



International
Standard

ISO/IEC 21122-1

**Information technology — JPEG
XS low-latency lightweight image
coding system —**

**Part 1:
Core coding system**

*Technologies de l'information — Système de codage d'images
léger à faible latence JPEG XS —*

Partie 1: Système de codage de noyau

**Third edition
2024-07**

Reference number
ISO/IEC 21122-1:2024(en)

Full document details: ISO IEC 21122-1-2024

 **Documents**
© ISO/IEC 2024 Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	iv
1 Scope	1
2 Normative references	1
3 Terms and definitions, abbreviated terms and symbols	1
3.1 Terms and definitions	1
3.2 Abbreviated terms	6
3.3 Symbols	6
4 Conventions	10
4.1 Conformance language	10
4.2 Operators	10
4.2.1 Arithmetic operators	10
4.2.2 Logical operators	10
4.2.3 Relational operators	10
4.2.4 Precedence order of operators	10
4.2.5 Mathematical functions	11
5 Functional concepts	11
5.1 Sample grid, sampling and components	11
5.2 Interpretation of CFA data	12
5.3 Wavelet decomposition	12
5.4 Codestream	13
6 Encoder requirements	13
7 Decoder	13
7.1 Decoding process general provisions	13
7.2 Decoder requirements	15
Annex A (normative) Codestream syntax	16
Annex B (normative) Image data structures	32
Annex C (normative) Entropy decoding	45
Annex D (normative) Quantization	65
Annex E (normative) Discrete wavelet transformation	70
Annex F (normative) Multiple component transformations	80
Annex G (normative) DC level shifting, non-linear transform and output clipping	91
Annex H (normative) Frame buffer	98
Annex I (informative) Example weight tables	106
Bibliography	114

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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 ISO 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).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

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.

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*.

This third edition cancels and replaces the second edition (ISO/IEC 21122-1:2022), which has been technically revised.

The main changes are as follows:

- coding tools for improving the compression rates for screen content images have been added;
- coding tools that enable lossless coding of images with up to 16 bits per sample have been added.

A list of all parts in the ISO/IEC 21122 series can be found on the ISO and IEC websites.

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 and www.iec.ch/national-committees.

Information technology — JPEG XS low-latency lightweight image coding system —

Part 1: Core coding system

1 Scope

This document specifies the syntax and an accompanying decompression process that is capable to represent continuous-tone grey-scale, or continuous-tone colour digital images without visual loss at moderate compression rates. Typical compression rates are between 2:1 and 18:1 but can also be higher depending on the nature of the image. In particular, the syntax and the decoding process specified in this document allow lightweight encoder and decoder implementations that limit the end-to-end latency to a fraction of the frame size. However, the definition of transmission channel buffer models necessary to ensure such latency is beyond the scope of this document.

This document:

- specifies decoding processes for converting compressed image data to reconstructed image data;
- specifies a codestream syntax containing information for interpreting the compressed image data;
- provides guidance on encoding processes for converting source image data to compressed image data.

2 Normative references

There are no normative references in this document.

3 Terms and definitions, abbreviated terms and symbols

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

3.1.1

band

input data to a specific *wavelet filter type* (3.1.54) that contributes to the generation of one of the *components* (3.1.14) of the image

3.1.2

band type

single number collapsing the information on the component, and horizontal and vertical wavelet filter types that are applied in the filter cascade reconstructing spatial image samples from inversely quantized wavelet coefficients

3.1.3

bit

binary choice encoded as either 0 or 1

3.1.4

bitplane

array of bits having all the same significance

3.1.5

bitplane count

number of significant bitplanes of a code group, counting from the LSB up to the most significant, non-empty bitplane

3.1.6

bitplane count subpacket

subset of a packet which decodes to the bitplane counts of all code groups within a packet, followed by padding and optional filler bytes

Note 1 to entry: See [subclause C.5.3](#).

3.1.7

byte

group of 8 bits

3.1.8

colour filter array

CFA

rectangular array of sensor elements yielding a 1-component image where the colour to which a sensor element is sensitive to depends on the position of the sensor element

3.1.9

codestream

compressed image data representation that includes all necessary data to allow a (full or approximate) reconstruction of the sample values of a digital image

3.1.10

code group

group of quantization indices in sign-magnitude representation before inverse quantization

3.1.11

coefficient

input value to the inverse wavelet transformation resulting from inverse quantization

3.1.12

coefficient group

number of horizontally adjacent wavelet coefficients from the same band

3.1.13

column

set of vertically aligned precincts

3.1.14

component

two-dimensional array of samples having the same designation such as red, green or blue in the output or display device

3.1.15

compression

process of reducing the number of bits used to represent source image data

3.1.16

continuous-tone image

image whose components have more than one bit per sample

3.1.17

data subpacket

subset of a packet which consists of the quantization index magnitudes, followed by padding and optional filler bytes

Note 1 to entry: See [subclause C.5.4](#).

3.1.18

deadzone quantizer

quantizer whose zero bucket has a size different from all other buckets

3.1.19

decoder

embodiment of a decoding process

3.1.20

decoding process

process which takes as its input a codestream and outputs a continuous-tone image

3.1.21

decomposition level

number of times a wavelet filter is applied to reconstruct image data from wavelet coefficients

3.1.22

encoder

embodiment of an encoding process

3.1.23

encoding process

process which outputs compressed image data in the form of a codestream

3.1.24

filler bytes

integer number of bytes a decoder will skip over on decoding without interpreting the values of the bytes itself

3.1.25

intra coding

coding process which decodes data independent from data decoded in a previous frame

Note 1 to entry: ISO/IEC 21122-1:2022 only defined intra coding tools.

3.1.26

inverse quantization

inverse procedure to quantization by which the decoder recovers a representation of the coefficients

3.1.27

inverse reversible multiple component transformation

inverse RCT

inverse transform across multiple component sample values located at the same sample grid point that is invertible without loss

Note 1 to entry: See [subclauses F.3](#) and [F.4](#).

3.1.28

LL band

input to a series of wavelet filters where only inverse low-pass filters are applied in horizontal and vertical direction

3.1.29

lossless

descriptive term for encoding and decoding processes and procedures in which the output of the decoding procedure(s) is identical to the input to the encoding procedure(s)

3.1.30

lossless coding

mode of operation which refers to any one of the coding processes defined in this document in which all of the procedures are lossless

3.1.31

lossy

descriptive term for encoding and decoding processes which are not lossless

3.1.32

packet

segment of the codestream containing entropy coded information on a single precinct, line and a subset of the bands within this precinct and line

3.1.33

padding

bits within the codestream whose only purpose is to align syntax elements to byte boundaries and that carry no information

3.1.34

precinct

collection of quantization indices of all bands contributing to a given spatial region of the image

3.1.35

precision

number of bits allocated to a particular sample, coefficient, or other binary numerical representation

3.1.36

procedure

set of steps which accomplishes one of the tasks which comprise an encoding or decoding process

3.1.37

quantization

method of reducing the precision of the individual coefficients

3.1.38

quantization index

input to the inverse quantization process which reconstructs the quantization index to a wavelet coefficient

3.1.39

quantization index magnitude

absolute value of a quantization index

3.1.40

sample

one element in the two-dimensional image array which comprises a component

3.1.41

sample grid

common coordinate system for all samples of an image, the samples at the top left edge of the image have the coordinates (0,0), the first coordinate increases towards the right, the second towards the bottom

3.1.42**sign subpacket**

subset of a packet that consists of the sign information of all non-zero quantization indices within a packet, followed by padding and optional filler bytes

Note 1 to entry: See [subclause C.5.5](#).

3.1.43**significance**

attribute of code groups that applies if, depending on the Run Mode flag in the picture header, either at least one of coefficients in the code group is non-zero, or the bitplane count prediction residual of the code group is non-zero

3.1.44**significance group**

group of a horizontally adjacent code groups sharing the same significance information in the significance subpacket

3.1.45**significance subpacket**

subset of a packet that identifies which significance groups within a packet are insignificant, followed by padding and optional filler bytes

Note 1 to entry: see [subclause C.5.2](#)

3.1.46**slice**

integral number of precincts whose wavelet coefficients can be entropy-decoded independently

3.1.47**star-tetrix**

decorrelation transformation that combines a spatial with an inter-component decorrelation transformation particularly tuned for CFA pattern compression

Note 1 to entry: see [subclause F.5](#)

3.1.48**subpacket**

substructure of a packet containing information of one or multiple bands of one line of a single precinct

3.1.49**super pixel**

2×2 arrangement of sensor elements in a CFA pattern array containing at least one sensor element for each colour filter type

3.1.50**temporal differential coding****TDC**

coding process which decodes a differential signal relative to a sample decoded in a previous frame

3.1.51**truncation position**

number of least significant bitplanes not included in the quantization index of a wavelet coefficient

3.1.52**uniform quantizer**

quantizer whose buckets are all of equal size

3.1.53**upsampling**

procedure by which the spatial resolution of a component is increased

3.1.54**wavelet filter type**

single number that uniquely identifies each element of the wavelet filter with regard to the number and type of horizontal and vertical decompositions

Note 1 to entry: Unlike the band type, the wavelet filter type does not include component information.

3.2 Abbreviated terms

JPEG XS	informal name of this standard where XS stands for “extra speed”
LSB	least significant bit
MSB	most significant bit

3.3 Symbols

$B[i]$	bit precision of component i
β	wavelet filter type
b	band type
$b_x[\beta, i]$	band existence flag for filter type β in component i . 1 if the filter exists, 0 otherwise.
$b'_x[b]$	band existence flag for band type b . 1 if the filter exists, 0 otherwise.
B_w	nominal overall bit precision of the wavelet coefficients
B_r	number of bits required to encode a bitplane count in raw
C_{pih}	colour transformation type
$c[p, \lambda, b, x]$	wavelet coefficient residual in precinct p , line λ , band b and position x
$c'[p, \lambda, b, x]$	wavelet coefficient in precinct p , line λ , band b and position x
C_s	width of precincts other than the rightmost precinct in sample grid positions
C_t	colour transformation CFA pattern type derived from the component registration
C_f	colour transformation reflection and extension flags
C_w	width of precincts in multiples of 8 LL subsampled band sample grid positions
$D[p, b]$	bitplane count coding mode of band b in precinct p
$D_i[p, b]$	TDC mode of band b in precinct p
$D_r[p, s]$	raw coding mode override flag for packet s in precinct p
DCO	DC offset
$d_x[\beta, i]$	horizontal decomposition level of wavelet filter type β of component i
$d_y[\beta, i]$	vertical decomposition level of wavelet filter type β of component i
$\delta_x[c]$	horizontal position of component c in a CFA super pixel
$\delta_y[c]$	vertical position of component c in a CFA super pixel

E	exponent of the slope of the linear region of the extended non-linearity
e_1	colour transformation exponent of first chroma component
e_2	colour transformation exponent of second chroma component
$f[p,\lambda,b,x]$	contents of the frame buffer at precinct p , line λ , band b and position x
F_s	sign packing flag
F_{slc}	slice coding mode
F_q	number of fractional bits in the representation of wavelet coefficients
$G[b]$	gain of band b
$G_r[b]$	gain of band b under forced refresh
$H_b[\beta,k]$	height of filter type β of component k in wavelet coefficients
$H_c[i]$	height of the component i in sample points
H_f	height of the image in sampling grid points
H_p	height of a precinct in lines
H_{sl}	height of a slice in precincts
I_{sl}	slice TDC flag, set if wavelet coefficients within a slice may use TDC
$I[p,\lambda,b,s]$	line inclusion flag, set if line λ of band b and precinct p is included in packet s , reset otherwise
$k[\delta_x, \delta_y]$	component within CFA super pixel at position δ_x, δ_y
$L_0[p,b]$	first line of band b in precinct p
$L_1[p,b]$	last line + 1 of band b in precinct p
L_{cod}	codestream length in bytes
$L_{cnt}[p,s]$	size of the bitplane count subpacket of precinct p and packet s in bytes
$L_{dat}[p,s]$	size of the data subpacket of precinct p and packet s in bytes
L_h	long header flag in in the picture header, set if long headers are enforced, reset otherwise
$L_{prc}[p]$	length of the entropy coded data in precinct p
$L_{sgn}[p,s]$	size of the sign subpacket of precinct p and packet s in bytes
$L_{sig}[p,s]$	size of the significance subpacket of precinct p and packet s in bytes
$M[p,\lambda,b,g]$	bitplane count of precinct p , line λ , band b and code group g
$M_{top}[p,\lambda,b,g]$	vertical predictor of the bitplane count of precinct p , line λ , band b and code group g
N_c	number of components in an image
$N_{cg}[p,b]$	number of code groups in precinct p and band b
N_β	number of bands per component

N_g	number of coefficients in a code group
$N_i[p,b]$	number of TDC selection groups per line in band b of precinct p
$N_s[p,b]$	number of significance groups per line in band b of precinct p
$N_p[t]$	number of precincts in slice t
N_L	number of bands in the wavelet decomposition of the image (wavelet filter types times components)
$N_{L,x}$	maximal number of horizontal decomposition levels
$N'_{L,x}[i]$	number of horizontal decomposition levels of component i
$N_{L,y}$	maximal number of vertical decomposition levels over all components
$N'_{L,y}[i]$	number of vertical decomposition levels of component i
$N_{p,x}$	number of precincts per sampling grid line
$N_{p,y}$	number of precincts per sampling grid column
$N_{pc}[p]$	number of packets in precinct p
$O[c,x,y]$	unscaled output of the inverse wavelet transformation at coordinates x and y of the component c
$\Omega[c,x,y]$	output of the inverse multiple component transformation at position x,y for component c
$P[b]$	priority of band b
$P_r[b]$	priority of band b under forced refresh
P_{lev}	level a particular codestream complies to
P_{pih}	profile a particular codestream complies to
P_{poc}	progression order in which bands are transmitted in the codestream
$Q[p]$	quantization parameter of precinct p
$Q_f[p]$	quantization parameter to which precinct p will be quantized for storage in the frame buffer
$Q'_f[p]$	quantization parameter of the data stored in the frame buffer corresponding to precinct p
Q_{bi}	quantization adjustment for intra-coded coefficients that are intra-coded due to a rate-decision
Q_{br}	quantization adjustment for intra-coded coefficients that are intra-coded due to refresh, overriding any rate-based TDC decision
Q_{pih}	quantization type of the image
R_l	raw-mode selection per packet flag
Rm	run mode used for significance coding
$R[p]$	refinement of precinct p
$R_f[p]$	refinement parameter of the quantization to which precinct p will be quantized for storage in the frame buffer

$R'_f[p]$	refinement parameter of the quantization of the data stored in the frame buffer corresponding to precinct p
$R[c,x,y]$	reconstructed sample value at position x,y for component c
S_d	number of components for which wavelet decomposition is suppressed
$S_h[b]$	intra refresh hash mask exponent of band b
S_i	size of a TDC selection group in code groups
S_s	size of a significance group in code groups
$s_x[i]$	sampling factor of component i in horizontal direction
$s_y[i]$	sampling factor of component i in vertical direction
$s[p,\lambda,b,x]$	sign of the wavelet coefficient in precinct p , line λ , band b and position x .
$T1$	first threshold of the extended non-linearity
$T2$	second threshold of the extended non-linearity
$T[p,b]$	truncation position of precinct p and band b
$T_{top}[p,b]$	vertical truncation position predictor of precinct p and band b
$T[\beta,x,y]$	temporary wavelet coefficient of filter type β at location x,y .
$v[x,y]$	sample value at the sample grid position x,y
$v[p,\lambda,b,x]$	quantization index magnitude of the wavelet coefficient in precinct p , line λ , band b and position x
$W_b[\beta,k]$	width of filter type β of component k in wavelet coefficients
$W_c[i]$	width of component i in samples
W_f	width of the image in sampling grid points
$W_p[p]$	width of the precinct p in sampling grid points
$W_{pb}[p,b]$	width of band b of precinct p in coefficients
Wt_x	wavelet filter type for horizontal filtering
Wt_y	wavelet filter type for vertical filtering
$X[y]$	one-dimensional temporal array of wavelet coefficients
$Xcrg[c]$	horizontal component registration of component c relative to the sample grid
$Ycrg[c]$	vertical component registration of component c relative to the sample grid
Ysl	slice index enumerating slices contiguously from top to bottom, starting at 0
$Y[p,\lambda,b,k]$	TDC selection flag of precinct p , line λ , band b and TDC selection group k
$Y_h[b]$	intra refresh position hash of band b
$Z[p,\lambda,b,j]$	significance flag of precinct p , line λ , band b and significance group j

4 Conventions

4.1 Conformance language

The keyword "reserved" indicates a provision that is not specified at this time, shall not be used, and may be specified in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be specified in the future.

4.2 Operators

NOTE Many of the operators used in document are similar to those used in the C programming language.

4.2.1 Arithmetic operators

&	bitwise AND operation
+	addition
−	subtraction (as a binary operator) or negation (as a unary prefix operator)
×	multiplication
/	division without truncation or rounding
<<	left shift: $x \ll s$ is defined as $x \times 2^s$
>>	right shift: $x \gg s$ is defined as $\lfloor x / 2^s \rfloor$
umod	$x \bmod a$ is the unique value y between 0 and $a-1$ for which $y + Na = x$ with a suitable integer N

4.2.2 Logical operators

	logical OR
&&	logical AND
!	logical NOT

4.2.3 Relational operators

>	greater than
≥	greater than or equal to
<	less than
≤	less than or equal to
==	equal to
!=	not equal to

4.2.4 Precedence order of operators

NOTE Operators are listed below in descending order of precedence. If several operators appear in the same line, they have equal precedence. When several operators of equal precedence appear at the same level in an expression, evaluation proceeds according to the associativity of the operator either from right to left or from left to right.

Operators	Type of operation	Associativity
()	expression	left to right
[]	indexing of arrays	left to right
-	unary negation	
×, /	multiplication, division	left to right
umod	modulo (remainder)	left to right
+, -	addition and subtraction	left to right
<<, >>	left shift and right shift	left to right
<, >, ≤, ≥	relational	left to right
&	bitwise AND	left to right

4.2.5 Mathematical functions

$\lceil x \rceil$	ceil of x : returns the smallest integer that is greater than or equal to x
$\lfloor x \rfloor$	floor of x : returns the largest integer that is less than or equal to x
$ x $	absolute value of x , $ x $ equals $-x$ for $x < 0$, otherwise x
$\log_2(x)$	logarithm to the basis of 2, e.g. $\log_2(1)=0$ and $\log_2(2)=1$
$\text{sign}(x)$	sign of x , 0 if x is 0, +1 if x is positive, -1 if x is negative
$\sqrt{x}$	square root of x , i.e. non-negative number y such that $y \times y = x$
$\text{clamp}(x, \min, \max)$	clamp x to the range $[\min, \max]$ $\text{clamp}(x, \min, \max)$ equals $\min$ if $x < \min$, $\max$ if $x > \max$ or otherwise x
$\max_i(x_i)$	maximum of a sequence of numbers $\{x_i\}$ enumerated by the index i
$\max(a, b)$	a if $a > b$, otherwise b
$\min_i(x_i)$	minimum of a sequence of numbers $\{x_i\}$ enumerated by the index i
$\min(a, b)$	a if $a < b$, otherwise b

5 Functional concepts

5.1 Sample grid, sampling and components

An image is defined as a rectangular array of scalar or vectorial samples regularly aligned along a sample grid of W_f sample positions horizontally and H_f sample positions vertically. W_f is called the *width* and H_f is called the *height* of the image. The vector dimension of the image samples corresponds to the number of colour components present and is indicated by N_c , the number of *components* of the image. Each dimension of this vectorial data corresponds to one *component* of the image, and typically represents one of multiple colour channels of the data. Components may be red, green and blue, or Luma (Y) and Chroma (Cb, Cr). These are only non-exhaustive examples of components, and other uses are possible.

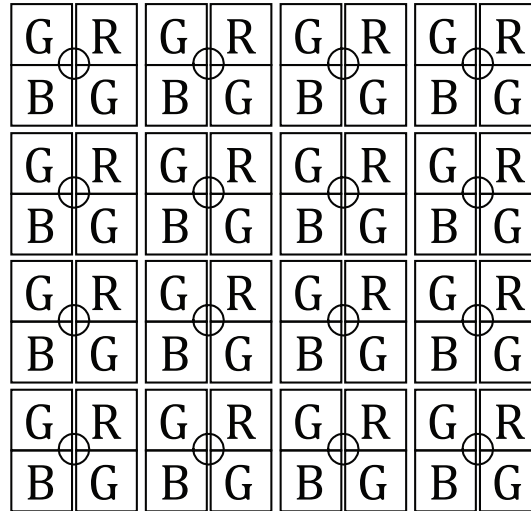
A given component may or may not populate every point on the sample grid. The distance, or *sampling factor*, between sample points of a component shall be constant in each spatial dimension throughout the image. The horizontal and vertical sampling factors of component i of an image are denoted by $s_x[i]$ respectively $s_y[i]$ where i enumerates the components. [Annex B](#) provides further specifications on component sampling.

This document *does not* specify how to interpret the sample values, or how to reconstruct from subsampled components an array of samples that populates the entire sample grid, i.e. it does not specify how to

upsample components to the full resolution of the sampling grid. The sampling grid provides only an abstract coordinate system for the computation of positions and dimensions of codestream elements.

5.2 Interpretation of CFA data

This document defines coding tools and signalling for compression of Bayer-type CFA image data. According to this specification, each sampling grid point represents a super pixel of four sensor elements containing at least one sample of each component. Thus CFA data is interpreted as an image having four components, where each sampling grid point describes four spatially disjoint sensor elements (one element per channel).



Squares represent individual sensor elements and circles represent sampling grid points. Groups of four sensor elements overlapping with the same sampling grid point form one super pixel.

Figure 1 — Example of the interpretation of a GRBG Bayer-type CFA image

Moreover, regardless of the CFA sensor spatial subpixel arrangement, the Star-Tetrix colour transform of this document defines a strict order on the components assigning the red channel to component 0, the green channels to components 1 and 2, and the blue channel to component 3. The spatial subpixel arrangement is signalled by the CRG marker. [Figure 1](#) shows only one of the four potential subpixel arrangements of a Bayer-type CFA.

5.3 Wavelet decomposition

This document provides an efficient representation of image signals through the mathematical tool of wavelet analysis. The wavelet filter process specified in [Annex E](#) separates each component into multiple *bands*, where each band consists of multiple *coefficients* describing the image signal of a given component within a frequency domain specific to the *wavelet filter type*, i.e. the particular filter corresponding to the band.

Wavelet coefficients are grouped into *precincts*, where each precinct includes all coefficients over all bands that contribute to a spatial region of the image. Each precinct is encoded into one or multiple *packets* in the codestream syntax specified in [Annex A](#).

Precincts are furthermore grouped into *slices*. Wavelet coefficients in precincts that are part of different slices can be decoded independently from each other. Note, however, that the wavelet transformation runs across slice boundaries. A slice always extends over the full width of the image, but may only cover parts of its height. Bands, band types, precincts and slices are formally defined in [Annex B](#).

5.4 Codestream

The codestream is a linear stream of bits from the first bit to the last bit. For convenience, it can be divided into (8-bit) bytes, starting with the first bit of the codestream. Bits within bytes are enumerated from the LSB to the MSB, with the least significant bit having the index zero.

[Annex A](#) specifies the codestream syntax that defines the coded representation of compressed image data for exchange between application environments. Any compressed image data shall comply with the syntax and code assignments appropriate for the decoding processes defined in this document.

The codestream consists of multiple syntax elements: *marker segments* define control information necessary to steer the decoding process, and *entropy coded data* organized in *packets* that represent image information itself. Packets are further grouped into *subpackets*, each of which includes particular information such as magnitude, signs or significance of parts of the encoded image data.

All marker segments defined in this document are specified in [Annex A](#). This annex also provides an overview on the organization of the codestream. Packets and subpackets are specified in [Annex C](#).

6 Encoder requirements

An encoder is an embodiment of a process that generates a codestream that conforms to the syntactical requirements specified in [Annex A](#). [Annex C](#) to [Annex G](#) include informative subclauses that indicate how an encoder may be implemented.

7 Decoder

7.1 Decoding process general provisions

[Figure 2](#) provides an overview on the decoding process and the layout of this document. Codestream decoding can be grouped into a syntax analysis part in block 1, an entropy decoding stage consisting of multiple blocks 2.1 to 2.5, an inverse quantization in block 3, an optional inverse temporal decorrelation in block 7, an inverse wavelet transformation in block 4 and an inverse multiple component transformation in block 5. In block 6, sample values are scaled, a DC offset is added, and they are clamped to their nominal ranges.

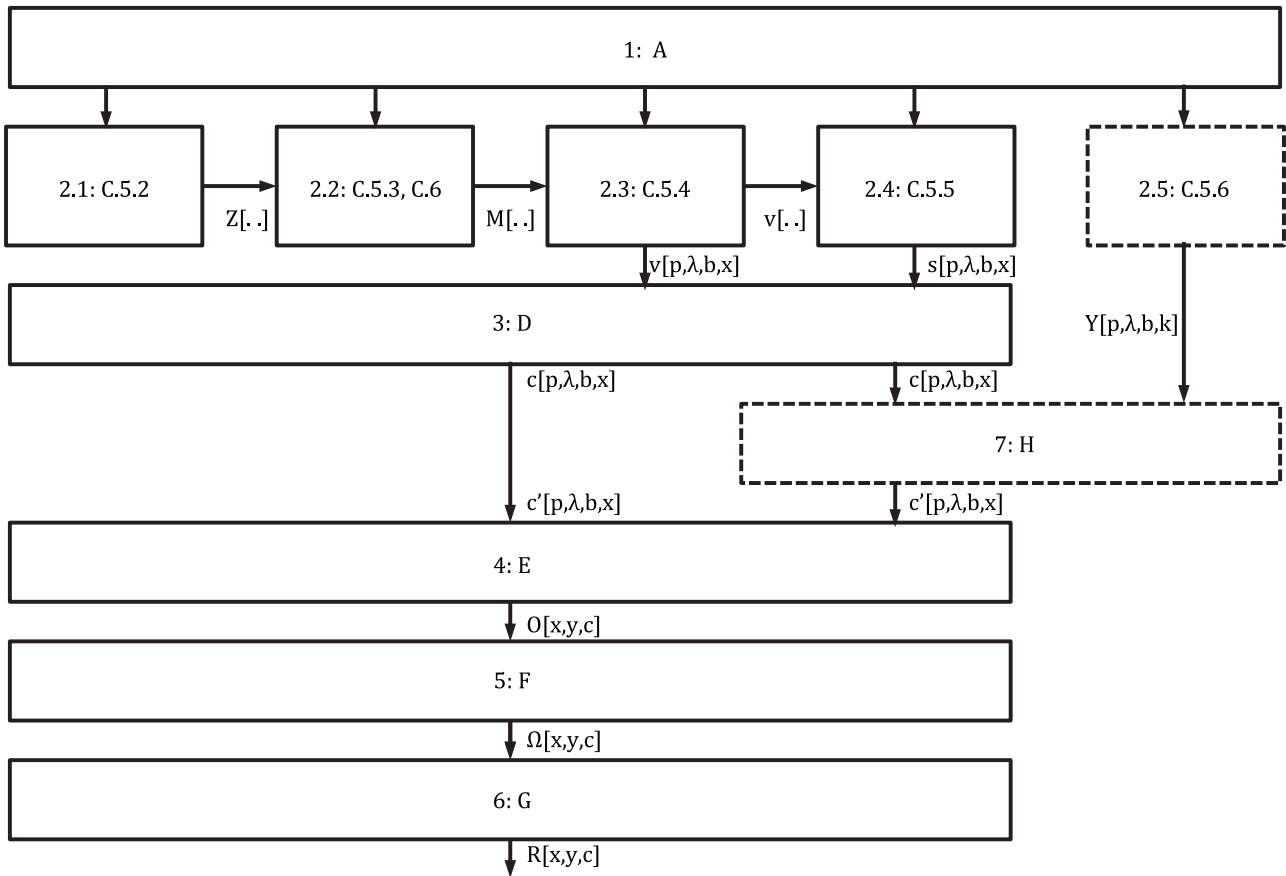


Figure 2 — Decoder overview

In Block 1, described in [Annex A](#), the decoder analyses the codestream syntax and retrieves information on the layout of the sampling grid, and the dimensions of slices and precincts.

The subpackets of the entropy coded data segment of the codestream are then decoded by the procedures given in [Annex C](#) to form significance information, sign information, bitplane count information, quantization indices and optionally TDC selection flags. This operation is performed in blocks 2.1 to 2.5 in [Figure 2](#).

In block 2.1, significance information is decoded from the *significance subpacket* as specified in [subclause C.5.2](#). Denoted by the array $Z[p,\lambda,b,j]$, significance information indicates the presence of significant code groups within the j^{th} significance group. Each significance group corresponds to a run of code groups indexed by precinct p , line λ and band b . A code group is significant if, depending on the Run Mode flag R_m in the picture header, it either contains non-zero coefficients, or has a non-zero bitplane count prediction residual.

In block 2.2, bitplane counts are decoded from the *bitplane count subpacket* as specified in [subclause C.5.3](#) by the procedures specified in [subclause E.6](#). The integer array $M[p,\lambda,b,g]$ indicates the bitplane counts of the wavelet coefficients in the code group g indexed by precinct p , line λ and band b .

In block 2.3, Quantization index magnitudes $v[p,\lambda,b,x]$ in precinct p , line λ , band b , and horizontal position x are decoded from the *data subpacket* as specified in [subclause C.5.4](#).

In block 2.4, the signs of the quantization indices $s[p,\lambda,b,x]$ are either interleaved in the data subpacket, or included in a separate sign subpacket as specified in [subclause C.5.5](#).

In block 2.5, the TDC selection flags $Y[p,\lambda,b,k]$ are decoded from the *TDC subpacket* as specified in [subclause C.5.6](#). This subpacket only exists for slices that enable TDC, see [subclause A.4.15](#). Otherwise, the TDC selection flags shall be inferred to be 0, indicating intra-coding.

In block 3, decoded quantization index magnitudes $v[p,\lambda,b,x]$ and signs $s[p,\lambda,b,x]$ are then inversely quantized by the dequantizer specified in [Annex D](#), giving wavelet coefficient residuals $c[p,\lambda,b,x]$.

In block 7, wavelet coefficient residuals $c[p,\lambda,b,x]$ are optionally inversely temporally predicted by means of the frame buffer $f[p,\lambda,b,x]$ and TDC selection flags $Y[p,\lambda,b,k]$ if they are present. The result of this operation are the wavelet coefficients $c'[p,\lambda,b,x]$. If the wavelet coefficient residuals are part of a precinct, band and line that does not use TDC, then $c'[p,\lambda,b,x]$ shall be identical to $c[p,\lambda,b,x]$. Inverse temporal decorrelation is specified in [Annex H](#).

In block 4, wavelet coefficients $c'[p,\lambda,b,x]$ are inversely wavelet transformed by the procedure specified in [Annex E](#). This process generates spatial sample values for all components, denoted by $O[x,y,c]$. Coordinates x and y are here subsampled sampling grid positions of component c .

In block 5, spatial sample values $O[x,y,c]$ undergo optionally an inverse multiple component transformation, giving intermediate image sample values $\Omega[x,y,c]$. The inverse multiple component transformation is specified in [Annex F](#).

In block 6, a DC offset is added to the decorrelated sample values $\Omega[x,y,c]$, an optional non-linear transformation is applied, they are scaled to their nominal range and then clamped to the range of the bit-precision of the output, giving the final reconstructed output sample values $R[x,y,c]$ populating the sample grid positions $x \times s_x[c], y \times s_y[c]$. This procedure is specified in [Annex G](#).

7.2 Decoder requirements

A decoder is an embodiment of the decoding process. The decoding process converts a codestream by performing the process specified in this document to sample values arranged on a rectangular sampling grid. [Annex A](#) to [Annex H](#) describe and normatively specify the decoding process. All decoding processes are normative. Decoder conformance and test procedures to test for conformance are specified in ISO/IEC 21122-4.

There is no normative or required specification for the particular internal steps or ordering of internal operations to be performed within the decoder that is used to produce the normatively specified result. Only the result that is externally observable as the decoded output image produced by the decoder is required to match the result produced by the decoding processes specified in this document up to a conformance-level dependent error bound that is specified in ISO/IEC 21122-4. The descriptions use particular implementation techniques for illustrative purposes only, and any implementation that is able to reproduce the same results as those generated by the algorithms specified herein is conforming to this document.

Annex A (normative)

Codestream syntax

A.1 General

A.1.1 Marker segments and entropy coded data

The compressed data format consists of an ordered collection of syntax elements. This document distinguishes between three types of syntax elements: *Marker*, *marker segments* and *entropy coded data*. Markers serve to identify the various structural parts of the codestream. Most markers start marker segments, where marker segments signal the characteristics of the encoded image and encapsulate parameters configuring the decoder. Some markers stand alone. Entropy coded data consists of the input to the decoding procedure described in [Annex C](#) to [Annex H](#) which reconstructs this data to the output image.

A.1.2 Key to syntax information

JPEG XS codestream syntax elements belong to one of two categories: *fixed-length* numerical values, or *variable-length* codes. In the syntax tables, the “Syntax” column indicates the category to which each codestream syntax element belongs, in the “Size” column the size of each field is identified (if applicable). Fixed-length numerical values are unsigned integers and are denoted by $u(n)$, where n is the number of bits used to represent the value. Variable-length codes are denoted by vlc , see [subclause C.7](#) for the normative decoding procedure of variable length codes. Bit strings and variable-length codes appear in the codestream with the left bit first; numerical values appear most-significant bit first. The notation $pad(n)$ indicates a variable number padding bits. Padding aligns the bitstream to an n -bit boundary, i.e. to an integer multiple of n bits relative to the start of the bitstream. Thus, $pad(n)$ expands to 0 to $n-1$ bits depending on the position within the bitstream. While padding bits can have arbitrary values, a decoder shall ignore their value. The notation $fill()$ indicates an arbitrary number of filler bytes a decoder shall remove without interpreting their value. The amount of filler bytes can be inferred from a length field of a corresponding syntax element.

Syntax elements may be conditionally included in the codestream; this is indicated by *if* clauses in the syntax column of the syntax tables. All syntactical elements enclosed in curly brackets following the *if* clause are only included if the expression following the *if* clause is non-zero.

The sequence of multiple similar elements is indicated by *for* clauses in the Syntax column of the syntax tables. The elements to repeat are enclosed in curly braces. The loop itself is specified through three syntax elements: an initializer setting a dummy count variable indicating the current iteration position of the loop, a condition on the count variable for continuing the loop, and an iteration statement that updates the count variable for the next loop. The three expressions are separated by semicolon.

NOTE The loop syntax and the syntax for conditional inclusion of elements follow closely the syntax of the C programming language.

A.2 Codestream syntax general provisions

A JPEG XS codestream describes an *image* consisting of 1 to 8 *components* aligned along a regular rectangular sampling grid. Each component is a rectangular arrangement of integer sample values on the sampling grid of the image. The samples of a component need not populate every possible position on the sampling grid, see [subclause B.1](#). The horizontal and vertical spacing between populated sample positions of a component relative to the sampling grid are denoted the *horizontal* and *vertical sampling factors* of the component and are indicated by the symbols $s_x[i]$ and $s_y[i]$. Subsampling factors may be either 1 or 2. The codestream

reconstruction process described by this document assigns to each sample of each image component an integer precision between 8 and 16 bits.

The dimensions of the image, along with the number of components, the sampling factors and the precision of the components are encoded in a syntax element denoted as *Picture Header*. The samples in the image are reconstructed by an inverse wavelet transformation and potentially from an inverse temporal prediction. This reconstruction uses entropy coded wavelet coefficients and/or wavelet coefficient residuals arranged in multiple wavelet bands, see [Annex B](#) for details, and potentially the contents of a frame buffer, see [Annex H](#). The wavelet reconstruction algorithm is specified in [Annex E](#).

Wavelet coefficients are grouped into precincts. A *precinct* includes entropy coded data decoding to a rectangular array of wavelet coefficients per bands such that the included wavelet coefficients contribute to a given spatial region of the image. A precinct is represented in the codestream by a *precinct header*, *entropy coded data* followed by *filler bytes*.

The entropy coded data is organized in multiple *packets*, where each packet s contributes to one line λ and one or multiple bands b of a precinct p . Each packet consists of multiple subpackets where each subpacket contributes to one aspect of the data, such as *significance*, *bitplane counts*, *magnitude* and *signs*. Filler bytes and padding does not have any impact on the decoded image. The purpose of the filler bytes is to prevent buffer underflow of a potential transmission buffer, the purpose of padding is to align the syntax elements in the bitstream to byte boundaries. Decoders learn the number of filler bytes following the precinct from the precinct header specified in [subclause C.2](#), and the number of filler bytes in the subpackets from the subpacket header in specified in [subclause C.3](#). Decoders shall ignore the value of the filler bytes.

NOTE 1 Filler bytes are in no relation to the output of the `pad()` function defined in [subclause A.1.2](#) whose purpose is to ensure alignment of the codestream to byte boundaries only.

NOTE 2 Buffer models, profiles and levels are specified in ISO/IEC 21122-2 and are beyond the scope of this document.

Precincts are grouped into *slices*. Each slice consists of an integral number of precincts, and extends over the full width of the image. Even though the wavelet coefficients/wavelet coefficient residuals within each slice can be decoded independently, the wavelet transformation runs across slices. A slice is represented in the codestream by a *slice header* and one or multiple *precincts* following the slice header.

[Figure 2](#) gives an overview of the hierarchy of JPEG XS codestream syntax structures, [Table A.1](#) defines the overall codestream syntax. While [Table A.1](#) lists the syntax elements of JPEG XS, it does not attempt to define a particular order of the markers in a JPEG XS codestream, and the order presented in this table is rather an illustration of one possible syntactically correct order. Requirements on the placement of a specific marker are normatively defined in the subclause of [Annex A](#) defining the marker.

Table A.1 — JPEG XS codestream syntax overview

Syntax	Notes	Defined in
<code>Picture() {</code>		
<code>SOC_marker()</code>	Identifies this codestream as JPEG XS codestream	Table A.3
<code>capabilities_marker()</code>	Identifies the capabilities a decoder needs to support to be able to decode the codestream	Table A.6
<code>picture_header()</code>	Defines the overall structure of the codestream	Table A.7
<code>component_table()</code>	Defines the precision and sampling factors of all components in the image	Table A.15
<code>weights_table()</code>	Defines weight and gain factors that steer the decoding process.	Table A.25
<code>refresh_weights_table()</code>	Defines weights and gain factors for forcibly refreshed bands in TDC-enabled slices	Table A.26
<code>nonlinearity_marker()</code>	Optional definition of non-linearities for component reconstruction	Table A.16

Table A.1 (continued)

Syntax	Notes	Defined in
<code>cwd_marker()</code>	Optionally disable wavelet decomposition on some components	Table A.18
<code>cts_marker()</code>	Colour transformation specification for the star-tetrix transformation	Table A.19
<code>crg_marker()</code>	Optional component registration, mandatory if the star-tetrix transformation is used	Table A.21
<code>tpc_marker()</code>	Optional indication of TDC modes and refresh groups.	Table A.22
<code>extension_marker()</code>	Optional extension of the codestream syntax	Table A.23
<code>for (t=0; p=0; !endofimage; t=t+1) {</code>	Loop over all slices until all wavelet coefficients of the image have been decoded, where t is the slice index and p the precinct index	
<code> slice_header() or tdc_slice_header()</code>	Identifies the ordering of slices and identifies the TDC mode of the slice. The slice header shall be either an SLH or SLI header, depending on the profile of the codestream.	Table A.27 , Table A.28
<code>for (u=0; u<N_p[t]; p=p+1, u=u+1) {</code>	Loops over all precincts in a slice, where t is the slice index, p is the precinct index and u enumerates precincts within a slice	
<code> compute_packet_inclusion(p)</code>	Determine the packets that are part of this precinct.	Table B.4
<code> precinct_header(p)</code>	Defines prediction modes and the quantization of the precinct	Table C.1
<code> for (s=0; s<N_{pc}[p]; s=s+1) {</code>	Loop over all packets of this precinct	
<code> packet_header(p, s)</code>	Defines flags and sizes of the packet	Table C.4
<code> packet_body(p, s)</code>	Contains the entropy coded data of this packet	Table C.5
<code> }</code>	End of loop over subpackets	
<code> fill()</code>	Possible byte-aligned filler bytes at the end of the precinct to reach the target bitrate. A decoder shall ignore this data. Subclause C.2 specifies how to determine the number of filler bytes.	Subclause C.2
<code> }</code>	End of loop over precincts within a slice	
<code>}</code>	End of loop over slices	
<code>EOC_marker()</code>	Identifies the end of the JPEG XS codestream	Table A.4
<code>}</code>		

NOTE 3 The number of packets $N_{pc}[p]$ depends on the precinct index p and is computed by the `compute_packet_inclusion(p)` function specified in [Table B.4](#); in particular, the last precinct of the image will, by this procedure, include less bands than all other precincts if the image height is not divisible by the precinct height.

A.3 Markers and marker segments

Markers serve the purpose to identify the various structural parts of the codestream format. Markers may either stand alone, or may start *marker segments* containing a related group of parameters. A *marker segment* consists of a marker, followed by a two-byte length field, followed by the parameters in the marker segment, denoted as *payload data* in the following. The two-byte length field identifies the length of the marker segment, which consists of the length of the payload data in bytes, and the size of the length field itself (two bytes). The length field does not include the size of the marker. Parameters are encoded with the most significant byte first, a convention often denoted as *big endian*.

All markers are assigned two-byte codes: a 0xff byte followed by a byte that is not equal to 0x00 or 0xff. [Table A.2](#) lists all markers used by this document, and, in combination with referenced tables from [Table A.1](#),

specifies whether they stand alone or introduce a marker segment. The semantics of each marker and associated marker segment are further specified in [subclause A.4](#).

Some marker segments are currently reserved for future ISO/IEC use. Marker segments can be mandatory, optional, or mandatory only if indicated by a capability signalled in the capability marker segment. Optional marker segments may be ignored by decoder implementations by skipping over the marker segment by using its length field. The capability marker may indicate the marker segments required to correctly decode a given codestream. If a decoder encounters a capability it does not implement, it should abort decoding.

If the length field of a marker segment does not match the specified value or is not in the specified range, the codestream is ill-formed. For that, decoders should check whether the marker length field has its specified value, or is within the range specified in this document.

NOTE 1 It is possible that future editions of this document will include additional fields in the marker segments and hence require an increase in the size the marker segments. Such additional fields can be necessary to decode the codestream. The above ensures that decoders fail properly when attempting to decode an extended codestream syntax that they do not support.

NOTE 2 Other International Standards implement bit-stuffing or byte-stuffing procedures to ensure that markers can be identified uniquely without decoding or interpreting entropy coded data segments. This is not the case for this document. If a decoder loses synchronization with the codestream, lower-level transport mechanisms are required to regain synchronization as it is not possible to search the codestream for markers; bit patterns within the entropy coded data segment can replicate the byte sequences used to identify marker segments.

Table A.2 — JPEG XS codestream markers

Code assignment	Symbol	Description	Mandatory/Optional	Reference
0xff10	SOC	Start of codestream	Mandatory	A.4.1
0xff11	EOC	End of codestream	Mandatory	A.4.2
0xff12	PIH	Picture header	Mandatory	A.4.4
0xff13	CDT	Component table	Mandatory	A.4.5
0xff14	WGT	Weights table	Mandatory	A.4.12
0xff15	COM	Extension marker	Optional	A.4.10
0xff16	NLT	Nonlinearity marker	Optional	A.4.6
0xff17	CWD	Component-dependent wavelet decomposition marker	Optional	A.4.7
0xff18	CTS	Colour transformation specification marker	Mandatory if $C_{pih}=3$, shall not be present otherwise.	A.4.8
0xff19	CRG	Component registration marker	Optional, mandatory if $C_{pih}=3$	A.4.9
0xff1a	TPC	Temporal prediction control marker	Optional	A.4.10
0xff1b	WGR	Refresh Weights table	Optional	A.4.13
0xff20	SLH	Slice header	Depending on profile, see ISO/IEC 21122-2	A.4.14
0xff21	SLI	TDC enabling slice header	Depending on profile, see ISO/IEC 21122-2	A.4.15
0xff50	CAP	Capabilities Marker	Mandatory	A.4.3
All other values			Optional	Reserved for future ISO/IEC purposes

A.4 Syntax description of marker segments

A.4.1 Start of codestream

Function: Identifies the codestream as containing an image represented in accordance with this document.

Usage: Shall be the first marker segment in a codestream. There shall be only one SOC marker at the beginning of each JPEG XS codestream.

Table A.3 — Start of codestream marker syntax

Syntax	Notes	Size	Values
start_of_codestream() {			
SOC		u(16)	0xff10
}			

A.4.2 End of codestream

Function: Identifies the end of a JPEG XS codestream.

Usage: Shall be the last marker segment in a codestream. There shall be exactly one EOC marker at the end of each JPEG XS codestream.

Table A.4 — End of codestream marker syntax

Syntax	Notes	Size	Values
end_of_codestream() {			
EOC		u(16)	0xff11
}			

A.4.3 Capabilities marker

Function: Identifies capabilities required to decode a JPEG XS codestream.

Usage: Shall be the second marker segment in a codestream. There shall be exactly one CAP marker which shall be placed behind the SOC marker.

NOTE 1 The above implies that the CAP marker is always present, even when the *cap[]* array is empty.

[Table A.5](#) specifies the assignment of capabilities to bits, [Table A.6](#) the syntax of the CAP marker segment. Furthermore, *Lcap* shall be selected such that for $Lcap > 2$, $cap[(Lcap-2) \times 8-8]$ to $cap[(Lcap-2) \times 8-1]$ are not all 0.

NOTE 2 This condition on the *cap* array can always be arranged by selecting a smaller *Lcap*.

NOTE 3 Bit 0 of the capability marker is intentionally unused.

Table A.5 — Capability bit assignment

Bit number i in $cap[]$ array	Bit value	Meaning
1	0	Support for Star-Tetrix transform not required, and support for CTS marker not required
1	1	Support for Star-Tetrix transform and CTS marker required
2	0	Support for quadratic non-linear transform not required
2	1	Support for quadratic non-linear transform required
3	0	Support for extended non-linear transform not required
3	1	Support for extended non-linear transform required
4	0	$s_y[i]=1$ for all components i
4	1	component i with $s_y[i]>1$ present
5	0	Support for component-dependent wavelet decomposition not required
5	1	Support for component-dependent wavelet decomposition required
6	0	Support for lossless decoding not required
6	1	Support for lossless decoding required
8	0	Support for packet-based raw-mode switch not required
8	1	Support for packet-based raw-mode switch required
9	0	Support for TDC not required
9	1	Support for TDC required, a frame buffer is required
All other bits	0	Reserved for ISO/IEC purposes

Table A.6 — Capabilities marker syntax

Syntax	Notes	Size	Values
<code>capabilities_marker() {</code>			
CAP		u(16)	0xff50
Lcap	Size of the capabilities marker in bytes (not including the marker)	u(16)	Variable
<code>for (i=0; i<(Lcap-2)*8; i=i+1) {</code>	Loop over capabilities bits		
<code>cap[i]</code>	Requirement of capability i . $cap[i]$ is 1 if capability i is required for decoding a codestream, and 0 otherwise.	u(1)	
<code>}</code>	End of loop over capabilities bits		
<code>padding</code>	Pad to an integer number of bytes	pad(8)	
<code>}</code>			

A.4.4 Picture header

[Table A.7](#) defines the syntax of the picture header.

Function: Provides information on the dimensions of the image, the precision of its component and the configuration of the decoder.

Usage: Shall be the third marker segment in a codestream directly after the CAP marker. There shall be exactly one PIH marker in a JPEG XS codestream.

Table A.7 — Picture header syntax

Syntax	Notes	Size	Values
<code>picture_header() {</code>			
PIH		u(16)	0xff12
Lpih	Size of the segment in bytes (not including the marker)	u(16)	26
Lcod	Size of the entire codestream in bytes from SOC to EOC, including all markers, if constant bitrate coding is used. 0 if variable bitrate coding is used.	u(32)	$0 \text{ --- } (2^{32} - 1)$
Ppih	Profile this codestream complies to. Decoders should abort decoding if they identify a profile they do not implement.	u(16)	0 for no restrictions, see ISO/IEC 21122-2 for profiles.
Plev	Level and sublevel to which this codestream complies. Decoders should abort decoding if they identify a level they do not support.	u(16)	0 for no restrictions, see ISO/IEC 21222-2 for additional levels.
W_f	Width of the image in sample grid positions	u(16)	$\max_i (s_x[i]) \times 2^{N_{L,x}} \text{ --- } (2^{16} - 1)$
H_f	Height of the image in sample grid positions	u(16)	$\max_i (s_y[i]) \times 2^{N_{L,y}} \text{ --- } (2^{16} - 1)$
C_w	Width of a precinct in multiples of $8 \times \max_i (s_x[i]) \times 2^{N_{L,x}}$ sample positions, other than the rightmost precincts. If this field is 0, precincts are as wide as the image. See subclause B.5 for details.	u(16)	$0 \text{ --- } (2^{16} - 1)$ such that either $C_w=0$ or $W_f \bmod (8 \times C_w \times \max_i (s_x[i]) \times 2^{N_{L,x}}) \geq \max_i (s_x[i]) \times 2^{N_{L,x}}$
H_{sl}	Height of a slice in precincts other than the last slice	u(16)	$1 \text{ --- } (2^{16} - 1)$
N_c	Number of components in the image	u(8)	1—8
N_g	Number of coefficients per code group	u(8)	4
S_s	Number of code groups per significance group	u(8)	8
B_w	Nominal bit precision of the wavelet coefficients	u(8)	20,18 or $B[0]^a$
F_q	Number of fractional bits in the wavelet coefficients	u(4)	8,6 or 0^a
B_r	Number of bits to encode a bitplane count in raw	u(4)	4 or 5^a
Fslc	Slice coding mode	u(1)	See Table A.14
Ppoc	Progression order of bands within precincts	u(3)	See Table A.13
C_{pih}	Colour transformation to be used for inverse decorrelation	u(4)	See Table A.9

^a See [Table A.8](#) for valid combinations of B_w , B_r and F_q and further restrictions.

Table A.7 (continued)

Syntax	Notes	Size	Values
$N_{L,x}$	Number of horizontal wavelet transformations	u(4)	1–8
$N_{L,y}$	Number of vertical wavelet transformations of non-vertically subsampled components	u(4)	$\max_i(\log_2(s_y[i])) - \min(N_{L,x}, 6)$
L_h	Long header enforcement flag	u(1)	0 or 1
R_1	Raw-mode selection per packet flag	u(1)	0 or 1
Q_{pih}	Inverse quantizer type	u(2)	See Table A.10
F_s	Sign handling strategy	u(2)	See Table A.11
R_m	Run mode	u(2)	See Table A.12
^a See Table A.8 for valid combinations of B_w , B_r and F_q and further restrictions.			

NOTE 1 The condition on C_w ensures that all but the rightmost precincts have in the LL band at least 8 samples, and that all bands of the rightmost precincts are non-empty.

NOTE 2 In case $Wf \times N_c > 16376$ and $N_{L,y} = 0$, or for components that do participate in the wavelet decomposition, it can happen that $L_{sgn}[p,s] \geq 2048$ or $L_{dat}[p,s] \geq 32768$. As such sizes cannot be expressed by the short header, the L_h flag allows to enforce long headers. L_h can safely remain 0 as long as at least one vertical decomposition is performed.

Table A.8 — Valid combinations of B_w , B_r , $B[i]$ and F_q

B_w	F_q	B_r	Additional constraints	Notes
$B[0]$	0	4	$B[i] = B[0]$ for all i . $B[0] \leq 12$	This combination indicates lossless coding for source data with bit depth ≤ 12 .
$B[0]$	0	5	$B[i] = B[0]$ for all i . $B[0] > 12$	This combination indicates lossless coding for source data with bit depth > 12 .
18	6	4	A NLT marker segment shall be present. Bit 2 or bit 3 of the CAP marker segment shall be 1, see subclause A.4.3	This combination is used in case a non-linearity is present.
20	8	4	Bits 2 and 3 of the CAP marker segment not present or shall be 0.	Regular case.

Table A.9 — Colour transformation

C_{pih}	Meaning
0	No colour transform
1	Reversible colour transformation (see Annex F)
3	Star-Tetrix transform (see Annex F)
2,4–15	Reserved for ISO/IEC purposes

Table A.10 — Quantizer type

Q_{pih}	Meaning
0	Deadzone quantizer (see Annex D)
1	Uniform quantizer (see Annex D)
2–3	Reserved for ISO/IEC purposes

Table A.11 — Sign handling strategy

<i>F_s</i>	Meaning
0	Signs encoded jointly with the data
1	Signs encoded separately
2–3	Reserved for ISO/IEC purposes

Table A.12 — Run mode

<i>R_m</i>	Meaning
0	Runs indicate zero prediction residuals
1	Runs indicate zero coefficients
2–3	Reserved for ISO/IEC purposes

Table A.13 — Progression order

<i>P_{poc}</i>	Meaning
0	The progression order as defined by subclause B.7 (resolution-line-band-component)
1–7	Reserved for ISO/IEC purposes

Table A.14 — Slice coding mode

<i>F_{slc}</i>	Meaning
0	The wavelet transformation runs across slice boundaries
1	Reserved for ISO/IEC purposes

A.4.5 Component table

[Table A.15](#) defines the syntax of the component table.

Function: This marker segment specifies the component precision and the sampling factors of each component in the image. The number of components itself is given by the N_c parameter of the picture header.

Usage: There shall be exactly one component table in each JPEG XS codestream. It shall precede the first slice header.

Table A.15 — Component table syntax

Syntax	Notes	Size	Values
<code>component_table() {</code>			
CDT		u(16)	0xff13
Lcdt	Size of the segment in bytes, not including the marker	u(16)	$2 \times N_c + 2$
<code>for (c=0; c<N_c; c=c+1) {</code>	Loop over components. The number of components is specified in the picture header.		
B[c]	Bit precision of component c	u(8)	8–16
s_x[c]	Horizontal sampling factor of component c	u(4)	1 or 2 for components 1 and 2, 1 for all other components
s_y[c]	Vertical sampling factor of component c	u(4)	$1 - s_x[c]$
<code>}</code>	End of loop over components		
<code>}</code>			

A.4.6 Nonlinearity marker

[Table A.16](#) defines the syntax of the nonlinearity marker segment.

Function: Defines an optional non-linear transform to be applied after inverse multiple component transformation.

Usage: Zero or one nonlinearity marker segments may be present in a codestream. If present, any nonlinearity marker segment shall precede the first slice header in the codestream and shall follow the picture header. This marker shall not be present if $F_q=0$, i.e. in the lossless mode.

Table A.16 — Nonlinearity marker segment syntax

Syntax	Notes	Size	Values
<code>nonlinearity_marker() {</code>			
NLT		u(16)	0xff16
Lnlt	Size of the marker segment, not including the marker	u(16)	5 or 12
Tnlt	Type of the non-linearity	u(8)	See Table A.17
<code>if (Tnlt == 1) {</code>	Additional data for quadratic non-linearity		
σ	Sign bit of the DC offset	u(1)	0—1
α	Remaining bits of the DC offset	u(15)	0— $2^{15}-1$
$DCO = \alpha - \sigma \times 2^{15}$	Compute the <i>DCO</i> offset from two's complement representation.		
<code>}</code>			
<code>if (Tnlt == 2) {</code>	Additional data for extended non-linearity		
T1	Upper threshold for region 1, in units of B_w	u(32)	$1-2^{B_w}-1$
T2	Upper threshold for region 2, in units of B_w	u(32)	$1-2^{B_w}-1$
E	Exponent of the linear slope in region 2	u(8)	1—4
<code>}</code>			
<code>}</code>			

Table A.17 — Tnlt encoding

Tnlt	Meaning
1	Quadratic non-linearity
2	Extended non-linearity
All other values	Reserved for ISO/IEC use

A.4.7 Component-dependent wavelet decomposition marker

[Table A.18](#) defines the syntax of the component-dependent decomposition marker.

Function: Optionally suppresses the wavelet decomposition for one or more components. If this marker is not present, the value of S_d shall be 0.

Usage: Zero or one component-dependent wavelet decomposition marker segments may be present in a codestream. Shall only be present if $N_c > 3$. If present, shall precede the first slice header in the codestream and shall follow the picture header.

Table A.18 — Component-dependent wavelet decomposition marker segment syntax

Syntax	Notes	Size	Values
<code>cwd_marker() {</code>			
CWD		u(16)	0xff17
Lcwd	Size of the marker segment, not including the marker	u(16)	3
S_d	Number of components for which the wavelet decomposition is suppressed.	u(8)	1— N_c-1 Furthermore, S_d shall be selected such that all components that are excluded from the wavelet transformation have $s_x[c]=1$ and $s_y[c]=1$
<code>}</code>			

A.4.8 Colour transformation specification marker

[Table A.19](#) defines the syntax of the colour transformation specification marker segment.

Function: Defines parameters of the Star-Tetrix transformation.

Usage: Zero or one colour transformation specification markers shall be present in a codestream. Shall be present if and only if $C_{pih}=3$, i.e. if the Star-Tetrix transformation is in use. Shall precede the first slice header in the codestream and shall follow the picture header.

Table A.19 — Colour transformation specification marker segment syntax

Syntax	Notes	Size	Values
<code>cts_marker() {</code>			
CTS		u(16)	0xff18
Lcts	Size of the marker segment, not including the marker	u(16)	4
Reserved	Reserved for ISO/IEC purposes	u(4)	0
C_f	Size and extent of the transformation	u(4)	See Table A.20
e₁	Exponent of first chroma component	u(4)	0..3
e₂	Exponent of second chroma component	u(4)	0..3
<code>}</code>			

Table A.20 — C_f encoding

C _f	Meaning
0	Full transformation, access to the line below and above required
3	Restricted in-line transformation, no access to neighbouring lines
All other values	Reserved for ISO/IEC use

A.4.9 Component registration marker

[Table A.21](#) defines the syntax of the component registration marker segment.

Function: Defines the relative placement of the component to the sample grid. If not present, the components are placed at the vertices of the sampling grid.

Usage: Zero or one component registration markers shall be present in a codestream. If $C_{pih}=3$, exactly one CRG marker segment shall be present. If present, any CRG marker segment shall precede the first slice header in the codestream and shall follow the picture header.

Table A.21 — Component registration marker segment syntax

Syntax	Notes	Size	Values
<code>crg_marker() {</code>			
CRG		u(16)	0xff19
Lcrg	Size of the marker segment, not including the marker	u(16)	$2+4 \times N_c$
<code>for (c=0; c<N_c; c=c+1) {</code>	Loop over all components		
Xcrg[c]	Relative horizontal placement of component c to the sample grid points in units of 1/65536. 0 indicates placement at the horizontal position of the sample grid, a positive value a displacement to the right of the sample grid position. A value of 32768 corresponds to a placement mid-way between the sample grid point and the next sample grid point to the right.	u(16)	0—65535
Ycrg[c]	Relative vertical placement of component c to the sample grid points in units of 1/65536. 0 indicates placement at the vertical position of the sample grid, a positive value a displacement to the bottom of the sample grid position. A value of 32768 corresponds to a placement mid-way between the sample grid point and the next sample grid point below.	u(16)	0—65535
<code>}</code>	End of loop over all components		
<code>}</code>			

A.4.10 Temporal prediction control marker

[Table A.22](#) defines the syntax of the temporal prediction control marker segment.

Function: Defines which TDC selection groups are forcibly encoded without temporal prediction and identifies the quantization adjustment for TDC selection groups. If the TPC marker is not present, the inferred values of Q_{bi} and Q_{br} shall be 0, the inferred value of S_i shall be 8, and the inferred values of $Sh[b]$ shall be $-\infty$, indicating that no position matches the hash position. That is, if this marker is not present, the temporal prediction of TDC selection groups only depends on the TDC mode selection $D_1[p, b]$ in the precinct header, see [subclause C.2](#).

NOTE 1 The value for S_i is always equal to 8, even in the case that no TPC marker is present.

Usage: Zero or one temporal prediction control markers shall be present in a codestream.

Table A.22 — Temporal prediction control marker segment syntax

Syntax	Notes	Size	Values
<code>tpc_marker() {</code>			
TPC		u(16)	0xff1a
Ltpc	Size of the marker segment, not including the marker	u(16)	$4+2 \times N_L$
S_i	Size of a TDC selection group in code groups	u(8)	8
σ_{bi}	Sign bit of Q_{bi}	u(1)	0—1
α_{bi}	Remaining bits of Q_{bi}	u(3)	0—7

Table A.22 (continued)

Syntax	Notes	Size	Values
$Q_{bi} = \alpha_{bi} - \sigma_{bi} \times 2^3$	Compute the quantization adjustment of wavelet coefficient residuals that are intra coded due to TDC selection flags $Y[p, \lambda, b, k]$ and $D_i[p, b]$ TDC mode selection		
σ_{br}	Sign bit of Q_{br}	u(1)	0—1
α_{br}	Remaining bits of Q_{br}	u(3)	0—7
$Q_{br} = \alpha_{br} - \sigma_{br} \times 2^3$	Compute the quantization adjustment of wavelet coefficient residuals that are intra coded due to positional match, see subclause H.7 for the condition and details.		
for (b=0; b<N _L ; b=b+1) {	Loop over all bands		
$Y_h[b]$	Positional hash value of the intra-refresh of band b	u(8)	0—255
$S_h[b]$	Hash mask exponent for intra-refresh of band b	u(8)	1—7
}			
}			

NOTE 2 The hash value $Y_h[b]$ and the hash mask exponent $S_h[b]$ are used to identify coefficient positions that are forcefully coded without prediction from the frame buffer, see Annex [H.3](#) for details.

A.4.11 Extension marker

[Table A.23](#) defines the syntax of the extension marker segment.

Function: Extends the codestream by generic or vendor-specific metadata.

Usage: Zero or more extension marker segments may be present in a codestream. If present, any extension marker segment shall precede the first slice header in the codestream.

Table A.23 — Extension marker segment syntax

Syntax	Notes	Size	Values
extension_marker() {			
COM		u(16)	0xff15
Lcom	Size of the marker segment, not including the marker	u(16)	Variable, at least 4
Tcom	Type of the extension	u(16)	See Table A.24
Dcom	User-defined data	Variable	Variable
padding	Padding to an integer number of bytes	pad(8)	0
}			

Table A.24 — Tcom encoding

Tcom	Meaning
0x0000	Vendor of the encoder, Dcom is a zero-terminated, ISO 10646 encoded string identifying the vendor of the encoder
0x0001	Copyright statement that the codestream creator want to convey to users of the codestream. The interpretation of this statement is beyond the scope of this document. Dcom is a zero-terminated, ISO/IEC 10646 encoded string identifying this statement.
0x8000-0xffff	Vendor-specific information. Tcom identifies the type of extension and the vendor.
All other values	Reserved for ISO/IEC use

A.4.12 Weights table

[Table A.25](#) defines the syntax of the weights table.

Function: This marker segment contains parameters required to set the gain of each band relative to the precinct quantization. Together with the parameters in the precinct header, see [subclause C.2](#), this allows to determine the quantization of the wavelet coefficients in this band. Details on how to use these parameters are specified in [subclause C.6.2](#). The number of wavelet bands and the relation between band, component and wavelet filter type are specified in [Annex B](#).

Usage: There shall be exactly one weights table in each JPEG XS codestream. This marker shall appear before the first slice header in the codestream and shall follow the picture header.

Table A.25 — Weights table syntax

Syntax	Notes	Size	Values
<code>weights_table() {</code>			
WGT		u(16)	0xff14
Lwgt	Size of the segment in bytes, not including the marker	u(16)	Variable
<code>for (b=0; b < N_L; b = b+1) {</code>	Loop over all bands.		
<code>if (b' _x [b]) {</code>	Check whether band <i>b</i> exists		
G[b]	Gain of band <i>b</i>	u(8)	0—15
P[b]	Priority of band <i>b</i>	u(8)	0—255
<code>}</code>	End of test whether <i>b</i> exists.		
<code>}</code>	End of loop over bands		
<code>}</code>			

NOTE Example weights tables for various configurations are given in [Annex I](#). These configurations have been optimized for PSNR performance of the encoder. Other choices are possible and can result in improved visual quality for a certain viewing distance.

A.4.13 Refresh Weights table

[Table A.25](#) defines the syntax of the Refresh Weights table.

Function: This marker segment contains parameters required to set the gain of each band relative to the precinct quantization for wavelet bands that select explicitly intra-prediction for refresh purposes through the TDC selection flags $D_i[p, b]$ in the precinct header, see [subclause C.2](#). Together with the parameters in the precinct header, this allows to determine the quantization of the wavelet coefficients in this band. Details on how to use these parameters are specified in [subclause C.6.2](#). The number of wavelet bands and the relation between band, component and wavelet filter type are specified in [Annex B](#). If this marker is not present, the inferred values for $G_r[b]$ shall be $G[b]$ for all *b*, and the inferred values for $P_r[b]$ shall be $P[b]$ for all *b*.

Usage: Zero or one Refresh Weights tables shall be present in each JPEG XS codestream. This marker shall appear before the first slice header in the codestream and shall follow the picture header.

Table A.26 — Refresh Weights table syntax

Syntax	Notes	Size	Values
refresh_weights_table() {			
WGR		u(16)	0xff1b
Lwgr	Size of the segment in bytes, not including the marker	u(16)	Variable
for (b=0; b < N _L ; b = b+1) {	Loop over all bands.		
if (b' _x [b]) {	Check whether band <i>b</i> exists		
G_r [b]	Refresh-Gain of band <i>b</i>	u(8)	0—15
P_r [b]	Refresh-Priority of band <i>b</i>	u(8)	0—255
}	End of test whether <i>b</i> exists.		
}	End of loop over bands		
}			

NOTE The Refresh Weights table allows encoders to prioritize refreshed bands differently from bands for which the intra-mode is selected due to large temporal differences; such bands are usually given lower priorities than the weights in the (regular) Weights table.

A.4.14 Slice header

[Table A.27](#) defines the syntax of the slice header.

Function: This marker segment identifies the start of an intra-only slice and provides sufficient information for the order of the slices within the image.

Usage: A codestream contains one or more slice headers. The entropy coded data for one slice follows the slice header and extends either to the next slice header or the end of the codestream. Even though the slice header includes the relative order of the slice in the image, a codestream shall contain slices in incremental order, i.e. progressing from the top of the image to the bottom of the image.

This particular slice header identifies slices that only use intra coding tools, i.e. no data from a previous frame is required to decode the content of an intra-only slice. For wavelet coefficients within an intra-coded slice, the wavelet coefficients $c'[p, \lambda, b, x]$ are set to $c[p, \lambda, b, x]$.

Table A.27 — Slice header syntax

Syntax	Notes	Size	Values
slice_header() {			
SLH		u(16)	0xff20
Lslh	Size of the segment in bytes, not including the marker	u(16)	4
Ysl	Index of the slice regardless of the slice type, counting from line 0 (at the top of the image) downwards (towards the bottom of the image)	u(16)	0—(2 ¹⁶ −1)
Isl = 0	Indicate that the data in this slice is intra-only coded.		
}			

NOTE Slice indices ease to regain synchronization in case transmission errors corrupted the codestream.

A.4.15 TDC enabling slice header

[Table A.28](#) defines the syntax of the TDC enabling slice header.

Function: This marker segment identifies the start of a TDC slice and provides sufficient information for the order of the slices within the image.

Usage: Similar to the slice header specified in [subclause A.4.13](#), this marker introduces a slice, though all precincts within this slice can reference data from the frame buffer. The entropy coded data for one slice follows the slice header and extends either to the next slice header or the end of the codestream. Even though the slice header includes the relative order of the slice in the image, a codestream shall contain slices in incremental order, i.e. progressing from the top of the image to the bottom of the image.

The TDC enabling slice header identifies slices that can reference wavelet coefficients from the frame buffer; a per-band flag recorded in the precinct header, see [subclause C.2](#), indicates for entire bands within this slice whether wavelet coefficients are temporally predicted. For temporally predicted wavelet coefficients, the inverse temporal decorrelation procedure specified in [subclause H.4](#) is applied to reconstruct the wavelet coefficients $c'[p,\lambda,b,x]$ from the wavelet coefficient residuals $c[p,\lambda,b,x]$ decoded through the procedures specified in [Annex C](#). For intra-coded wavelet coefficients, $c'[p,\lambda,b,x]$ is set to $c[p,\lambda,b,x]$. Temporal prediction, and the selection of the prediction mechanism is specified in [Annex H](#).

Table A.28 — TDC enabling slice header syntax

Syntax	Notes	Size	Values
<code>tdc_slice_header() {</code>			
SLI		u(16)	0xff21
Lsli	Size of the segment in bytes, not including the marker	u(16)	4
Ysl	Index of the slice regardless of the slice type, counting from line 0 (at the top of the image) downwards (towards the bottom of the image)	u(16)	0— ($2^{16}-1$)
$I_{sl} = 1$	Indicate that the data in this slice can use temporal prediction.		
<code>}</code>			

Annex B (normative)

Image data structures

B.1 Dimensions of chroma subsampled image planes

An image consists of N_c components aligned along a regular rectangular sampling grid. Each component is a rectangular arrangement of integer sample values aligned to the sampling grid of the image. The samples of a component are not required to populate every possible position on the sampling grid. The horizontal spacing between samples of component i is denoted by $s_x[i]$ and is called the *horizontal sampling factor*. The vertical spacing of component i is denoted by $s_y[i]$ and is called the *vertical sampling factor*. Both, horizontal and vertical sampling factors, are signalled in the component table specified in [subclause A.4.5](#). [Figure B.1](#) provides an example of how samples are assigned to positions on the sampling grid.

The number of samples in every row of component i is given by

$$W_c[i] = \left\lceil \frac{W_f}{s_x[i]} \right\rceil$$

The number of samples in each column of component i is given by

$$H_c[i] = \left\lceil \frac{H_f}{s_y[i]} \right\rceil$$

W_f and H_f are the horizontal and vertical dimensions of the sampling grid of the image as signalled in the picture header.

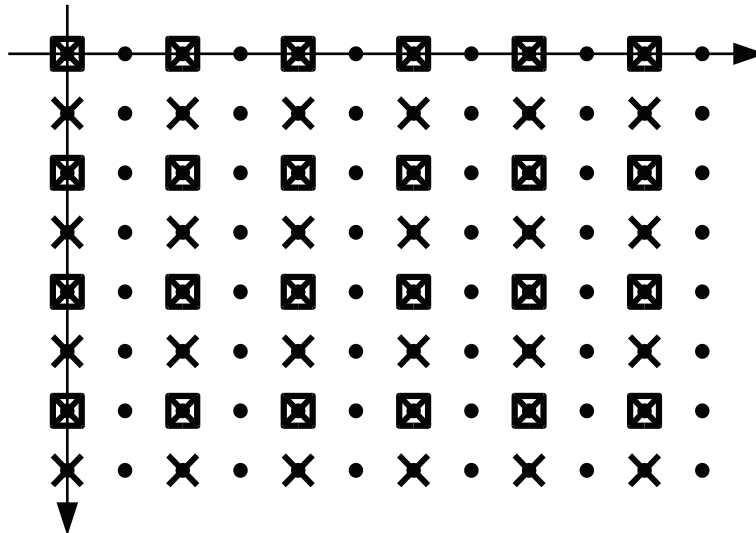


Figure B.1 — Sampling grid illustration