity\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_complexity\_idc greater than 1 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_complexity\_idc.

**nnpfc\_out\_format\_flag** equal to 0 indicates that the sample values output by the post-processing filter are real numbers and the functions OutY( ) and OutC( ) for converting luma sample values and chroma sample values, respectively, output by the post-processing, to integer values at bit depths BitDepthY and BitDepthC, respectively, are specified as follows:

OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1, Round( x \* ( ( 1  <<  BitDepthY ) − 1 ) ) )(83)

OutC( x )= Clip3( 0, ( 1  <<  BitDepthC ) − 1, Round( x \* ( ( 1  <<  BitDepthC ) − 1 ) ) )(84)

nnpfc\_out\_format\_flag equal to 1 indicates that the sample values output by the post-processing filter are unsigned integer numbers and the functions OutY( ) and OutC( ) are specified as follows:

shiftY = outTensorBitDepth − BitDepthY  
if( shiftY > 0 )  
 OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1, ( x + ( 1  <<  ( shiftY − 1 ) ) )  >>  shiftY ) (85)  
else  
 OutY( x ) = x  <<  ( BitDepthY − outTensorBitDepth )

shiftC = outTensorBitDepth − BitDepthC  
if( shiftC > 0 )  
 OutC( x )= Clip3( 0, ( 1  <<  BitDepthC ) − 1, ( x + ( 1  <<  ( shiftC − 1 ) ) )  >>  shiftC ) (86)  
else  
 OutC( x ) = x  <<  ( BitDepthC − outTensorBitDepth )

The variable outTensorBitDepth is derived from the syntax element nnpfc\_out\_tensor\_bitdepth\_minus8 as described below.

**nnpfc\_out\_tensor\_bitdepth\_minus8** plus 8 specifies the bit depth of sample values in the output integer tensor. The value of outTensorBitDepth is derived as follows:

outTensorBitDepth = nnpfc\_out\_tensor\_bitdepth\_minus8 + 8(87)

It is a requirement of bitstream conformance that the value of nnpfc\_out\_tensor\_bitdepth\_minus8 shall be in the range of 0 to 24, inclusive.

**nnpfc\_out\_order\_idc** indicates the output order of samples resulting from the post-processing filter. Table 24 contains an informative description of nnpfc\_out\_order\_idc values. The semantics of nnpfc\_out\_order\_idc in the range of 0 to 3, inclusive, are specified in Table 25, which specifies a process for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensors outputTensor for different values of nnpfc\_out\_order\_idc and a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors. When nnpfc\_purpose is equal to 2 or 4, nnpfc\_out\_order\_idc shall not be equal to 3. The value of nnpfc\_out\_order\_idc shall be in the range of 0 to 255, inclusive. Values of nnpfc\_out\_order\_idc greater than 3 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_out\_order\_idc.

**Table 24 – Informative description of nnpfc\_out\_order\_idc values**

|  |  |
| --- | --- |
| **nnpfc\_out\_ order\_idc** | **Description** |
| 0 | Only the luma matrix is present in the output tensor, thus the number of channels is 1. |
| 1 | Only the chroma matrices are present in the output tensor, thus the number of channels is 2. |
| 2 | The luma and chroma matrices are present in the output tensor, thus the number of channels is 3. |
| 3 | Four luma matrices and two chroma matrices are present in the output tensor, thus the number of channels is 6. This nnpfc\_out\_order\_idc can only be used when the chroma format is 4:2:0. |
| 4..255 | Reserved |

**Table 25 – Process for deriving sample values in the filtered output sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic from the output tensors outputTensor for a given vertical sample coordinate cTop and a horizontal sample coordinate cLeft specifying the top-left sample location for the patch of samples included in the input tensors**

|  |  |
| --- | --- |
| **nnpfc\_out\_ order\_idc** | **Process StoreOutputTensors( ) for deriving sample values in the filtered picture from the output tensors** |
| 0 | for( yP = 0; yP < outPatchHeight; yP++)  for( xP = 0; xP < outPatchWidth; xP++ ) {  yY = cTop \* outPatchHeight / inpPatchHeight + yP  xY = cLeft \* outPatchWidth / inpPatchWidth + xP  if ( yY < nnpfc\_pic\_height\_in\_luma\_samples && xY < nnpfc\_pic\_width\_in\_luma\_samples )  if( nnpfc\_component\_last\_flag = = 0 )  FilteredYPic[ xY ][yY ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  else  FilteredYPic[ xY ][ yY ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  } |
| 1 | for( yP = 0; yP < outPatchCHeight; yP++)  for( xP = 0; xP < outPatchCWidth; xP++ ) {  xSrc = cLeft \* horCScaling + xP  ySrc = cTop \* verCScaling + yP  if ( ySrc < nnpfc\_pic\_height\_in\_luma\_samples / outSubHeightC &&  xSrc < nnpfc\_pic\_width\_in\_luma\_samples / outSubWidthC )  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredCbPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredCrPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ 1 ][ yP ][ xP ] )  } else {  FilteredCbPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredCrPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 1 ] )  }  } |
| 2 | for( yP = 0; yP < outPatchHeight; yP++)  for( xP = 0; xP < outPatchWidth; xP++ ) {  yY = cTop \* outPatchHeight / inpPatchHeight + yP  xY = cLeft \* outPatchWidth / inpPatchWidth + xP  yC = yY / outSubHeightC   xC = xY / outSubWidthC   yPc = ( yP / outSubHeightC ) \* outSubHeightC  xPc = ( xP / outSubWidthC ) \* outSubWidthC  if ( yY < nnpfc\_pic\_height\_in\_luma\_samples && xY < nnpfc\_pic\_width\_in\_luma\_samples)  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredYPic[ xY ][ yY ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredCbPic[ xC ][ yC ] = OutC( outputTensor[ 0 ][ 1 ][ yPc ][ xPc ] )  FilteredCrPic[ xC ][ yC ] = OutC( outputTensor[ 0 ][ 2 ][ yPc ][ xPc ] )  } else {  FilteredYPic[ xY ][ yY ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredCbPic[ xC ][ yC ] = OutC( outputTensor[ 0 ][ yPc ][ xPc ][ 1 ] )  FilteredCrPic[ xC ][ yC ] = OutC( outputTensor[ 0 ][ yPc ][ xPc ][ 2 ] )  }  } |
| 3 | for( yP = 0; yP < outPatchHeight; yP++ )  for( xP = 0; xP < outPatchWidth; xP++ ) {  ySrc = cTop / 2 \* outPatchHeight / inpPatchHeight + yP  xSrc = cLeft / 2 \* outPatchWidth / inpPatchWidth + xP  if ( ySrc < nnpfc\_pic\_height\_in\_luma\_samples / 2 &&  xSrc < nnpfc\_pic\_width\_in\_luma\_samples / 2 )  if( nnpfc\_component\_last\_flag = = 0 ) {  FilteredYPic[ xSrc \* 2 ][ ySrc \* 2 ] = OutY( outputTensor[ 0 ][ 0 ][ yP ][ xP ] )  FilteredYPic[ xSrc \* 2 + 1 ][ ySrc \* 2 ] = OutY( outputTensor[ 0 ][ 1 ][ yP ][ xP ] )  FilteredYPic[ xSrc \* 2 ][ ySrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ 2 ][ yP ][ xP ] )  FilteredYPic[ xSrc \* 2 + 1][ ySrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ 3 ][ yP ][ xP ] )  FilteredCbPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ 4 ][ yP ][ xP ] )  FilteredCrPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ 5 ][ yP ][ xP ] )  } else {  FilteredYPic[ xSrc \* 2 ][ ySrc \* 2 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 0 ] )  FilteredYPic[ xSrc \* 2 + 1 ][ ySrc \* 2 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 1 ] )  FilteredYPic[ xSrc \* 2 ][ ySrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 2 ] )  FilteredYPic[ xSrc \* 2 + 1][ ySrc \* 2 + 1 ] = OutY( outputTensor[ 0 ][ yP ][ xP ][ 3 ] )  FilteredCbPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 4 ] )  FilteredCrPic[ xSrc ][ ySrc ] = OutC( outputTensor[ 0 ][ yP ][ xP ][ 5 ] )  }  } |
| 4..255 | Reserved |

A base post-processing filter for a cropped decoded output picture picA is the filter that is identified by the first neural-network post-filter characteristics SEI message, in decoding order, that has a particular nnpfc\_id value within a CLVS.

If there is another neural-network post-filter characteristics SEI message that has the same nnpfc\_id value, has nnpfc\_mode\_idc equal to 1, has different content than the neural-network post-filter characteristics SEI message that defines the base post-processing filter, and pertains to the picture picA, the base post-processing filter is updated by decoding the ISO/IEC 15938-17 bitstream in that neural-network post-filter characteristics SEI message to obtain a post-processing filter PostProcessingFilter( ). Otherwise, the post-processing processing filter PostProcessingFilter( ) is assigned to be the same as the base post-processing filter.

The following process is used to filter the cropped decoded output picture with the post-processing filter PostProcessingFilter( ) to generate the filtered picture, which contains Y, Cb, and Cr sample arrays FilteredYPic, FilteredCbPic, and FilteredCrPic, respectively, as indicated by nnpfc\_out\_order\_idc.

if( nnpfc\_inp\_order\_idc = = 0 )  
 for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight )  
 for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 1 )  
 for( cTop = 0; cTop < CroppedHeight / SubHeightC; cTop += inpPatchHeight )  
 for( cLeft = 0; cLeft < CroppedWidth / SubWidthC; cLeft += inpPatchWidth ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 2 )  
 for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight) (88)  
 for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }  
else if( nnpfc\_inp\_order\_idc = = 3 )  
 for( cTop = 0; cTop < CroppedHeight; cTop += inpPatchHeight \* 2 )  
 for( cLeft = 0; cLeft < CroppedWidth; cLeft += inpPatchWidth \* 2 ) {  
 DeriveInputTensors( )  
 outputTensor = PostProcessingFilter( inputTensor )  
 StoreOutputTensors( )  
 }

**nnpfc\_reserved\_zero\_bit** shall be equal to 0.

**nnpfc\_uri\_tag**[ i ] contains a NULL-terminated UTF-8 character string specifying a tag URI. The UTF-8 character string shall contain a URI, with syntax and semantics as specified in IETF RFC 4151, uniquely identifying the format and associated information about the neural network used as the post-processing filter specified by nnrpf\_uri[ i ] values.

NOTE 4 – nnrpf\_uri\_tag[ i ] elements represent a 'tag' URI, which enables uniquely identifying the format of neural network data specified by nnrpf\_uri[ i ] values without needing a central registration authority.

**nnpfc\_****uri**[ i ] shall contain a NULL-terminated UTF-8 character string, as specified in ISO/IEC 10646. The UTF-8 character string shall contain a URI, with syntax and semantics as specified in IETF Internet Standard 66, identifying the neural network information (e.g. data representation) used as the post-processing filter.

**nnpfc\_payload\_byte**[ i ] contains the i-th byte of a bitstream conforming to ISO/IEC 15938-17. The byte sequence nnpfc\_payload\_byte[ i ] for all present values of i shall be a complete bitstream that conforms to ISO/IEC 15938-17.

**nnpfc\_parameter\_type\_idc** equal to 0 indicates that the neural network uses only integer parameters. nnpfc\_parameter\_type\_flag equal to 1 indicates that the neural network may use floating point or integer parameters. nnpfc\_parameter\_type\_idc equal to 2 indicates that the neural network uses only binary parameters. nnpfc\_parameter\_type\_idc equal to 3 is reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved value of nnpfc\_parameter\_type\_idc.

**nnpfc\_log2\_parameter\_bit\_length\_minus3** equal to 0, 1, 2, and 3 indicates that the neural network does not use parameters of bit length greater than 8, 16, 32, and 64, respectively. When nnpfc\_parameter\_type\_idc is present and nnpfc\_log2\_parameter\_bit\_length\_minus3 is not present the neural network does not use parameters of bit length greater than 1.

**nnpfc\_num\_parameters\_idc** indicates the maximum number of neural network parameters for the post processing filter in units of a power of 2048. nnpfc\_num\_parameters\_idc equal to 0 indicates that the maximum number of neural network parameters is not specified. The value nnpfc\_num\_parameters\_idc shall be in the range of 0 to 52, inclusive. Values of nnpfc\_num\_parameters\_idc greater than 52 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this document. Decoders conforming to this version of this document shall ignore SEI messages that contain reserved values of nnpfc\_num\_parameters\_idc.

If the value of nnpfc\_num\_parameters\_idc is greater than zero, the variable maxNumParameters is derived as follows:

maxNumParameters = ( 2048  <<  nnpfc\_num\_parameters\_idc ) − 1(88)

It is a requirement of bitstream conformance that the number of neural network parameters of the post-processing filter shall be less than or equal to maxNumParameters.

**nnpfc\_num\_kmac\_operations\_idc** greater than 0 specifies that the maximum number of multiply-accumulate operations per sample of the post-processing filter is less than or equal to nnpfc\_num\_kmac\_operations\_idc \* 1000. nnpfc\_num\_kmac\_operations\_idc equal to 0 specifies that the maximum number of multiply-accumulate operations of the network is not specified. The value of nnpfc\_num\_kmac\_operations\_idc shall be in the range of 0 to 232 − 1, inclusive.

* 1. **Neural-network post-filter activation SEI message**
     1. **Neural-network post-filter activation SEI message syntax**

|  |  |
| --- | --- |
| nn\_post\_filter\_activation( payloadSize ) { | **Descriptor** |
| **nnpfa\_id** | ue(v) |
| } |  |

* + 1. **Neural-network post-filter activation SEI message syntax**

This SEI message specifies the neural-network post-processing filter that may be used for post-processing filtering for the current picture.

The neural-network post-processing filter activation SEI message persists only for the current picture.

NOTE – There can be several neural-network post-processing filter activation SEI messages present for the same picture, for example, when the post-processing filters are meant for different purposes or filter different colour components.

**nnpfa\_id** specifies that the neural-network post-processing filter specified by one or more neural-network post-processing filter characteristics SEI messages that pertain to the current picture and have nnpfc\_id equal to nnfpa\_id may be used for post-processing filtering for the current picture.

* 1. **Phase indication SEI message**
     1. **Phase indication SEI message syntax**

|  |  |
| --- | --- |
| phase\_indication( payloadSize ) { | **Descriptor** |
| **hor\_phase\_num** | u(8) |
| **hor\_phase\_den\_minus1** | u(8) |
| **ver\_phase\_num** | u(8) |
| **ver\_phase\_den\_minus1** | u(8) |
| } |  |

* + 1. **Phase indication SEI message semantics**

The phase indication SEI message provides the decoder with information about the position of luma sampling locations in cropped decoded pictures relative to a rendering window. This information may be used by a decoder to ensure the correct spatial alignment of rendered pictures, for example when switching between picture resolutions.



**Figure 1: The ratios and represent the horizontal and vertical locations of the luma samples (marked with x) relative to a rendering window. is equal to hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ), and to ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ).**

NOTE 1 – When the number of luma output samples in horizontal direction is equal to the width of the rendering window, and hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ) is equal to 1÷2, the picture is intended to be rendered without applying any horizontal phase shift. Correspondingly, when the number of luma output samples in vertical direction is equal to the height of the rendering window, and ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ) is equal to 1÷2, the picture is intended to be rendered without applying any vertical phase shift.

The phase indication SEI message applies to the current cropped decoded picture and persists for all subsequent pictures of the current layer in output order with the same value of ph\_pic\_parameter\_set\_id as the current picture until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer with an associated phase indication SEI message and the same value of ph\_pic\_parameter\_set\_id as the current picture is output and follows the current picture in output order.

**hor\_phase\_num** and **hor\_phase\_den\_minus1** specify the horizontal position of luma sampling locations relative to a rendering window. The horizontal position hor\_phase\_num ÷ ( hor\_phase\_den\_minus1 + 1 ) is expressed in units of the horizontal distance between two horizontally adjacent luma sampling locations. The value of hor\_phase\_num shall be greater than or equal to 0 and less than or equal to hor\_phase\_den\_minus1 + 1.

**ver\_phase\_num** and **ver\_phase\_den\_minus1** specify the vertical position of luma sampling locations relative to a rendering window. The vertical position ver\_phase\_num ÷ ( ver\_phase\_den\_minus1 + 1 ) is expressed in units of the vertical distance between two vertically adjacent luma sampling locations. The value of ver\_phase\_num shall be greater than or equal to 0 and less than or equal to ver\_phase\_den\_minus1 + 1.

NOTE 2 – The phase indicators can be used during the rendering process. For example, texture coordinates can be offset by an amount proportional to the signalled horizontal and vertical phase indicators.

NOTE 3 – The signalled phase indicators applies to the luma samples of the decoded pictures. The phase offset for chroma samples can be derived from the signalled phase indicators taking into account the chroma sample location relative to luma sample location as indicated by ChromaFormatIdc, vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field. When ChromaFormatIdc is equal to 1 (4:2:0 chroma format) and the value of vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field, as applicable, are equal to 6 or are inferred to be equal to 6, the nominal vertical and horizontal relative locations of luma and chroma samples that corresponds to vui\_chroma\_sample\_loc\_type\_frame, vui\_chroma\_sample\_loc\_type\_top\_field and vui\_chroma\_sample\_loc\_type\_bottom\_field equal to 0 may be assumed.

*Bibliography*

Add a bibliographic reference as follows:

[16] ATSC A/341:2022-03, *ATSC Standard: Video – HEVC*.