INTERNATIONAL STANDARD

ISO/IEC 15444-13

First edition
2008-07-15

Information technology — JPEG 2000 image coding system: An entry level JPEG 2000 encoder

Technologies de l'information — Système de codage d'images JPEG 2000: Un encodeur JPEG 2000 de niveau d'entrée

Reference number
ISO/IEC 15444-13:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

Page

1 Scope 1
1.1 Context 1
1.2 Requirements 1
2 References 1
2.1 Identical Recommendations | International Standards 1
3 Definitions 2
4 Abbreviations and symbols 5
4.1 Abbreviations 5
4.2 Symbols 5
5 General description 6
5.1 Codestream 6
5.2 Coding principles 6
6 Encoder requirements 8
6.1 General 8
6.2 Encoder function definition 8
6.3 Implementation 12
6.4 Codestream description 13
7 Optional file format requirements 13
Annex A – Codestream syntax 14
Annex B – Image and compressed image data ordering 15
Annex C – Arithmetic entropy coding 16
C.1 Binary encoding 16
C.2 Description of the arithmetic encoder 17
Annex D – Coefficient bit modelling 25
D.1 Code-block scan pattern within code-blocks 25
D.2 Coefficient bits and significance 25
D.3 Encoding passes over the bit-planes 26
D.4 Initializing and terminating 29
D.5 Error resilience segmentation symbol 31
D.6 Selective arithmetic coding bypass 31
D.7 Vertically causal context formation 32
D.8 Flow diagram of the code-block coding 32
Annex E – Quantization 35
E.1 Inverse quantization procedure (Informative) 35
E.2 Scalar coefficient quantization 36
Annex F – Discrete wavelet transformation of tile-components 37
F.1 Tile-component parameters 37
F.2 Discrete wavelet transformations 37
F.3 Forward transformation 37
F.4 Sub-sampling of components 45
F.5 Visual frequency weighting 45
Annex G – DC level shifting and multiple component transformations 48
G.1 DC level shifting of tile-components 48
G.2 Forward reversible multiple component transformation (RCT) 48
G.3 Forward irreversible multiple component transformation (ICT) 49
G.4 Chrominance component sub-sampling and the reference grid 49
Annex H – Coding of images with regions of interest 50
H.1 Description of the Maxshift method 50
H.2 Remarks on region of interest coding 51
Annex I – JP2 file format syntax 53

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 15444-13 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. The identical text is published as ITU-T Rec. T.812.

ISO/IEC 15444 consists of the following parts, under the general title *Information technology — JPEG 2000 image coding system*:

— *Part 1: Core coding system*

— *Part 2: Extensions*

— *Part 3: Motion JPEG 2000*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Compound image file format*

— *Part 8: Secure JPEG 2000*

— *Part 9: Interactivity tools, APIs and protocols*

— *Part 10: Extensions for three-dimensional data*

— *Part 11: Wireless*

— *Part 12: ISO base media file format*

— *Part 13: An entry level JPEG 2000 encoder*

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: An entry level JPEG 2000 encoder

1 Scope

This Recommendation | International Standard was developed by the Joint Photographic Experts Group (JPEG), the joint ISO/I–TU committee responsible for developing standards for continuous-tone still picture coding. It also refers to the Recommendations | International Standards produced by this committee: ITU-T Rec. T.81 | ISO/IEC 10918-1, ITU-T Rec. T.83 | ISO/IEC 10918-2, ITU-T Rec. T.84 | ISO/IEC 10918-3 and ITU-T Rec. T.87 | ISO/IEC 14495-1.

1.1 Context

This Recommendation | International Standard defines a set of lossless (bit-preserving) and lossy compression methods for coding bi-level, continuous-tone greyscale, palletized colour, or continuous-tone colour digital still images. This Recommendation | International Standard:

- specifies normative but optional encoding processes for converting source image data to JPEG 2000 compressed image data;
- specifies a complete encoding path to produce a conforming codestream as defined in Part 1 Annex A (ITU-T Rec. T.800 | ISO/IEC 15444-1);
- provides guidance on encoding processes for converting source image data to compressed image data;
- provides guidance on how to implement these processes in practice.

1.2 Requirements

This subclause contains a list of requirements for the definitions of an entry-level encoder.

An entry-level JPEG 2000 encoder implementation (this Recommendation | International Standard):

- shall be normative but optional; implementers shall be allowed to select necessary technologies/paths that would suite their application needs;
- shall define a JPEG 2000 Part 1 codestream encoder implementation; should define a JP2 file format encoder implementation;
- shall define a complete encoding path to produce a conforming codestream as defined in Annex A of ITU-T Rec. T.800 | ISO/IEC 15444-1:2004;
- shall consist of technology with clear IP status being royalty fee-free.

2 References

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation T.81 (1992) | ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*.
- ITU-T Recommendation T.84 (1996) | ISO/IEC 10918-3:1997, *Information technology – Digital compression and coding of continuous-tone still images: Extensions*.

- ITU-T Recommendation T.86 (1998) | ISO/IEC 10918-4:1999, *Information technology – Digital compression and coding of continuous-tone still images: Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF colour spaces, APPn markers, SPIFF compression types and Registration Authorities (REGAUT)*.
- ITU-T Recommendation T.87 (1998) | ISO/IEC 14495-1:2000, *Information technology – Lossless and near-lossless compression of continuous-tone still images: Baseline*.
- ITU-T Recommendation T.88 (2000) | ISO/IEC 14492:2001, *Information technology – Lossy/lossless coding of bi-level images*.
- ITU-T Recommendation T.800 (2002) | ISO/IEC 15444-1:2004, *Information technology – JPEG 2000 image coding system: Core coding system*.
- ITU-T Recommendation T.801 (2002) | ISO/IEC 15444-2:2004, *Information technology – JPEG 2000 image coding system: Extensions*.
- ITU-T Recommendation T.803 (2002) | ISO/IEC 15444-4:2004, *Information technology – JPEG 2000 image coding system: Conformance testing*.
- ITU-T Recommendation T.804 (2002) | ISO/IEC 15444-5:2003, *Information technology – JPEG 2000 image coding system: Reference software*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. The definitions defined in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 clause 3 also apply to this Recommendation | International Standard.

3.1 5-3 reversible filter (5-3R): A particular filter pair used in the wavelet transformation. This reversible filter pair has 5 taps in the low-pass and 3 taps in the high-pass.

3.2 9-7 irreversible filter (9-7I): A particular filter pair used in the wavelet transformation. This irreversible filter pair has 9 taps in the low-pass and 7 taps in the high-pass.

3.3 arithmetic coder: An entropy coder that converts variable length strings to variable length codes (encoding) and visa versa (decoding).

3.4 bit-plane: A two-dimensional array of bits. In this Recommendation | International Standard, a bit-plane refers to all the bits of the same magnitude in all coefficients or samples. This could refer to a bit-plane in a component, tile-component, code-block, region of interest, or other.

3.5 bit stream: The actual sequence of bits resulting from the coding of a sequence of symbols. It does not include the markers or marker segments in the main and tile-part headers or the EOC marker. It does include any packet headers and in stream markers and marker segments not found within the main or tile-part headers.

3.6 channel: One logical component of the image. A channel may be a direct representation of one component from the codestream, or may be generated by the application of a palette to a component from the codestream.

3.7 cleanup pass: A coding pass performed on a single bit-plane of a code-block of coefficients. The first pass and only coding pass for the first significant bit-plane is a cleanup pass; the third and the last pass of every remaining bit-plane is a cleanup pass.

3.8 codestream: A collection of one or more bit streams and the main header, tile-part headers, and the EOC required for their decoding and expansion into image data. This is the image data in a compressed form with all of the signalling needed to decode.

3.9 code-block: A rectangular grouping of coefficients from the same sub-band of a tile-component.

3.10 coder: An embodiment of either an encoding or decoding process.

3.11 coding pass: A complete pass through a code-block where the appropriate coefficient values and context are applied. There are three types of coding passes: significance propagation pass, magnitude refinement pass and cleanup pass. The result of each pass (after arithmetic coding, if selective arithmetic coding bypass is not used) is a stream of compressed image data.

3.12 coefficient: The values that are the result of a transformation.

3.13 component: A two-dimensional array of samples. An image typically consists of several components, for instance representing red, green and blue.

3.14 **context**: Function of coefficients previously decoded and used to condition the decoding of the present coefficient.

3.15 **decoder**: An embodiment of a decoding process, and optionally a colour transformation process.

3.16 **decoding process**: A process which takes as its input all or part of a codestream and outputs all or part of a reconstructed image.

3.17 **decomposition level**: A collection of wavelet sub-bands where each coefficient has the same spatial impact or span with respect to the source component samples. These include the HL, LH, and HH sub-bands of the same two dimensional sub-band decomposition. For the last decomposition level, the LL sub-band is also included.

3.18 **discrete wavelet transformation (DWT)**: A transformation that iteratively transforms one signal into two or more filtered and decimated signals corresponding to different frequency bands. This transformation operates on spatially discrete samples.

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

3.20 **encoding process**: A process, that takes as its input all or part of a source image data and outputs a codestream.

3.21 **file format**: A codestream and additional support data and information not explicitly required for the decoding of codestream. Examples of such support data include text fields providing titling, security and historical information, data to support placement of multiple codestream within a given data file, and data to support exchange between platforms or conversion to other file formats.

3.22 **guard bits**: Additional most significant bits that have been added to sample data.

3.23 **header**: Either a part of the codestream that contains only markers and marker segments (main header and tile part header) or the signalling part of a packet (packet header).

3.24 **image area**: A rectangular part of the reference grid, registered by offsets from the origin and the extent of the reference grid.

3.25 **image area offset**: The number of reference grid points down and to the right of the reference grid origin where the origin of the image area can be found.

3.26 **irreversible**: A transformation, progression, system, quantization, or other process that, due to systemic or quantization error, disallows lossless recovery. An irreversible process can only lead to lossy compression.

3.27 **JP2 file**: The name of a file in the file format described in this Recommendation | International Standard. Structurally, a JP2 file is a contiguous sequence of boxes.

3.28 **JPEG**: Used to refer globally to the encoding and decoding process of the following Recommendations | International Standards:

- ITU-T Recommendation T.81 (1992) | ISO/IEC 10918-1:1994, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*.
- ITU-T Recommendation T.83 (1994) | ISO/IEC 10918-2:1995, *Information technology – Digital compression and coding of continuous-tone still images: Compliance testing*.
- ITU-T Recommendation T.84 (1996) | ISO/IEC 10918-3:1997, *Information technology – Digital compression and coding of continuous-tone still images: Extensions, plus Amendment 1 (1999), Provisions to allow registration of new compression types and versions in the SPIFF header.*
- ITU-T Recommendation T.86 (1998) | ISO/IEC 10918-4:1999, *Information technology – Digital compression and coding of continuous-tone still images: Registration of JPEG Profiles, SPIFF Profiles, SPIFF Tags, SPIFF Colour Spaces, APPn Markers, SPIFF Compression Types and Registration Authorities (REGAUT).*

3.29 **JPEG 2000**: Used to refer globally to the encoding and decoding processes in this Recommendation | International Standard and their embodiment in applications.

3.30 **JPX file**: JPEG 2000 File Format defined in ITU-T Rec. T.801 | ISO/IEC 15444-2:2004; JPEG 2000 File Format Extended.

3.31 **layer**: A collection of compressed image data from coding passes of one, or more, code-blocks of a tile component. Layers have an order for encoding and decoding that must be preserved.

3.32 **lossless**: A descriptive term for the effect of the overall encoding and decoding processes in which the output of the decoding process is identical to the input to the encoding process. Distortion free restoration can be assured. All of the coding processes or steps used for encoding and decoding are reversible.

3.33 **lossy**: A descriptive term for the effect of the overall encoding and decoding processes in which the output of the decoding process is not identical to the input to the encoding process. There is distortion (measured mathematically). At least one of the coding processes or steps used for encoding and decoding is irreversible.

3.34 **magnitude refinement pass**: A type of coding pass.

3.35 **main header**: A group of markers and marker segments at the beginning of the codestream that describe the image parameters and coding parameters that can apply to every tile and tile-component.

3.36 **marker**: A two-byte code in which the first byte is hexadecimal FF (0xFF) and the second byte is a value between 1 (0x01) and hexadecimal FE (0xFE).

3.37 **marker segment**: A marker and associated (not empty) set of parameters.

3.38 **packet**: A part of the bit stream comprising a packet header and the compressed image data from one layer of one precinct of one resolution level of one tile-component.

3.39 **packet header**: Portion of the packet that contains signalling necessary for decoding that packet.

3.40 **precinct**: A one rectangular region of a transformed tile-component, within each resolution level, used for limiting the size of packets.

3.41 **precision**: Number of bits allocated to a particular sample, coefficient, or other binary numerical representation.

3.42 **progression**: The order of a codestream where the decoding of each successive bit contributes to a "better" reconstruction of the image. What metrics make the reconstruction "better" is a function of the application. Some examples of progression are increasing resolution or improved sample fidelity.

3.43 **quantization**: A method of reducing the precision of the individual coefficients to reduce the number of bits used to entropy-code them. This is equivalent to division while compressing and multiplying while decompressing. Quantization can be achieved by an explicit operation with a given quantization value or by dropping (truncating) coding passes from the codestream.

3.44 **reference grid**: A regular rectangular array of points used as a reference for other rectangular arrays of data. Examples include components and tiles.

3.45 **region of interest (ROI)**: A collections of coefficients that are considered of particular relevance by some user-defined measure.

3.46 **resolution level**: Equivalent to decomposition level with one exception: the LL sub-band is also a separate resolution level.

3.47 **reversible**: A transformation, progression, system, or other process that does not suffer systemic or quantization error and, therefore, allows lossless signal recovery.

3.48 **segmentation symbol**: A special symbol coded with a uniform context at the end of each coding pass for error resilience.

3.49 **selective arithmetic coding bypass**: A coding style where some of the code-block passes are not coded by the arithmetic coder. Instead the bits to be coded are appended directly to the bit stream without coding.

3.50 **significance propagation pass**: A coding pass performed on a single bit-plane of a code-block of coefficients.

3.51 **sub-band**: A group of transform coefficients resulting from the same sequence of low-pass and high-pass filtering operations, both vertically and horizontally.

3.52 **tile**: A rectangular array of points on the reference grid, registered with and offset from the reference grid origin and defined by a width and height. The tiles that overlap are used to define tile-components.

3.53 **tile-component**: All the samples of a given component in a tile.

3.54 **tile index**: The index of the current tile ranging from zero to the number of tiles minus one.

3.55 **tile-part**: A portion of the codestream with compressed image data for some, or all, of a tile. The tile-part includes at least one, and up to all, of the packets that make up the coded tile.

3.56 **tile-part header**: A group of markers and marker segments at the beginning of each tile-part in the codestream that describe the tile-part coding parameters.

3.57 **tile-part index**: The index of the current tile-part ranging from zero to the number of tile-parts minus one in a given tile.

3.58 **transformation**: A mathematical mapping from one signal space to another.

4 Abbreviations and symbols

4.1 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply. The abbreviations defined in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 clause 4 also apply to this Recommendation | International Standard.

1D-DWT	One-dimensional Discrete Wavelet Transformation
FDWT	Forward Discrete Wavelet Transformation
ICC	International Colour Consortium
ICT	Irreversible Component Transformation
IDWT	Inverse Discrete Wavelet Transformation
JPEG	Joint Photographic Experts Group
JURA	JPEG Utilities Registration Authority
RCT	Reversible Component Transformation
ROI	Region of Interest

4.2 Symbols

For the purposes of this Recommendation | International Standard, the following symbols apply. The abbreviations defined in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004 clause 4 also apply to this Recommendation | International Standard.

0x----	Denotes a hexadecimal number
nnn	A three-digit number preceded by a backslash indicates the value of a single byte within a character string, where the three digits specify the octal value of that byte
ε_b	Exponent of the quantization value for a sub-band defined in QCD and QCC
μ_b	Mantissa of the quantization value for a sub-band defined in QCD and QCC
M_b	Maximum number of bit-planes coded in a given code-block
N_L	Number of decomposition levels as defined in COD and COC
R_b	Dynamic range of a component sample as defined in SIZ
COC	Coding style component marker
COD	Coding style default marker
COM	Comment marker
CRG	Component registration marker
EOC	End of codestream marker
EPH	End of packet header marker
PLM	Packet length, main header marker
PLT	Packet length, tile-part header marker
POC	Progression order change marker
PPM	Packed packet headers, main header marker
PPT	Packed packet headers, tile-part header marker
QCC	Quantization component marker
QCD	Quantization default marker
RGN	Region-of-interest marker
SIZ	Image and tile size marker

SOC	Start of codestream marker
SOD	Start of data marker
SOP	Start of packet marker
SOT	Start of tile-part marker
TLM	Tile-part lengths marker

5 General description

5.1 Codestream

The definition of codestream is the same as in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

5.2 Coding principles

The main procedures for this Recommendation | International Standard are shown in Figure 1. This shows the encoding order, a reverse ordering of the block diagram of Figure 5-1 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

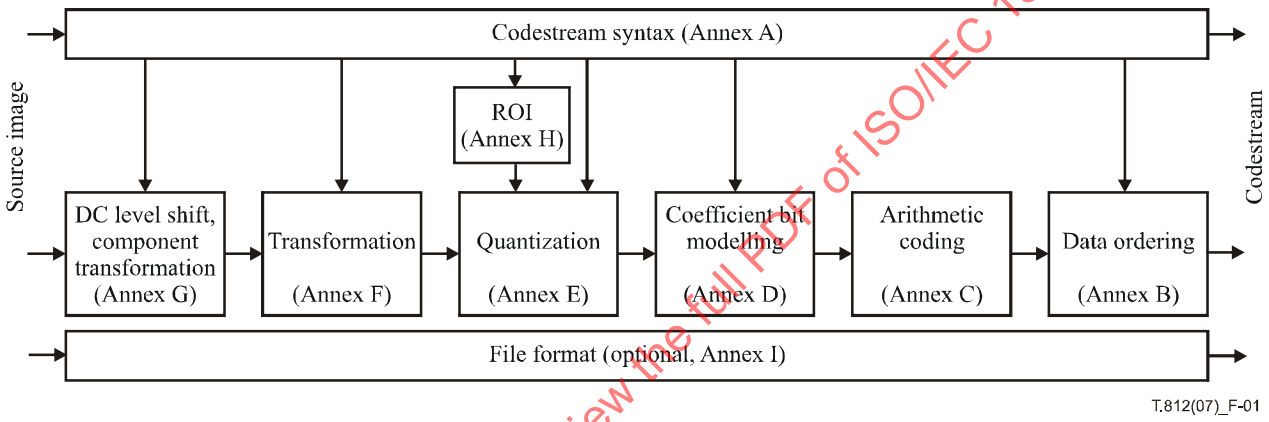

Figure 1 – Specification block diagram

Procedures are presented in the annexes. The category of the annexes is the same as in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004. Also the index of the annex is the same.

The coding process is summarized below.

Many images have multiple components. This Specification has a multiple component transformation to correlate three components. This is the only function in this Specification that relates components to each other. (See Annex G.)

The image components may be divided into tiles. These tile-components are rectangular arrays that relate to the same portion of each of the components that make up the image. Thus, tiling of the image actually creates tile-components that can be encoded independently of each other. This tile independence provides one of the methods for encoding a region of the image. (See Annex B.)

The tile-components are composed into different decomposition levels using a wavelet transformation. These decomposition levels contain a number of sub-bands populated with coefficients that describe the horizontal and vertical spatial frequency characteristics of the original tile-components. The coefficients provide frequency information about a local area, rather than across the entire image like the Fourier transformation. A decomposition level is related to the next decomposition level by a spatial factor of two. That is, each successive decomposition level of the sub-bands has approximately half the horizontal and half the vertical resolution of the previous. (See Annex F.)

Although there are as many coefficients as there are samples, the information content tends to be concentrated in just a few coefficients. Through quantization, the information content of a large number of small-magnitude coefficients is further reduced (Annex E). Additional processing by the entropy coder reduces the number of bits required to represent these quantized coefficients, sometimes significantly compared to the original image. (See Annexes C, D and B.)

The individual sub-bands of a tile-component are further divided into code-blocks. These rectangular arrays of coefficients can be extracted independently. The individual bit-planes of the coefficients in a code-block are coded with three coding passes. Each of these coding passes collects contextual information about the bit-plane compressed image data. (See Annex D.)

An arithmetic coder uses this contextual information, and its internal state, to encode coefficients. (See Annex C.) Different termination mechanisms allow different levels of independent extraction of this coding pass compressed image data.

The bit stream compressed image data created from these coding passes is grouped in layers. Layers are arbitrary groupings of coding passes from code-blocks. (See Annex B.)

NOTE – Although there is great flexibility in layering, the premise is that each successive layer contributes to a higher quality image.

Sub-band coefficients at each resolution level are partitioned into rectangular areas called precincts. (See Annex B.)

Packets are a fundamental unit of the compressed codestream. A packet contains compressed image data from one layer of a precinct of one resolution level of one tile-component. Packets provide another method for extracting a spatial region independently from the codestream. These packets are interleaved in the codestream using a few different methods. (See Annex B.)

A mechanism is provided that allows the compressed image data corresponding to regions of interest in the original tile components to be coded and placed earlier in the bit stream. (See Annex H.)

Several mechanisms are provided to allow the detection and concealment of bit errors that might occur over a noisy transmission channel. (See Annex D.)

The codestream relating to a tile, organized in packets, are arranged in one, or more, tile-parts. A tile-part header, comprised of a series of markers and marker segments, contains information about the various mechanisms and coding styles that are needed to locate, extract, decode, and reconstruct every tile-component. At the beginning of the entire codestream is a main header, comprised of markers and marker segments, that offers similar information as well as information about the original image. (See Annex A.)

The codestream is optionally wrapped in a file format that allows applications to interpret the meaning of, and other information about, the image. The file format may contain data besides the codestream. (See Annex I.)

To review, procedures that divide the original image are the following:

- The components of the image are divided into rectangular tiles. The tile-component is the basic unit of the original or reconstructed image.
- Performing the wavelet transformation on a tile-component creates decomposition levels.
- These decomposition levels are made up of sub-bands of coefficients that describe the frequency characteristics of local areas (rather than across the entire tile-component) of the tile-component.
- The sub-bands of coefficients are quantized and collected into rectangular arrays of code-blocks.
- Each bit-plane of the coefficients in a code-block is entropy coded in three types of coding passes.
- Some of the coefficients can be coded first to provide a region of interest.

At this point the image data is fully converted to compressed image data. The procedures that reassemble these bit stream units into the codestream are the following:

- The compressed image data from the coding passes are collected in layers.
- Packets are composed compressed image data from one precinct of a single layer of a single resolution level of a single tile-component. The packets are the basic unit of the compressed image data.
- All the packets from a tile are interleaved in one of several orders and placed in one, or more, tile-parts.
- The tile-parts have a descriptive tile-part header and can be interleaved in some orders.
- The codestream has a main header at the beginning that describes the original image and the various decomposition and coding styles.
- The optional file format describes the meaning of the image and its components in the context of the application.

6 Encoder requirements

6.1 General

An encoding process converts source image data to compressed image data. Annexes A, B, C, D, E, F, G and H describe the encoding process. All encoding processes are normative. An encoder is an embodiment of the encoding process. In order to conform to this Recommendation | International Standard, an encoder shall convert source image data to compressed image data, that conform to the codestream syntax specified in Annex A.

There is no required specific implementation for the encoder. In some cases, the descriptions use particular implementation techniques for illustrative purposes only.

The definition of an entry-level encoder is separated into three groups:

- encoder function;
- implementation;
- codestream description.

6.2 Encoder function definition

There are four encoding styles described in this Recommendation | International Standard:

- lossless based colour encoding;
- lossy based colour encoding;
- lossless based greyscale encoding;
- lossy based greyscale encoding.

The definition of image and compressed data ordering is applied commonly above four styles. Table 1 shows the definition of image and compressed data ordering. All reference of Table 1 is described in Annex B in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Table 1 – Function definition of image and compressed data ordering

Item	Reference
Reference grid	B.2*
Tiling	B.3*
Sub-band	B.5*
Precinct	B.6*
Code-block (64x64 or 32x32)	B.7*
Layer	B.8*
Packet	B.9*
Packet header coding	B.10*
Tile-parts (plural)	B.11*
Part 1 progression order (one of 5 progression methods)	B.12.1*
Progression order change	B.12.2, B.12.3*
* Described in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.	

6.2.1 Lossless based colour encoding

Lossless based colour encoding process is shown in Figure 2. Because a basic concept of lossless based colour encoding is colour image exchange lossless, reversible component transformation (RCT) and reversible wavelet transformation (5-3R) shall be executed. Also scalar quantization shall not be executed. It may be used in truncated codestream to provide lossy compression.

Function definition of Lossless based colour encoding is shown in Table 2.

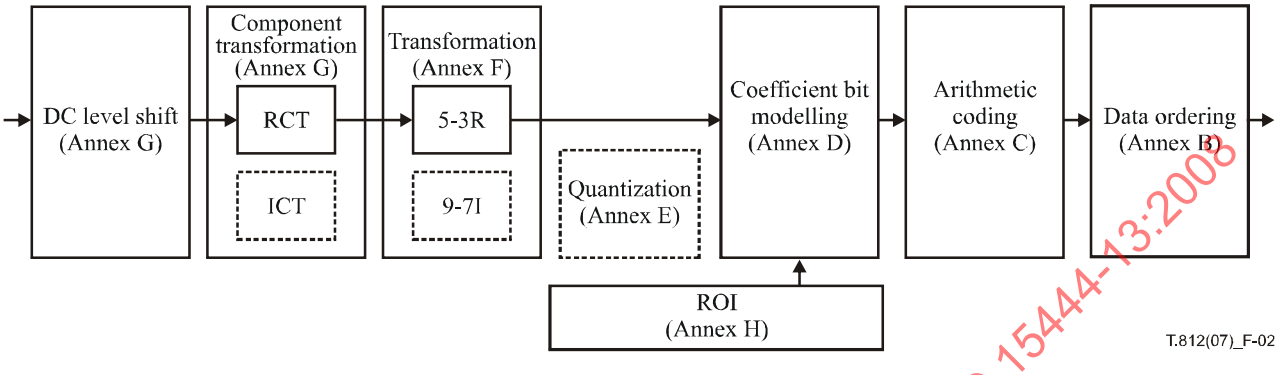

Figure 2 – Lossless based colour encoding block diagram

Table 2 – Function definition of lossless based colour encoding

Category	Item	Reference
DC level shifting and multiple component transformations	DC level shifting data	G.1
	RCT	G.2
Discrete wavelet transform	Wavelet transform (5-3R)	F.3
	Sub-sampling by discarding coefficients	F.4
	Visual frequency weighting	F.5
Coefficient bit modelling	3-coding pass, 4-coding mode (EBCOT)	D.3
	Predictable termination	D.4
	Reset context probabilities on coding pass boundaries	D.4
	Termination on each coding pass	D.4
	Segmentation symbols	D.5
	Selective AC bypass	D.6
	Vertically causal context	D.7
Arithmetic entropy coding	MQ-Coder Encoding	C.2
Region of Interest	Max-shift	H.1, H.2

6.2.2 Lossy based colour encoding

Lossy-based colour encoding process is shown in Figure 3. Lossy-based colour encoding has several purposes such as high compression performance or fixed codestream size. Irreversible component transformation (ICT) and irreversible wavelet transformation (9-7I) shall be executed. Scalar quantization may be executed.

Function definition of Lossy based colour encoding is shown in Table 3.

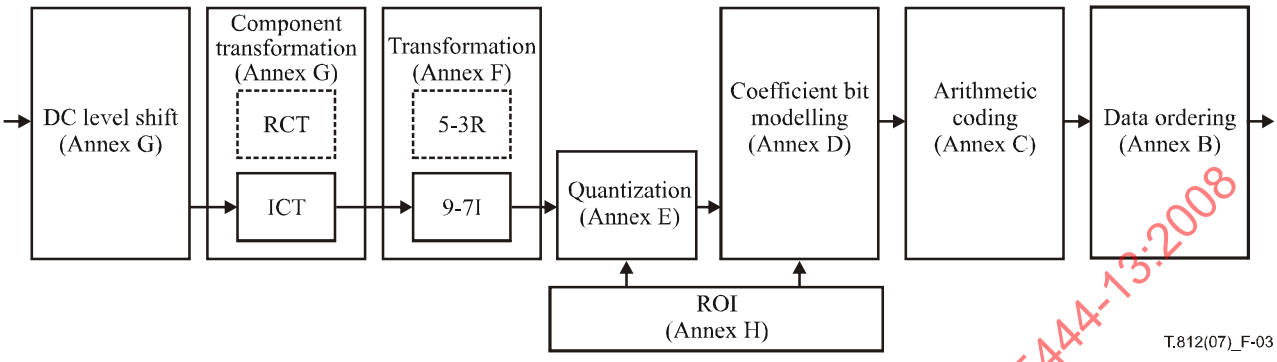

Figure 3 – Lossy based colour encoding block diagram

Table 3 – Function definition of lossy based colour encoding

Category	Item	Reference
DC level shifting and multiple component transformations	DC level shifting data	G.1
	ICT	G.3
Discrete wavelet transform	Wavelet transform (9-7I)	F.3
	Sub-sampling by discarding coefficients	F.4
	Visual frequency weighting	F.5
Quantization	Scalar quantization	E.2
Coefficient bit modelling	3-coding pass, 4-coding mode (EBCOT)	D.3
	Predictable termination	D.4
	Reset context probabilities on coding pass boundaries	D.4
	Termination on each coding pass	D.4
	Segmentation symbols	D.5
	Selective AC bypass	D.6
	Vertically causal context	D.7
Arithmetic entropy coding	MQ-Coder Encoding	C.2
Region of Interest	Max-shift	H.1, H.2

6.2.3 Lossless based greyscale encoding

Lossless based greyscale encoding process is shown in Figure 4. A basic concept of lossless based greyscale encoding is almost the same as lossless based colour encoding, but source image is changed colour to greyscale. Reversible wavelet transformation (5-3R) shall be executed. Component transformation and scalar quantization shall not be executed. It may be used in truncated codestream to provide lossy compression.

Function definition of lossless based greyscale encoding is shown in Table 4.

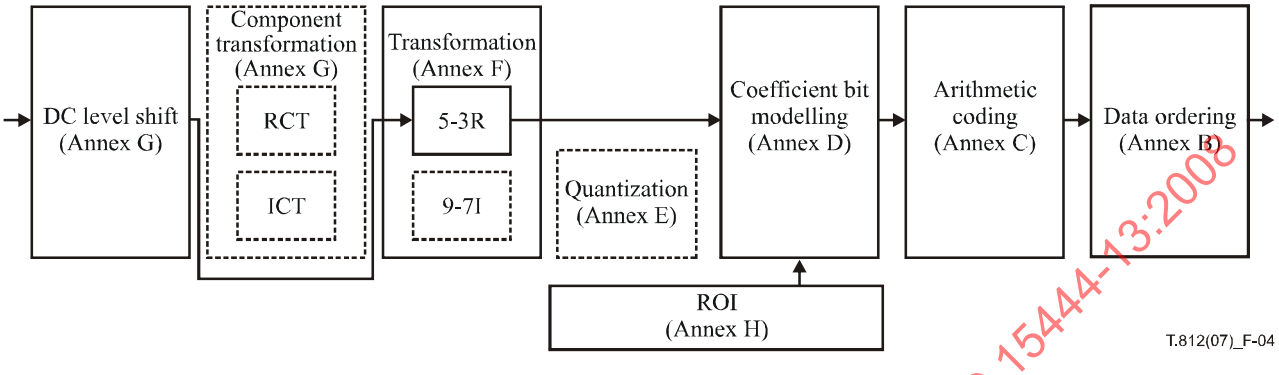

Figure 4 – Lossless based greyscale encoding block diagram

Table 4 – Function definition of lossless based greyscale encoding

Category	Item	Reference
DC level shifting and multiple component transformations	DC level shifting data	G.1
Discrete wavelet transform	Wavelet transform (5-3R)	F.3
	Visual frequency weighting	F.5
Coefficient bit modelling	3-coding pass, 4-coding mode (EBCOT)	D.3
	Predictable termination	D.4
	Reset context probabilities on coding pass boundaries	D.4
	Termination on each coding pass	D.4
	Segmentation symbols	D.5
	Selective AC bypass	D.6
	Vertically causal context	D.7
Arithmetic entropy coding	MQ-Coder Encoding	C.2
Region of Interest	Max-shift	H.1, H.2

6.2.4 Lossy based greyscale encoding

Lossy-based greyscale encoding process is shown in Figure 5. A concept of lossy-based greyscale encoding is almost the same as lossy based colour encoding, but source image is changed colour to greyscale. Irreversible wavelet transformation (9-7I) shall be executed. Scalar quantization may be executed. Component transformation shall not be executed.

Function definition of lossy based greyscale encoding is shown in Table 5.

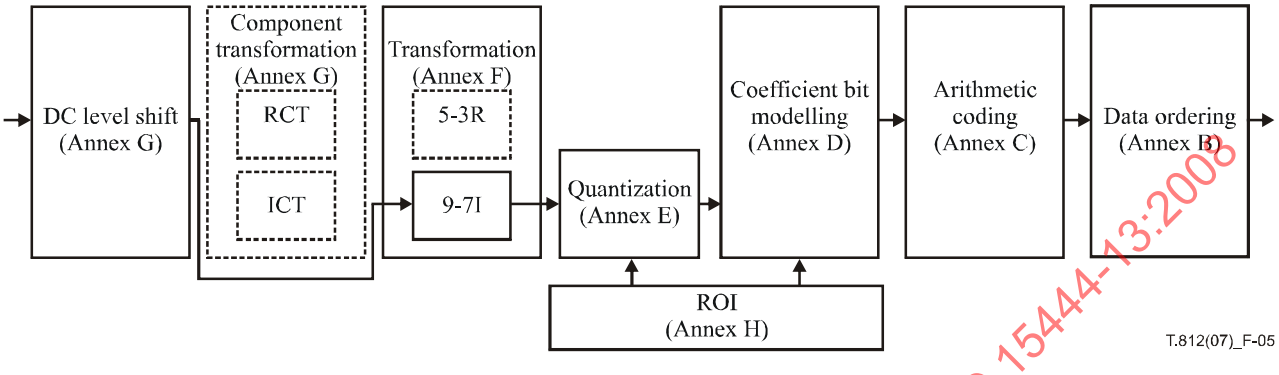

Figure 5 – Lossy based greyscale encoding block diagram

Table 5 – Function definition of lossy based greyscale encoding

Category	Item	Reference
DC level shifting and multiple component transformations	DC level shifting data	G.1
Discrete wavelet transform	Wavelet transform (9-7I)	F.3
	Visual frequency weighting	F.5
Quantization	Scalar quantization	E.2
Coefficient bit modelling	3-coding pass, 4-coding mode (EBCOT)	D.3
	Predictable termination	D.4
	Reset context probabilities on coding pass boundaries	D.4
	Termination on each coding pass	D.4
	Segmentation symbols	D.5
	Selective AC bypass	D.6
	Vertically causal context	D.7
Arithmetic entropy coding	MQ-Coder Encoding	C.2
Region of Interest	Max-shift	H.1, H.2

6.3 Implementation

There is no required specific implementation for this Recommendation | International Standard. This subclause defines optional implementation choices.

Implementation definition is shown in Table 6. All reference of Table 6 is described in Annex J in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Table 6 – Implementation definition

Category	Item	Reference
Implementation	Row-based wavelet transform	J.5*
	Visual frequency weighting	J.12*
	Rate control	J.14*
* Described in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.		

6.4 Codestream description

Any compressed image data shall comply with the syntax and code assignments appropriate for the coding processes defined in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

ITU-T Rec. T.803 | ISO/IEC 15444-4:2004 (JPEG 2000 Part 4) describes a definition of compliance or conformance.

This subclause defines the use of a particular marker segment for an encoder.

Codestream description is shown in Table 7. All reference of Table 7 is described in Annex A in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Table 7 – Codestream description

Category	Item	Reference
Implementation	SOC marker	A.4.1*
	SOT marker segment	A.4.2*
	SOD marker	A.4.3*
	EOC marker	A.4.4*
	SIZ marker segment	A.5.1*
	COD marker segment	A.6.1*
	COC marker segment	A.6.2*
	RGN marker segment	A.6.3*
	QCD marker segment	A.6.4*
	QCC marker segment	A.6.5*
	POC marker segment	A.6.6*
	TLM marker segment	A.7.1*
	PLM marker segment	A.7.2*
	PLT marker segment	A.7.3*
	PPM marker segment	A.7.4*
	PPT marker segment	A.7.5*
	SOP marker segment	A.8.1*
	EPH marker segment	A.8.2*
	CRG marker segment	A.9.1*
	COM marker segment	A.9.2*
* Described in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.		

7 Optional file format requirements

Annex I describes the optional file format containing metadata about the image in addition to the codestream. This data allows, for example, screen display or printing at a specific resolution. The optional file format, when used, shall comply with the file format syntax and code assignments appropriate for the coding processes defined in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Annex A

Codestream syntax

(This annex forms an integral part of this Recommendation | International Standard)

There is no additional codestream syntax of JPEG 2000 codestream.

This annex is identical to Annex A in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Annex B

Image and compressed image data ordering

(This annex forms an integral part of this Recommendation | International Standard)

There is no additional definition of image and compressed data ordering.

This annex is identical to Annex B in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

Annex C

Arithmetic entropy coding

(This annex forms an integral part of this Recommendation | International Standard)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

C.1 Binary encoding

Figure C.1 shows a simple block diagram of the binary adaptive arithmetic encoder. The decision (D) and context (CX) pairs are processed together to produce compressed image data (CD) output. Both D and CX are provided by the model unit (not shown). CX selects the probability estimate to use during the coding of D. In this Recommendation | International Standard, CX is a label for a context.

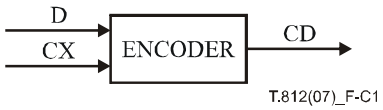

Figure C.1 – Arithmetic encoder inputs and outputs

C.1.1 Recursive interval subdivision

The recursive probability interval subdivision of Elias coding is the basis for the binary arithmetic coding process. With each binary decision the current probability interval is subdivided into two sub-intervals, and the code string is modified (if necessary) so that it points to the base (the lower bound) of the probability sub-interval assigned to the symbol that occurred.

In the partitioning of the current interval into two sub-intervals, the sub-interval for the more probable symbol (MPS) is ordered above the sub-interval for the less probable symbol (LPS). Therefore, when the MPS is coded, the LPS sub-interval is added to the code string. This coding convention requires that symbols be recognized as MPS or LPS, rather than 0 or 1. Consequently, the size of the LPS interval and the sense of the MPS for each decision must be known in order to code that decision.

Since the code string always points to the base of the current interval, the decoding process is a matter of determining, for each decision, which sub-interval is pointed to by the compressed image data. This is also done recursively, using the same interval sub-division process as in the encoder. Each time a decision is decoded, the decoder subtracts any interval the encoder added to the code string. Therefore, the code string in the decoder is a pointer into the current interval relative to the base of the current interval. Since the coding process involves addition of binary fractions rather than concatenation of integer code words, the more probable binary decisions can often be coded at a cost of much less than one bit per decision.

C.1.2 Coding conventions and approximations

The coding operations are done using fixed precision integer arithmetic and using an integer representation of fractional values in which 0x8000 is equivalent to decimal 0.75. The interval A is kept in the range $0.75 \le A < 1.5$ by doubling it whenever the integer value falls below 0x8000.

The code register C is also doubled each time A is doubled. Periodically – to keep C from overflowing – a byte of compressed image data is removed from the high order bits of the C-register and placed in an external compressed image data buffer. Carry-over into the external buffer is prevented by a bit-stuffing procedure.

Keeping A in the range $0.75 \le A < 1.5$ allows a simple arithmetic approximation to be used in the interval subdivision. The interval is A and the current estimate of the LPS probability is Qe, a precise calculation of the sub-intervals would require:

$$A - (Qe * A) = \text{sub-interval for the MPS} \qquad \text{(C-1)}$$

$$Qe * A = \text{sub-interval for the LPS} \qquad \text{(C-2)}$$

Because the value of A is of order unity, these are approximated by:

$$A - Qe = \text{sub-interval for the MPS} \quad \text{(C-3)}$$

$$Qe = \text{sub-interval for the LPS} \quad \text{(C-4)}$$

Whenever the MPS is coded, the value of Qe is added to the code register and the interval is reduced to A – Qe. Whenever the LPS is coded, the code register is left unchanged and the interval is reduced to Qe. The precision range required for A is then restored, if necessary, by renormalization of both A and C.

With the process illustrated above, the approximations in the interval subdivision process can sometimes make the LPS sub-interval larger than the MPS sub-interval. If, for example, the value of Qe is 0.5 and A is at the minimum allowed value of 0.75, the approximate scaling gives 1/3 of the interval to the MPS and 2/3 to the LPS. To avoid this size inversion, the MPS and LPS intervals are exchanged whenever the LPS interval is larger than the MPS interval. This MPS/LPS conditional exchange can only occur when a renormalization is needed.

Whenever a renormalization occurs, a probability estimation process is invoked which determines a new probability estimate for the context currently being coded. No explicit symbol counts are needed for the estimation. The relative probabilities of renormalization after coding an LPS or MPS provide an approximate symbol counting mechanism which is used to directly estimate the probabilities.

C.2 Description of the arithmetic encoder

The ENCODER (Figure C.2) initializes the encoder through the INITENC procedure. CX and D pairs are read and passed on to ENCODE until all pairs have been read. The probability estimation procedures that provide adaptive estimates of the probability for each context are imbedded in ENCODE. Bytes of compressed image data are output when necessary. When all of the CX and D pairs have been read, FLUSH sets the contents of the C-register to as many 1 bits as possible and then outputs the final bytes. FLUSH also terminates the encoding and generates the required terminating marker.

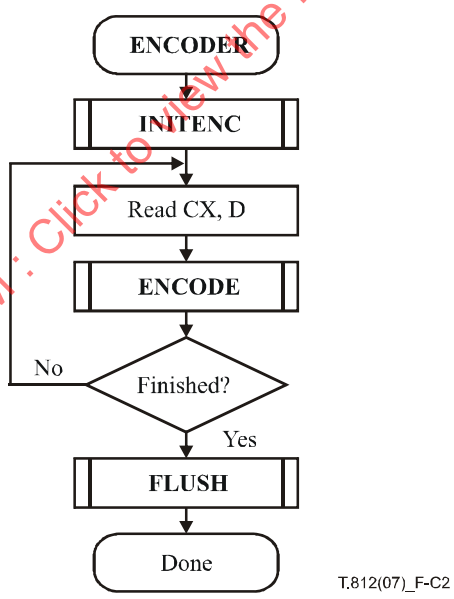

Figure C.2 – Encoder for the MQ-coder

C.2.1 Encoder code register conventions

The flow charts given in this annex assume the register structures for the encoder shown in Table C.1.

Table C.1 – Encoder register structures

	MSB			LSB
C-register	0000 cbbb	bbbb bsss	xxxx xxxx	xxxx xxxx
A-register	0000 0000	0000 0000	1aaa aaaa	aaaa aaaa

The "a" bits are the fractional bits in the A register (the current interval value) and the "x" bits are the fractional bits in the code register. The "s" bits are spacer bits which provide useful constraints on carry-over, and the "b" bits indicate the bit positions from which the completed bytes of the compressed image data are removed from the C-register. The "c" bit is a carry bit. The detailed description of bit stuffing and the handling of carry-over will be given in a later part of this annex.

C.2.2 Encoding a decision (ENCODE)

The ENCODE procedure determines whether the decision D is a 0 or not. Then a CODE0 or a CODE1 procedure is called appropriately. Often embodiments will not have an ENCODE procedure, but will call the CODE0 or CODE1 procedures directly to code a 0-decision or a 1-decision. Figure C.3 shows this procedure.

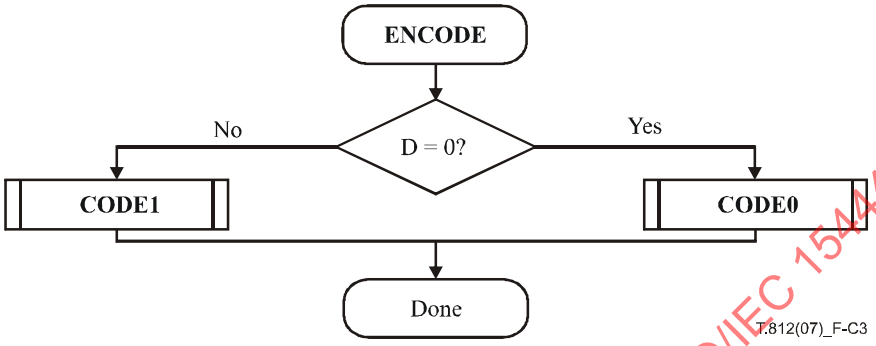

Figure C.3 – ENCODE procedure

C.2.3 Encoding a 1 or a 0 (CODE1 and CODE0)

When a given binary decision is coded, one of two possibilities occurs – the symbol is either the more probable symbol or it is the less probable symbol. CODE1 and CODE0 are illustrated in Figures C.4 and C.5. In these figures, CX is the context. For each context, the index of the probability estimate that is to be used in the coding operations and the MPS value are stored. MPS (CX) is the sense (0 or 1) of the MPS for context CX.

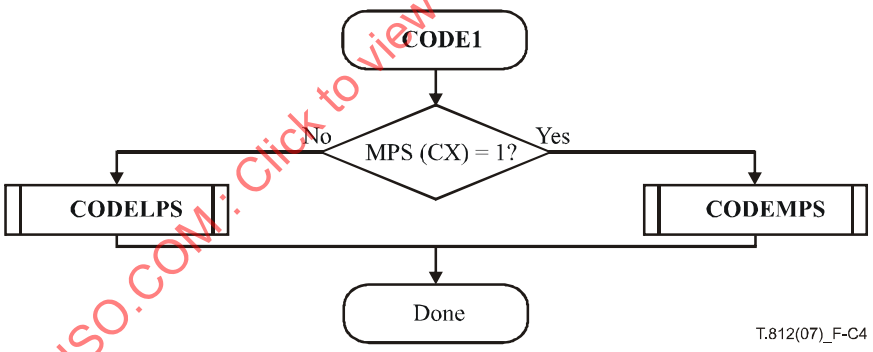

Figure C.4 – CODE1 procedure

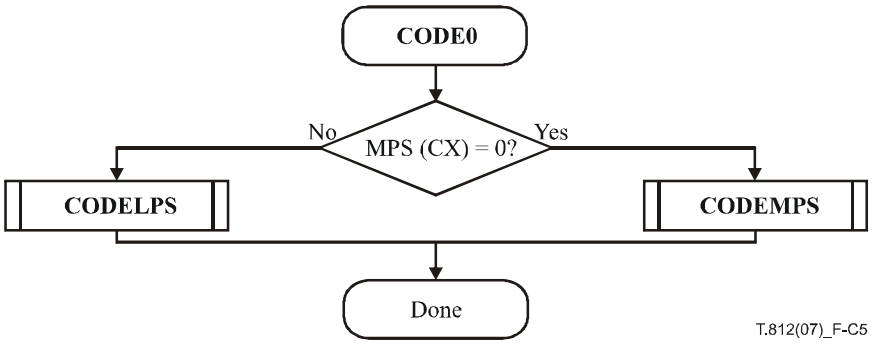

Figure C.5 – CODE0 procedure

C.2.4 Encoding an MPS or LPS (CODEMPS and CODELPS)

The CODELPS (Figure C.6) procedure usually consists of a scaling of the interval to Qe(I(CX)), the probability estimate of the LPS determined from the index I stored for context CX. The upper interval is first calculated so it can be compared to the lower interval to confirm that Qe has the smaller size. It is always followed by a renormalization (RENORME). In the event that the interval sizes are inverted, however, the conditional MPS/LPS exchange occurs and the upper interval is coded. In either case, the probability estimate is updated. If the SWITCH flag for the index I(CX) is set, then the MPS(CX) is inverted. A new index I is saved at CX as determined from the next LPS index (NLPS) column in Table C.2.

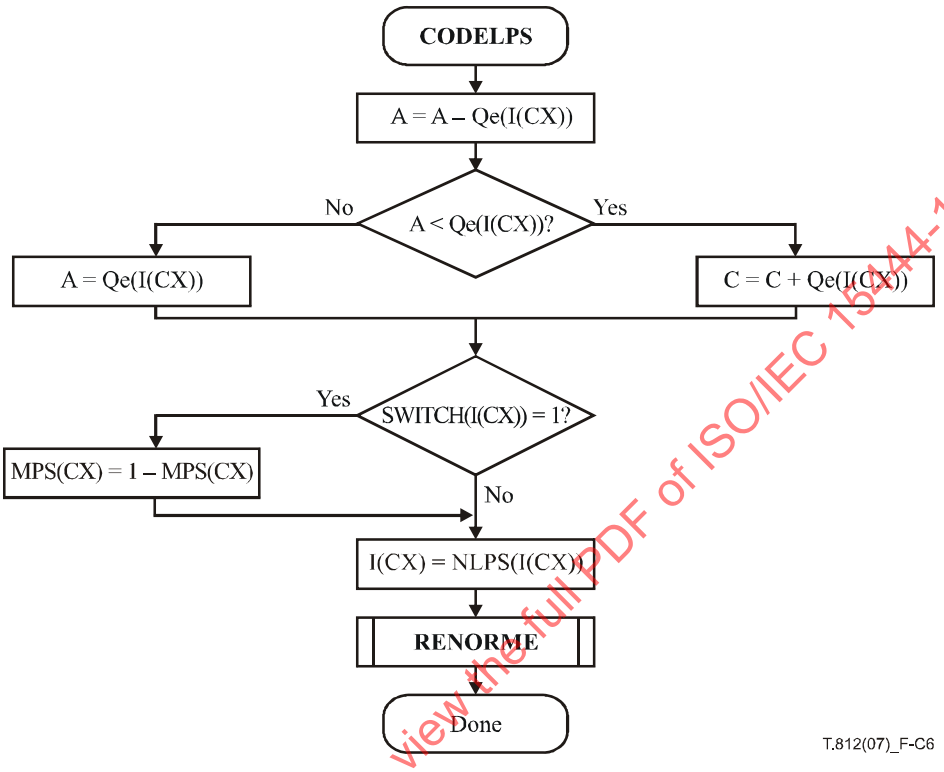

Figure C.6 – CODELPS procedure with conditional MPS/LPS exchange

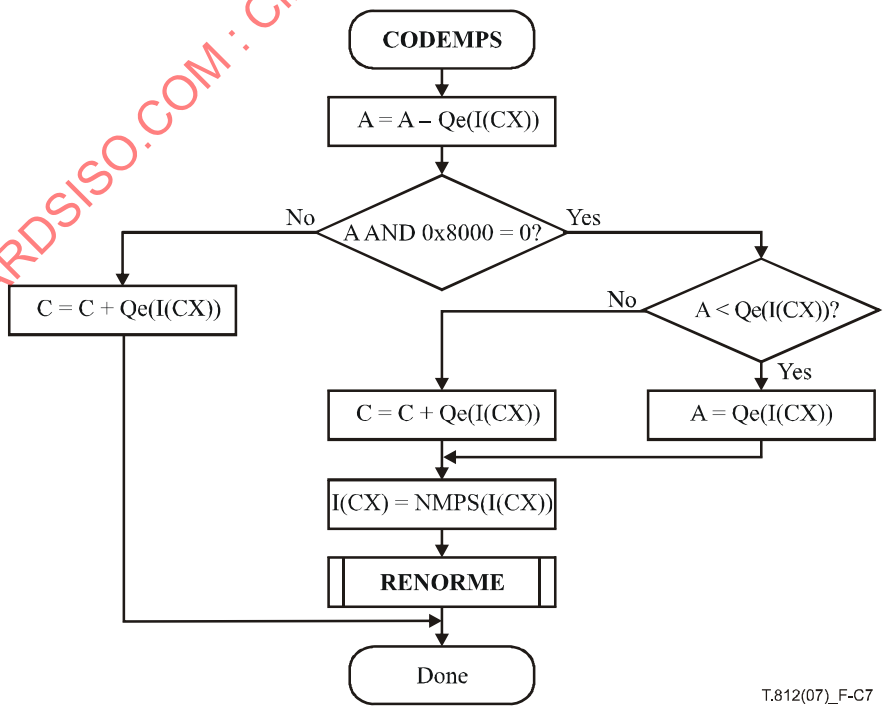

Figure C.7 – CODEMPS procedure with conditional MPS/LPS exchange

Table C.2 – Qe values and probability estimation

Index	Qe_Value			NMPS	NLPS	SWITCH
	(hexadecimal)	(binary)	(decimal)			
0	0x5601	0101 0110 0000 0001	0.503937	1	1	1
1	0x3401	0011 0100 0000 0001	0.304715	2	6	0
2	0x1801	0001 1000 0000 0001	0.140650	3	9	0
3	0x0AC1	0000 1010 1100 0001	0.063012	4	12	0
4	0x0521	0000 0101 0010 0001	0.030053	5	29	0
5	0x0221	0000 0010 0010 0001	0.012474	38	33	0
6	0x5601	0101 0110 0000 0001	0.503937	7	6	1
7	0x5401	0101 0100 0000 0001	0.492218	8	14	0
8	0x4801	0100 1000 0000 0001	0.421904	9	14	0
9	0x3801	0011 1000 0000 0001	0.328153	10	14	0
10	0x3001	0011 0000 0000 0001	0.281277	11	17	0
11	0x2401	0010 0100 0000 0001	0.210964	12	18	0
12	0x1C01	0001 1100 0000 0001	0.164088	13	20	0
13	0x1601	0001 0110 0000 0001	0.128931	29	21	0
14	0x5601	0101 0110 0000 0001	0.503937	15	14	1
15	0x5401	0101 0100 0000 0001	0.492218	16	14	0
16	0x5101	0101 0001 0000 0001	0.474640	17	15	0
17	0x4801	0100 1000 0000 0001	0.421904	18	16	0
18	0x3801	0011 1000 0000 0001	0.328153	19	17	0
19	0x3401	0011 0100 0000 0001	0.304715	20	18	0
20	0x3001	0011 0000 0000 0001	0.281277	21	19	0
21	0x2801	0010 1000 0000 0001	0.234401	22	19	0
22	0x2401	0010 0100 0000 0001	0.210964	23	20	0
23	0x2201	0010 0010 0000 0001	0.199245	24	21	0
24	0x1C01	0001 1100 0000 0001	0.164088	25	22	0
25	0x1801	0001 1000 0000 0001	0.140650	26	23	0
26	0x1601	0001 0110 0000 0001	0.128931	27	24	0
27	0x1401	0001 0100 0000 0001	0.117212	28	25	0
28	0x1201	0001 0010 0000 0001	0.105493	29	26	0
29	0x1101	0001 0001 0000 0001	0.099634	30	27	0
30	0x0AC1	0000 1010 1100 0001	0.063012	31	28	0
31	0x09C1	0000 1001 1100 0001	0.057153	32	29	0
32	0x08A1	0000 1000 1010 0001	0.050561	33	30	0
33	0x0521	0000 0101 0010 0001	0.030053	34	31	0
34	0x0441	0000 0100 0100 0001	0.024926	35	32	0
35	0x02Al	0000 0010 1010 0001	0.015404	36	33	0
36	0x0221	0000 0010 0010 0001	0.012474	37	34	0
37	0x0141	0000 0001 0100 0001	0.007347	38	35	0
38	0x0111	0000 0001 0001 0001	0.006249	39	36	0
39	0x0085	0000 0000 1000 0101	0.003044	40	37	0
40	0x0049	0000 0000 0100 1001	0.001671	41	38	0
41	0x0025	0000 0000 0010 0101	0.000847	42	39	0
42	0x0015	0000 0000 0001 0101	0.000481	43	40	0
43	0x0009	0000 0000 0000 1001	0.000206	44	41	0
44	0x0005	0000 0000 0000 0101	0.000114	45	42	0

Table C.2 – Qe values and probability estimation

Index	Qe_Value			NMPS	NLPS	SWITCH
	(hexadecimal)	(binary)	(decimal)			
45	0x0001	0000 0000 0000 0001	0.000023	45	43	0
46	0x5601	0101 0110 0000 0001	0.503937	46	46	0

C.2.5 Probability estimation

Table C.2 shows the Qe value associated with each Qe index. The Qe values are expressed as hexadecimal integers, as binary integers, and as decimal fractions. To convert the 15-bit integer representation of Qe to the decimal probability, the Qe values are divided by (4/3) * (0x8000).

The estimator can be defined as a finite-state machine – a table of Qe indexes and associated next states for each type of renormalization (i.e., new table positions) – as shown in Table C.2. The change in state occurs only when the arithmetic coder interval register is renormalized. This is always done after coding the LPS, and whenever the interval register is less than 0x8000 (0.75 in decimal notation) after coding the MPS.

After an LPS renormalization, NLPS gives the new index for the LPS probability estimate. If the switch is 1, the MPS symbol sense is reversed.

The index to the current estimate is part of the information stored for context CX. This index is used as the index to the table of values in NMPS, which gives the next index for an MPS renormalization. This index is saved in the context storage at CX. MPS(CX) does not change.

The procedure for estimating the probability on the LPS renormalization path is similar to that of an MPS renormalization, except that when SWITCH(I(CX)) is 1, the sense of MPS(CX) is inverted.

The final index state 46 can be used to establish a fixed 0.5 probability estimate.

C.2.6 Renormalization in the encoder (RENORME)

Renormalization is very similar in both encoder and decoder, except that in the encoder it generates compressed bits and in the decoder it consumes compressed bits.

The RENORME procedure for the encoder renormalization is illustrated in Figure C.8. Both the interval register A and the code register C are shifted, one bit at a time. The number of shifts is counted in the counter CT, and when CT is counted down to zero, a byte of compressed image data is removed from C by the procedure BYTEOUT. Renormalization continues until A is no longer less than 0x8000.

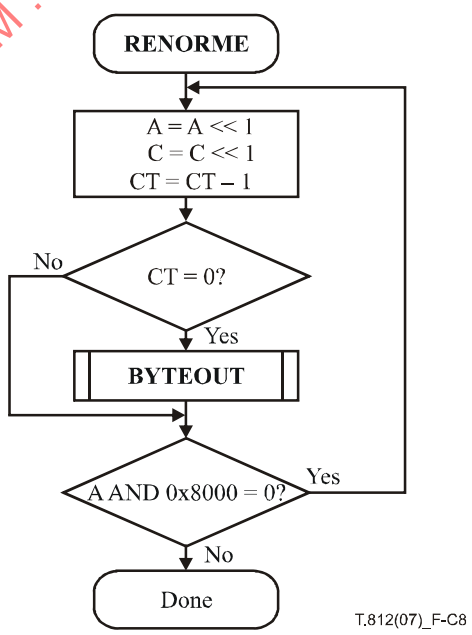

Figure C.8 – Encoder renormalization procedure

C.2.7 Compressed image data output (BYTEOUT)

The BYTEOUT routine called from RENORME is illustrated in Figure C.9. This routine contains the bit-stuffing procedures which are needed to limit carry propagation into the completed bytes of compressed image data. The conventions used make it impossible for a carry to propagate through more than the byte most recently written to the compressed image data buffer.

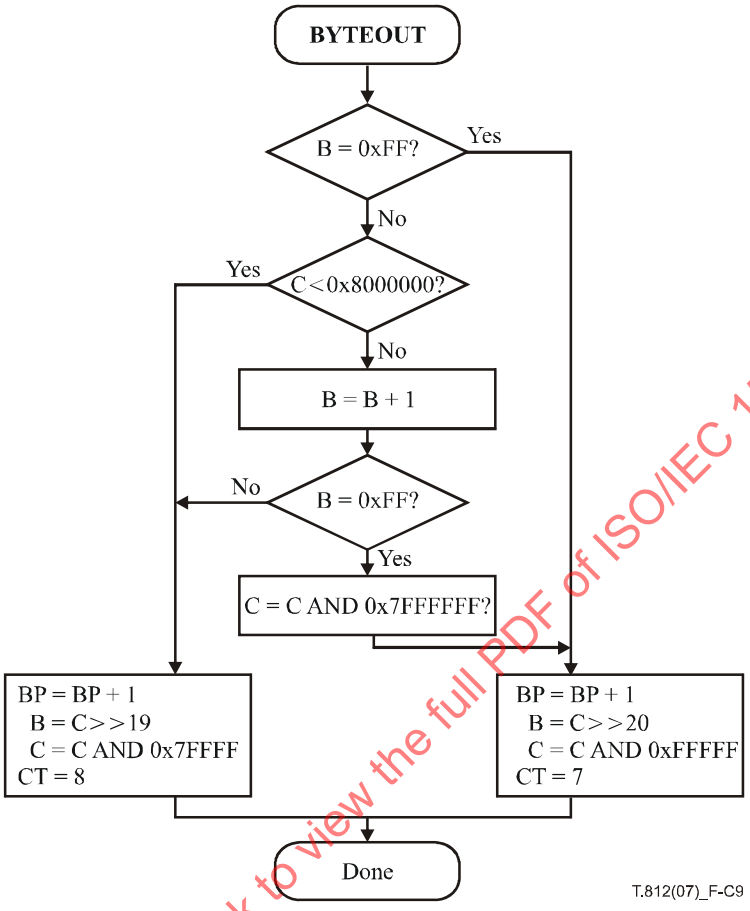

Figure C.9 – BYTEOUT procedure for encoder

The procedure in the block in the lower right section does bit stuffing after a 0xFF byte; the similar procedure on the left is for the case where bit stuffing is not needed.

B is the byte pointed to by the compressed image data buffer pointer BP. If B is not a 0xFF byte, the carry bit is checked. If the carry bit is set, it is added to B and B is again checked to see if a bit needs to be stuffed in the next byte. After the need for bit stuffing has been determined, the appropriate path is chosen, BP is incremented and the new value of B is removed from the code register "b" bits.

C.2.8 Initialization of the encoder (INITENC)

The INITENC procedure is used to start the arithmetic coder. After MPS and I are initialized, the basic steps are shown in Figure C.10.

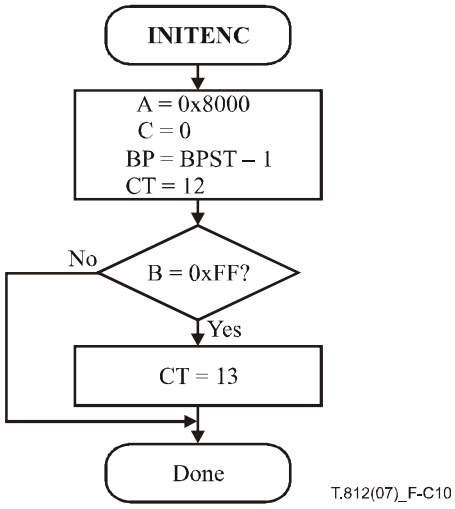

Figure C.10 – Initialization of the encoder

The interval register and code register are set to their initial values, and the bit counter is set. Setting CT = 12 reflects the fact that there are three spacer bits in the register which need to be filled before the field from which the bytes are removed is reached. BP always points to the byte preceding the position BPST where the first byte is placed. Therefore, if the preceding byte is a 0xFF byte, a spurious bit stuff will occur, but can be compensated for by increasing CT. The initial settings for MPS and I are shown in Table D.7.

C.2.9 Termination of coding (FLUSH)

The FLUSH procedure shown in Figure C.11 is used to terminate the encoding operations and generate the required terminating marker. The procedure guarantees that the 0xFF prefix to the marker code overlaps the final bits of the compressed image data. This guarantees that any marker code at the end of the compressed image data will be recognized and interpreted before decoding is complete.

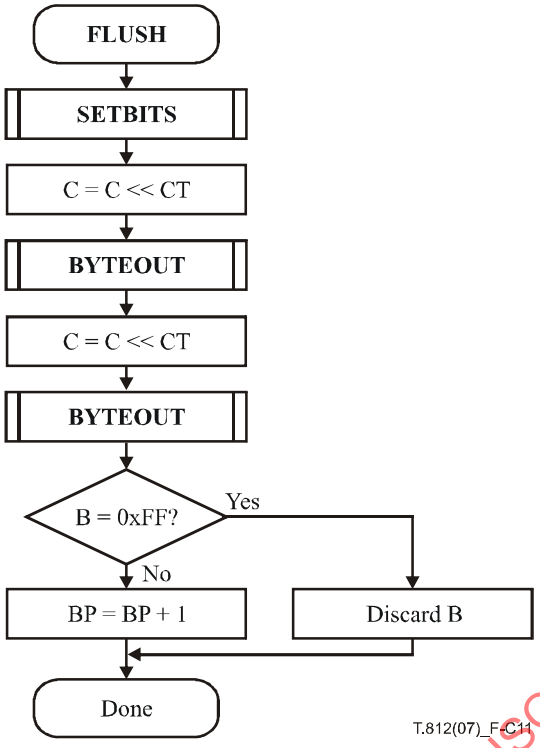

Figure C.11 – FLUSH procedure

The first part of the FLUSH procedure sets as many bits as possible in the C-register to 1 as shown in Figure C.12. The exclusive upper bound for the C-register is the sum of the C-register and the interval register. The low order 16 bits of C are forced to 1, and the result is compared to the upper bound. If C is too big, the leading 1 bit is removed, reducing C to a value which is within the interval.

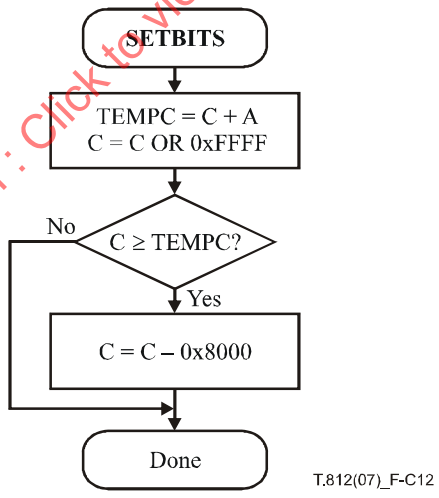

Figure C.12 – Setting the final bits in the C register

The byte in the C-register is then completed by shifting C, and two bytes are then removed. If the byte in buffer, B, is a 0xFF then it is discarded. Otherwise, buffer B is output to the bit stream.

NOTE – This is the only normative option for termination in ITU-T Rec. T.88 | ISO/IEC 14492. However, further reduction of the bit stream is allowed in this Recommendation | International Standard provided correct decoding is assured (see D.4.2).

Annex D

Coefficient bit modelling

(This annex forms an integral part of this Recommendation | International Standard)

In this annex, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

This annex defines the modelling and scanning of transform coefficient bits.

Code-blocks (see Annex B in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004) are encoded a bit-plane at a time starting from the most significant bit-plane with a non-zero element to the least significant bit-plane. For each bit-plane in a code-block, a special code-block scan pattern is used for each of three coding passes. Each coefficient bit in the bit-plane appears in only one of the three coding passes called significance propagation, magnitude refinement, and cleanup. For each pass contexts are created which are provided to the arithmetic coder, CX, along with the decision, D (see Annex C).

D.1 Code-block scan pattern within code-blocks

Each bit-plane of a code-block is scanned in a particular order. Starting at the top left, the first four coefficients of the first column are scanned, followed by the first four coefficients of the second column and so on, until the right side of the code-block is reached. The scan then returns to the left of the code-block and the second set of four coefficients in each column is scanned. The process is continued to the bottom of the code-block. If the code-block height is not divisible by 4, the last set of coefficients scanned in each column will contain fewer than 4 members. Figure D.1 shows an example of the code-block scan pattern for a code-block.

Code-block 16 wide by N high

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
64	...														
...															

T.812(07)_F-D1

Figure D.1 – Example scan pattern of a code-block bit-plane

D.2 Coefficient bits and significance

D.2.1 General case notations

The encoding procedures specified in this annex produce for each transform coefficient (u, v) of sub-band b the decoded bits to be encoded which will be used to reconstruct the transform coefficient value $q_b(u, v)$ at the decoder. The bits produced are: a sign bit $s_b(u, v)$ and a number $N_b(u, v)$ of magnitude MSBs to be encoded, ordered from most to least significant: $MSB_i(b, u, v)$ is the ith MSB of transform coefficient (u, v) of sub-band b ($i = 1, ..., N_b(u, v)$). As indicated in Equation (D-1), the sign bit $s_b(u, v)$ has a value of one for negative coefficients and of zero for positive coefficients. The number $N_b(u, v)$ of MSBs to be encoded includes the number of all zero most significant bit-planes signalled in the packet header (see B.10.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

D.2.2 Notation in the case with ROI

In the use of the RGN marker segment (indicating the presence of an ROI), modifications need to be made to the bits to be encoded, as well as the number $N_b(u, v)$ of magnitude MSBs to be encoded. These modifications are specified in clause H.1. In the absence of the RGN marker segment, no modification is required.

D.3 Encoding passes over the bit-planes

Each coefficient in a code-block has an associated binary state variable called its significance state. Significance states are initialized to 0 (coefficient is insignificant) and may become 1 (coefficient is significant) during the course of the encoding of the code-block. The "significance state" changes from insignificant to significant (see the subclause below) at the bit-plane where the most significant magnitude bit equal to 1 is found. The context vector for a given current coefficient is the binary vector consisting of the significance states of its 8 nearest-neighbour coefficients, as shown in Figure D.2. Any nearest neighbour lying outside the current coefficient's code-block is regarded as insignificant (i.e., it is treated as having a zero significance state) for the purpose creating a context vector for encoding the current coefficient.

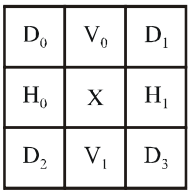

Figure D.2 – Neighbour states used to form the context

In general, a current coefficient can have 256 possible context vectors. These are clustered into a smaller number of contexts according to the rules specified below for context formation. Four different context formation rules are defined, one for each of the four coding passes: significance coding, sign coding, magnitude refinement coding, and cleanup coding. These coding operations are performed in three coding passes over each bit-plane: significance and sign coding in a significance propagation pass, magnitude refinement coding in a magnitude refinement pass, and cleanup and sign coding in a cleanup pass. For a given coding operation, the context label (or context) provided to the arithmetic coding engine is a label assigned to the current coefficient's context.

NOTE – Although (for the sake of concreteness) specific integers are used in the tables below for labelling contexts, the tokens used for context labels are implementation-dependent and their values are not mandated by this Recommendation | International Standard.

The first bit-plane within the current block with a non-zero element has a cleanup pass only. The remaining bit-planes are encoded in three coding passes. Each coefficient bit is encoded in exactly one of the three coding passes. Which pass a coefficient bit is encoded in depends on the conditions for that pass. In general, the significance propagation pass includes the coefficients that are predicted, or "most likely", to become significant and their sign bits, as appropriate. The magnitude refinement pass includes bits from already significant coefficients. The cleanup pass includes all the remaining coefficients.

D.3.1 Significance propagation encoding pass

The eight surrounding neighbour coefficients of a current coefficient (shown in Figure D.2 where X denotes the current coefficient) are used to create a context vector that maps into one of the 9 contexts shown in Table D.1. If a coefficient is significant then it is given a 1 value for the creation of the context, otherwise it is given a 0 value. The mapping to the contexts also depends on the sub-band.

Table D.1 – Contexts for the significance propagation and cleanup coding passes

LL and LH sub-bands (vertical high-pass)			HL sub-band (horizontal high-pass)			HH sub-band (diagonally high-pass)		Context label[a)]
$\sum H_i$	$\sum V_i$	$\sum D_i$	$\sum H_i$	$\sum V_i$	$\sum D_i$	$\sum (H_i + V_i)$	$\sum D_i$	
2	x[b)]	x	x	2	x	x	≥ 3	8
1	≥ 1	x	≥ 1	1	x	≥ 1	2	7
1	0	≥ 1	0	1	≥ 1	0	2	6
1	0	0	0	1	0	≥ 2	1	5
0	2	x	2	0	x	1	1	4
0	1	x	1	0	x	0	1	3
0	0	≥ 2	0	0	≥ 2	≥ 2	0	2
0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

a) Note that the context labels are indexed only for identification convenience in this Recommendation | International Standard. The actual identifiers used is a matter of implementation.

b) "x" indicates a "don't care" state.

The significance propagation pass only includes bits of coefficients that were insignificant (the significance state has yet to be set) and have a non-zero context. All other coefficients are skipped. The context is delivered to the arithmetic encoder (along with the coefficient bit to be encoded). If the value of this bit is 1, then the significance state is set to 1 and the immediate next bit to be encoded is the sign bit for the coefficient. Otherwise, the significance state remains 0. When the contexts of successive coefficients and coding passes are considered, the most current significance state for this coefficient is used.

D.3.2 Sign bit encoding

The context label for sign bit encoding is determined using another context vector from the neighbourhood. Computation of the context label can be viewed as a two-step process. The first step summarizes the contribution of the vertical and the horizontal neighbours. The second step reduces those contributions to one of 5 context labels.

For the first step, the two vertical neighbours (see Figure D.2) are considered together. Each neighbour may have one of three states: significant positive, significant negative, or insignificant. If the two vertical neighbours are both significant with the same sign, or if only one is significant, then the vertical contribution is 1 if the sign is positive or –1 if the sign is negative. If both vertical neighbours are insignificant, or both are significant with different signs, then the vertical contribution is 0. The horizontal contribution is created in the same way. Once again, if the neighbours fall outside the code-block they are considered to be insignificant. Table D.2 shows these contributions.

Table D.2 – Contributions of the vertical (and the horizontal) neighbours to the sign context

V0 (or H0)	V1 (or H1)	V (or H) contribution
significant, positive	significant, positive	1
significant, negative	significant, positive	0
insignificant	significant, positive	1
significant, positive	significant, negative	0
significant, negative	significant, negative	–1
insignificant	significant, negative	–1
significant, positive	insignificant	1
significant, negative	insignificant	–1
insignificant	insignificant	0

The second step reduces the nine permutations of the vertical and horizontal contributions into 5 context labels. Table D.3 shows these context labels. This context is provided to the arithmetic encoder with the decision. The sign bit is then logically exclusive ORed with the XORbit in Table D.3 to produce the decision, D (see Annex C). The following equation is used:

$$D = signbit \otimes XORbit \qquad \text{(D-1)}$$

where *signbit* is the sign bit of the current coefficient (a one bit indicates a negative coefficient, a zero bit a positive coefficient), *D* is the value provided to the arithmetic encoder with the context label, and the *XORbit* is found in Table D.3 for the current context label.

Table D.3 – Sign contexts from the vertical and horizontal contributions

Horizontal contribution	Vertical contribution	Context label	XORbit
1	1	13	0
1	0	12	0
1	–1	11	0
0	1	10	0
0	0	9	0
0	–1	10	1
–1	1	11	1
–1	0	12	1
–1	–1	13	1

D.3.3 Magnitude refinement pass

The magnitude refinement pass includes the bits from coefficients that are already significant (except those that have just become significant in the immediately preceding significance propagation pass).

The context used is determined by the summation of the significance state of the horizontal, vertical and diagonal neighbours. These are the states as currently known to the encoder, not the states used before the significance encoding pass. Further, it is dependent on whether this is the first refinement bit (the bit immediately after the significance and sign bits) or not. Table D.4 shows the three contexts for this pass.

Table D.4 – Contexts for the magnitude refinement coding passes

$\sum H_i + \sum V_i + \sum D_i$	First refinement for this coefficient	Context label
x[a)]	false	16
≥ 1	true	15
0	true	14

a) "x" indicates a "don't care" state.

The context is passed to the arithmetic coder along with the bit stream. The bit returned is the value of the current coefficient in the current bit-plane.

D.3.4 Cleanup pass

The remaining coefficients were previously insignificant and not handled by the significance propagation pass. The cleanup pass not only uses the neighbour context, like that of the significance propagation pass, from Table D.1, but also a run-length context.

During this pass, the neighbour contexts for the coefficients in this pass are recreated using Table D.1. The context label can now have any value because the coefficients that were found to be significant in the significance propagation pass are considered to be significant in the cleanup pass. Run-lengths are encoded with a unique single context. If the four contiguous coefficients in the column being scanned are all encoded in the cleanup pass and the context label for all is 0 (including context coefficients from previous magnitude, significance and cleanup passes), then the unique run-length context is given to the arithmetic encoder along with the decision, D (see Annex C). If all four contiguous coefficients in the column remain insignificant, then the decision is 0.

Otherwise, if at least one of the four contiguous coefficients in the column is significant, then the decision is 1. In this case, the next two bits, encoded with the UNIFORM context (index 46 in Table C.2), denote which coefficient from the top of the column down is the first to be found significant. The two bits encoded with the UNIFORM context are encoded MSB then LSB. That coefficient's sign bit is determined as described in D.3.2. The encoding of any remaining coefficients continues in the manner described in D.3.1.

If the four contiguous coefficients in a column are not all to be encoded in the cleanup pass or the context bin for any is non-zero, then the coefficient bits are encoded with the context in Table D.1 as in the significance propagation pass. The same contexts as the significance propagation are used here (the state is used as well as the model). Table D.5 shows the logic for the cleanup pass.

Table D.5 – Run-length encoder for cleanup passes

Four contiguous coefficients in a column remaining to be encoded and each currently has the 0 context	Symbols with run-length context	Four contiguous bits to be decoded are zero	Symbols decoded with UNIFORM[a] context	Number of coefficients to decode
true	0	true	none	none
true	1	false	MSB LSB	
		skip to first coefficient sign	00	3
		skip to second coefficient sign	01	2
		skip to third coefficient sign	10	1
		skip to fourth coefficient sign	11	0
false	none	x	none	rest of column

[a] See Annex C.

If there are fewer than four rows remaining in a code-block, then no run-length coding is used. Once again, the significance state of any coefficient is changed immediately after encoding the first 1 magnitude bit.

D.3.5 Example of coding passes and significance propagation

Table D.6 shows an example of the encoding order for the quantized coefficients of one 4-coefficient column in the scan. This example assumes all neighbours not included in the table are identically zero, and indicates in which pass each bit is encoded. The sign bit is encoded after the initial 1 bit and is indicated in the table by the + or – sign. The very first pass in a new block is always a cleanup pass because there can be no predicted significant, or refinement bits. After the first pass, the encoded 1 bit of the first coefficient causes the second coefficient to be encoded in the significance pass for the next bit-plane. The 1 bit encoded for the last coefficient in the second cleanup pass causes the third coefficient to be encoded in the next significance pass.

Table D.6 – Example of sub-bit-plane coding order and significance propagation

Coding passes	10	1	3	–7	Coefficient value
	+	+	+	–	Coefficient sign
	1	0	0	0	Coefficient magnitude
	0	0	0	1	(MSB to LSB)
	1	0	1	1	
	0	1	1	1	
Cleanup	1+	0	0	0	
Significance		0			
Refinement	0				
Cleanup			0	1–	
Significance		0	1+		
Refinement	1			1	
Cleanup					
Significance		1+			
Refinement	0		1	1	
Cleanup					

D.4 Initializing and terminating

When the contexts are initialized, or re-initialized, they are set to the values in Table D.7.

Table D.7 – Initial states for all contexts

Context	Initial index from Table C.1	MPS
UNIFORM	46	0
Run-length	3	0
All zero neighbours (context label 0 in Table D.1)	4	0
All other contexts	0	0

In normal operation (not selective arithmetic coding bypass), the arithmetic coder shall be terminated either at the end of every coding pass or only at the end of every code-block (see D.4.1). Table D.8 shows two examples of termination patterns for the coding passes in a code-block. The COD or COC marker signals which termination pattern is used (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

Table D.8 – Arithmetic coder termination patterns

#	Pass	Coding operation termination only on last pass	Coding operation termination on every pass
1	cleanup	Arithmetic Coder (AC)	AC, terminate
2	significance propagation	AC	AC, terminate
2	magnitude refinement	AC	AC, terminate
2	cleanup	AC	AC, terminate
...	...	...	...
final	significance propagation	AC	AC, terminate
final	magnitude refinement	AC	AC, terminate
final	cleanup	AC, terminate	AC, terminate

When multiple terminations of the arithmetic coder are present, the length of each terminated segment is signalled in the packet header as described in B.10.7 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

NOTE – Termination should never create a byte aligned value between 0xFF90 and 0xFFFF inclusive. These values are available as in-bit-stream marker values.

D.4.1 Expected codestream termination

The decoder anticipates that the given number of codestream bytes will decode a given number of coding passes before the arithmetic coder is terminated. During decoding, bytes are pulled successively from the codestream until all the bytes for those coding passes have been consumed. The number of bytes corresponding to the coding passes is specified in the packet header. Often at that point there are more symbols to be decoded. Therefore, the decoder shall extend the input bit stream to the arithmetic coder with 0xFF bytes, as necessary, until all symbols have been decoded.

It is sufficient to append no more than two 0xFF bytes. This will cause the arithmetic coder to have at least one pair of consecutive 0xFF bytes at its input which is interpreted as an end-of-stream marker (see C.3.4 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004). The bit stream does not actually contain a terminating marker. However, the byte length is explicitly signalled enabling the terminating marker to be synthesized for the arithmetic decoder.

NOTE – Two 0xFF bytes appended in this way is the simplest method. However, other equivalent extensions exist. This might be important since some arithmetic coder implementations might attach special meaning to the specific termination marker.

D.4.2 Arithmetic coder termination

The FLUSH procedure performs this task (see C.2.9). However, since the FLUSH procedure increases the length of the codestream, and frequent termination may be desirable, other techniques may be employed. Any technique that places all of the needed bytes in the codestream in such a way that the decoder need not backtrack to find the position at which the next segment of the codestream should begin is acceptable.

When the predictable termination flag is set (see COD and COC in A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), the following termination procedure shall be used. Using the notation of C.2, the following steps can be used:

1) Identify the number of bits in code register, C, which must be pushed out through the byte buffer. This is given by k = (11 – CT) + 1.
2) While (k > 0):
   – shift C left by CT and set CT = 0.

– execute the BYTEOUT procedure. This sets CT equal to the number of bits cleared out of the C register.

– subtract CT from k.

3) Execute the BYTEOUT procedure to push the contents of the byte buffer register out to the codestream. This step shall be skipped if the byte in the byte buffer has a 0xFF byte value.

The relevant truncation length in this case is simply the total number of bytes pushed out onto the codestream.

If the predictable termination flag is not set, the last byte output by the above procedure can generally be modified, within certain bounds, without affecting the symbols to be decoded. It will sometimes be possible to augment the last byte to the special value, 0xFF, which shall not be sent. It can be shown that this happens approximately 1/8 of the time.

D.4.3 Length computation

To include coding pass compressed image data into packets, the number of bytes to be included must be determined. If the coding pass compressed image data is terminated, the algorithm in the previous clause may be used. Otherwise, the encoder should calculate a suitable length such that corresponding bytes are sufficient for the decoder to reconstruct the coding passes.

D.5 Error resilience segmentation symbol

A segmentation symbol is a special symbol. Whether it is used or not is signalled in the COD or COC marker segments (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004). The symbol is coded with the UNIFORM context of the arithmetic coder at the end of each bit-plane. The correct decoding of this symbol confirms the correctness of the decoding of this bit-plane, which allows error detection. At the encoder, a segmentation symbol 1010 or 0xA should be encoded at the end of each bit-plane (at the end of a cleanup pass). If the segmentation symbol is not decoded correctly at the decoder, then bit errors occurred for this bit-plane.

NOTE – This can be used with or without the predictable termination.

D.6 Selective arithmetic coding bypass

This style of coding allows bypassing the arithmetic coder for the significance propagation pass and magnitude refinement coding passes starting in the fifth significant bit-plane of the code-block. The COD or COC marker signals whether or not this coding style is used (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

The first cleanup pass (which is the first bit-plane of a code-block with a non-zero element) and the next three sets of significance propagation, magnitude refinement, and cleanup coding passes are encoded with the arithmetic coder. The fourth cleanup pass shall include an arithmetic coder termination (see Table D.9).

Table D.9 – Selective arithmetic coding bypass

Bit-plane number	Pass type	Coding operations
1	cleanup	Arithmetic Coding (AC)
2	significance propagation	AC
2	magnitude refinement	AC
2	cleanup	AC
3	significance propagation	AC
3	magnitude refinement	AC
3	cleanup	AC
4	significance propagation	AC
4	magnitude refinement	AC
4	cleanup	AC, terminate
5	significance propagation	raw
5	magnitude refinement	raw, terminate
5	cleanup	AC, terminate
...	...	...
final	significance	raw
final	magnitude refinement	raw, terminate
final	cleanup	AC, terminate

Starting with the fourth significance propagation and magnitude refinement coding passes, the bits that would have been provided to the arithmetic coder are instead provided directly to the bit stream. After each magnitude refinement pass, the bit stream has been "terminated" by padding to the byte boundary.

When the predictable termination flag is set (see COD and COC in A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004) and all the bits from a magnitude refinement pass have been assembled, any remaining bits in the last byte are filled with an alternating sequence of 0s and 1s. This sequence should start with a 0 regardless of the number of bits to be padded.

When the termination on each coding pass flag is set (see COD and COC in A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), then the significance propagation passes are terminated in the same way as the magnitude refinement passes.

The cleanup coding passes continue to provide compressed image data directly to the arithmetic coder and are always terminated.

The sign bit is computed with Equation (D-2):

$$raw\_value = signbit \tag{D-2}$$

where *raw_value* = 1 is a negative sign bit and *raw_value* = 0 is a positive sign bit. Table D.9 shows the coding sequence.

The length of each terminated segment, plus the length of any remaining unterminated passes, is signalled in the packet header as described in B.10.7 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004. If termination on each coding pass is selected (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), then every pass is terminated (including both raw passes).

NOTE 1 – Using the selective bypass mode when encoding an image with an ROI may significantly decrease the compression efficiency.

Bits are packed into bytes from the most significant bit to the least significant bit. Once a complete byte is assembled, it is emitted to the bit stream. If the value of the byte is a 0xFF a single zero bit is stuffed into the most significant bit of the next byte. Once all bits of the coding pass have been assembled, the last byte is packed to the byte boundary and emitted. The last byte shall not be a 0xFF value.

At the decoder, if a 0xFF value is encountered in the bit stream, then the first bit of the next byte is discarded. The sequence of bits used in the selective arithmetic coding bypass have been stuffed into bytes using a bit-stuffing routine.

NOTE 2 – Since the decoder appends 0xFF values, as necessary, to the bit stream representing the coding pass (see D.4.1), truncation of the bit stream may be possible. When the predictable termination flag is set (see COD and COC in A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), such truncation is not permitted. The last byte cannot be an 0xFF, since the bit-stuffing routine appends a new byte following the FF, having most significant bit value of 0 and unused bits filled with the alternating sequence of 0 and 1 value bits.

D.7 Vertically causal context formation

This style of coding constrains the context formation to the current and past code-block scans (four rows of vertically scanned coefficients). That is, any coefficient from the next code-block scan are considered to be insignificant. The COD or COC marker signals whether or not this style of coding is used (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

To illustrate, the bit labelled 14 in Figure D.1 is encoded as usual using the neighbour states as specified in Figure D.2, independent of whether or not contexts are vertically causal. However when vertically causal context formation is used, the bit labelled 15 is encoded assuming $D_2 = V_1 = D_3 = 0$ in Figure D.2.

D.8 Flow diagram of the code-block coding

The steps for modelling each bit-plane of each code-block can be viewed graphically in Figure D.3. The decisions made are in Table D.10 and the bits and context sent to the coder are in Table D.11. These show the context without the selective arithmetic coding bypass or the vertically causal model.

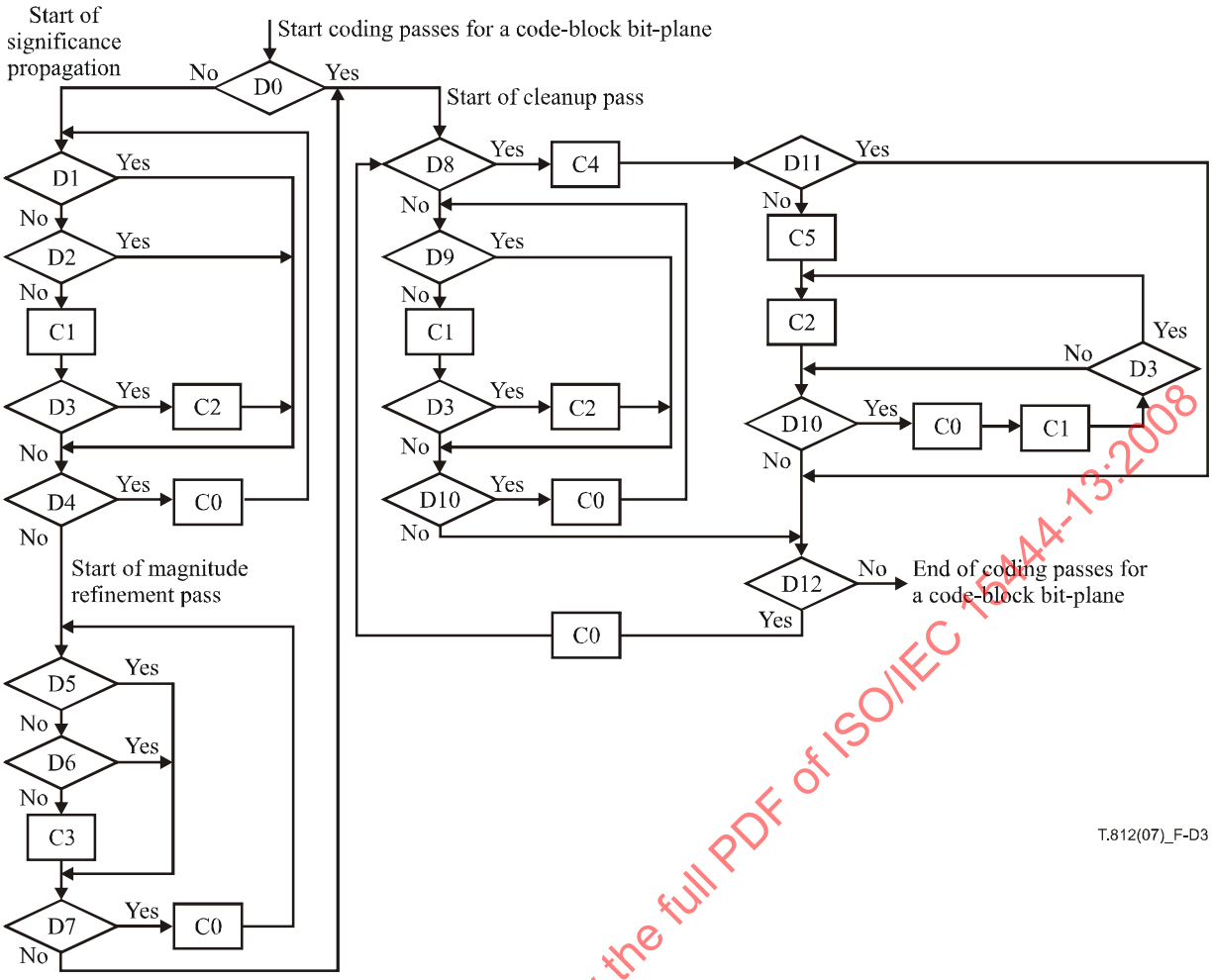

Figure D.3 – Flow chart for all coding passes on a code-block bit-plane

Table D.10 – Decisions in the context model flow chart

Decision	Question	Description
D0	Is this the first significant bit-plane for the code-block?	See D.3
D1	Is the current coefficient significant?	See D.3.1
D2	Is the context bin zero? (see Table D.1)	See D.3.1
D3	Did the current coefficient just become significant?	See D.3.1
D4	Are there more coefficients in the significance propagation?	
D5	Is the coefficient insignificant?	See D.3.3
D6	Was the coefficient coded in the last significance propagation?	See D.3.3
D7	Are there more coefficients in the magnitude refinement pass?	
D8	Are four contiguous undecoded coefficients in a column each with a 0 context?	See D.3.4
D9	Is the coefficient significant or has the bit already been coded during the Significance Propagation coding pass?	See D.3.4
D10	Are there more coefficients remaining of the four column coefficients?	
D11	Are the four contiguous bits all zero?	See D.3.4
D12	Are there more coefficients in the cleanup pass?	

Table D.11 – Decoding in the context model flow chart

Code	Decoded symbol	Context	Brief explanation	Description
C0	–	–	Go to the next coefficient or column	
C1	Newly significant?	Table D.1, 9 context labels	Decode significance bit of current coefficient (significance propagation or cleanup)	See D.3.1
C2	Sign bit	Table D.3, 5 context labels	Decode sign bit of current coefficient	See D.3.2
C3	Current magnitude bit	Table D.4, 3 context labels	Decode magnitude refinement pass bit of current coefficient	See D.3.3
C4	0 1	Run-length context label	Decode run-length of four zeros Decode run-length not of four zeros	See D.3.4
C5	00 01 10 11	UNIFORM	First coefficient is first with non-zero bit Second coefficient is first with non-zero bit Third coefficient is first with non-zero bit Fourth coefficient is first with non-zero bit	See D.3.4 and Table C.2

Annex E

Quantization

(This annex forms an integral part of this Recommendation | International Standard)

In this annex and all of its subclauses, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

This annex specifies the quantization of transform coefficients for encoding. Quantization is the process by which the transform coefficients are reduced in precision.

E.1 Inverse quantization procedure (Informative)

For each transform coefficient (u, v) of a given sub-band b, the decoded transform coefficient value $\overline{q_b}(u,v)$ is given by the following equation:

$$\overline{q_b}(u,v) = \left(1-2s_b(u,v)\right)\cdot\left(\sum_{i=1}^{N_b(u,v)} MSB_i(b,u,v)\cdot 2^{M_b-i}\right) \tag{E-1}$$

NOTE 1 – $q_b(u, v)$ is quantized value of $a_b(u, v)$ according to Equation (E-6) and $\overline{q_b}(u,v)$ is inverse quantization value of $q_b(u, v)$.

where $s_b(u, v)$, $N_b(u, v)$ and $MSB_i(b, u, v)$ are given in D.2, and where M_b is retrieved using Equation (E-2), where the number of guard bits G and the exponent ε_b are specified in the QCD or QCC marker segments (see A.6.4 and A.6.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

$$M_b = G + \varepsilon_b - 1 \tag{E-2}$$

Each decoded transform coefficient $q_b(u, v)$ of sub-band b is used to generate a reconstructed transform coefficient $Rq_b(u, v)$, as will be described in E.1.1.

NOTE 2 – Encoding only $N_b(u, v)$ (see D.2.1) bit-planes is equivalent to encoding data which has been encoded using a scalar quantizer with step size $2^{M_b - N_b(u,v)}\cdot\Delta_b$ for all the coefficients of this code-block. Due to the nature of the three coding passes (see D.3), $N_b(u, v)$ may be different for different coefficients within the same code-block.

E.1.1 Determination of the quantization step size in Irreversible transformation

For the irreversible transformation, the quantization step size Δ_b for a given sub-band b is calculated from the dynamic range R_b of sub-band b, the exponent ε_b and mantissa μ_b as given in Equation (E-3).

$$\Delta_b = 2^{R_b-\varepsilon_b}\left(1+\frac{\mu_b}{2^{11}}\right) \tag{E-3}$$

NOTE 1 – The denominator, 2^{11}, in Equation (E-3) is due to the allocation of 11 bits in the codestream for μ_b, as given in Table A.30 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004.

In Equation (E-3), the exponent ε_b and the mantissa μ_b are specified in the QCD or QCC marker segments (see A.6.4 and A.6.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), and the nominal dynamic range R_b (as given by Equation (E-4)) is the sum of R_I (the number of bits used to represent the original tile-component samples which can be extracted from the SIZ marker – see Table A.11 in A.5.1 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004) and the base 2 exponent of the sub-band gain ($gain_b$) of the current sub-band b, which varies with the type of sub-band b (levLL, levLH or levHL, levHH – see F.3.1) and can be found in Table E.1.

Table E.1 – Sub-band gains

sub-band b	$gain_b$	$\log_2(gain_b)$
*lev*LL	1	0
*lev*LH	2	1
*lev*HL	2	1
*lev*HH	4	2

$$R_b = R_I + log_2(gain_b) \tag{E-4}$$

The exponent/mantissa pairs (ε_b,μ_b) are either signalled in the codestream for every sub-band (expounded quantization) or else signalled only for the N_LLL sub-band and derived for all other sub-bands (derived quantization) (see Table A.30 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004). In the case of derived quantization, all exponent/mantissa pairs (ε_b,μ_b) are derived from the single exponent/mantissa pair (ε_o,μ_o) corresponding to the N_LLL sub-band, according to Equation (E-5):

$$(\varepsilon_b,\mu_b) = (\varepsilon_o - N_L + n_b,\mu_o) \tag{E-5}$$

where n_b denotes the number of decomposition levels from the original tile-component to the sub-band b.

NOTE 2 – For a given sub-band b, a quantized transform coefficient may exceed the dynamic range R_b.

E.1.2 Determination of the quantization step size in reversible transformation

For the reversible transformation, the quantization step size Δ_b is equal to one (no quantization is performed).

E.2 Scalar coefficient quantization

For irreversible compression, after the irreversible forward discrete wavelet transformation (see Annex F), each of the transform coefficients $a_b(u, v)$ of the sub-band is quantized to the value $q_b(u, v)$ according to Equation (E-6).

$$q_b(u,v) = sign(a_b(u,v)) \cdot \left\lfloor \frac{|a_b(u,v)|}{\Delta_b} \right\rfloor \tag{E-6}$$

where Δ_b is the quantization step size. The exponent ε_b and mantissa corresponding to Δ_b can be derived from Equation (E-5), and must be recorded in the codestream in the QCD or QCC markers (see A.6.4 and A.6.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

For reversible compression, the quantization step size is required to be 1. In this case, a parameter ε_b has to be recorded in the codestream in the QCD or QCC markers (see A.6.4 and A.6.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), and is calculated as:

$$\varepsilon_b = R_I + log_2(gain_b) + \zeta_c \tag{E-7}$$

where R_I and $gain_b$ are as described in E.1.1, and where ζ_c is zero if the RCT is not used and ζ_c is the number of additional bits added by the RCT if the RCT is used, as described in G.2.

For both reversible and irreversible compression, in order to prevent possible overflow or excursion beyond the nominal range of the integer representation of $|q_b(u,v)|$ arising, for example during floating point calculations, the number M_b of bits for the integer representation of $q_b(u, v)$ used at the encoder side is defined by Equation (E-2). The number G of guard bits has to be specified in the QCD or QCC marker (see A.6.4 and A.6.5 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004). Typical values for the number of guard bits are $G = 1$ or $G = 2$.

Annex F

Discrete wavelet transformation of tile-components

(This annex forms an integral part of this Recommendation | International Standard)

In this annex, the flow charts and tables are normative only in the sense that they are defining an output that alternative implementations shall duplicate.

This annex describes the forward discrete wavelet transformation applied to one tile-component.

F.1 Tile-component parameters

Consider the tile-component defined by the coordinates, tcx_0, tcx_1, tcy_0 and tcy_1 given in Equation (B-12) in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004, are reproduced herein:

$$tcx_0 = \left\lceil \frac{tx_0}{XRsiz^i} \right\rceil \quad tcx_1 = \left\lceil \frac{tx_1}{XRsiz^i} \right\rceil \quad tcy_0 = \left\lceil \frac{ty_0}{YRsiz^i} \right\rceil \quad tcy_1 = \left\lceil \frac{ty_1}{YRsiz^i} \right\rceil$$

Then the coordinates (x, y) of the tile-component (with sample values $I(x, y)$) lie in the range defined by:

$$tcx_0 \le x < tcx_1 \quad \text{and} \quad tcy_0 \le y < tcy_1 \tag{F-1}$$

F.2 Discrete wavelet transformations

F.2.1 Low-pass and high-pass filtering

To perform the forward discrete wavelet transformation (FDWT), this Recommendation | International Standard uses a one-dimensional sub-band decomposition of a one-dimensional array of samples into low-pass coefficients, representing a downsampled low-resolution version of the original array, and high-pass coefficients, representing a downsampled residual version of the original array, needed to perfectly reconstruct the original array from the low-pass array.

F.2.2 Decomposition levels

Each tile-component is transformed into a set of two-dimensional sub-band signals (called sub-bands), each representing the activity of the signal in various frequency bands, at various spatial resolutions. N_L denotes the number of decomposition levels.

F.2.3 Discrete wavelet filters

This Recommendation | International Standard specifies one reversible transformation and one irreversible transformation. Given that tile-component samples are integer-valued, a reversible transformation requires the specification of a rounding procedure for intermediate non-integer-valued transform coefficients.

F.3 Forward transformation

F.3.1 The FDWT procedure

The forward discrete wavelet transformation (FDWT) transforms DC-level shifted tile-component samples $I(x, y)$ into a set of sub-bands with coefficients $a_b(u_b, v_b)$ (FDWT procedure). The FDWT procedure (see Figure F.1) also takes as input the number of decomposition levels N_L signalled in the COD or COC markers (see A.6.1 and A.6.2 in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

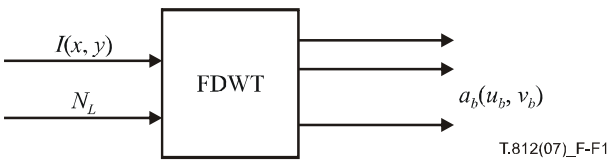

Figure F.1 – Inputs and outputs of the FDWT procedure

As illustrated in Figure F.2, all the sub-bands in the case where $N_L = 2$ can be represented in the following way:

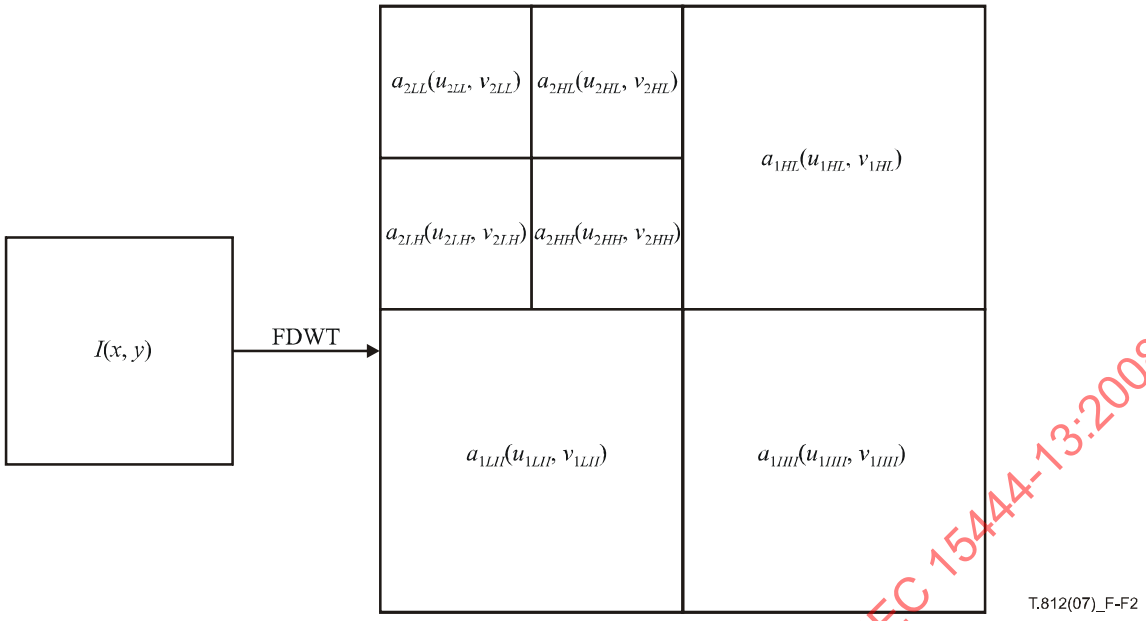

Figure F.2 – The FDWT (N_L = 2)

The FDWT procedure starts with the initialization of the variable *lev* (the current decomposition level) to zero, and setting the sub-band $a_{0LL}(u_{0LL}, v_{0LL})$ to the input array $I(u, v)$. The 2D_SD procedure is performed at every level *lev*, where the level *lev* increases by one at each iteration, and until N_L iterations are performed. The 2D_SD procedure is iterated over the *lev*LL sub-band produced at each iteration.

As defined in Annex B (see Equation (B-15) in Annex B in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004), the coordinates of the sub-band $a_{levLL}(u, v)$ lie in the range defined by:

$$tbx_0 \le u < tbx_1 \quad and \quad tby_0 \le v < tby_1 \tag{F-2}$$

Figure F.3 describes the FDWT procedure.

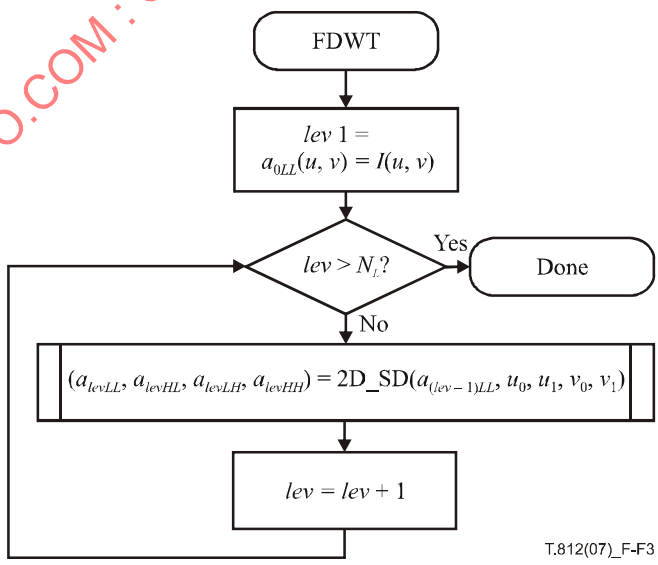

Figure F.3 – The FDWT procedure

F.3.2 The 2D_SD procedure

The 2D_SD procedure performs a decomposition of a two-dimensional array of coefficients or samples $a_{(lev-1)LL}(u, v)$ into four groups of sub-band coefficients $a_{levLL}(u, v)$, $a_{levHL}(u, v)$, $a_{levLH}(u, v)$, and $a_{levHH}(u, v)$.

The total number of coefficients of the lev_{LL} sub-band is equal to the sum of the total number of coefficients of the four sub-bands resulting from the 2D_SD procedure.

Figure F.4 describes the input and output parameters of the 2D_SD procedure.

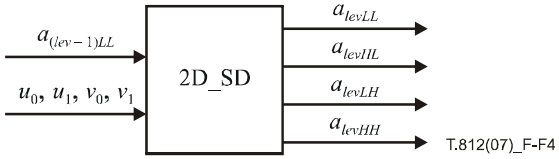

Figure F.4 – Inputs and outputs of the 2D_SD procedure

Figure F.5 illustrates the sub-band decomposition performed by the 2D_SD procedure.

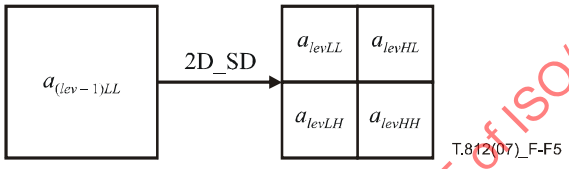

Figure F.5 – One-level decomposition into four sub-bands (2D_SD procedure)

The 2D_SD procedure first applies the VER_SD procedure to all columns of $a(u, v)$. It then applies the HOR_SD procedure to all rows of $a(u, v)$. The coefficients thus obtained from $a(u, v)$ are deinterleaved into the four sub-bands using the 2D_DEINTERLEAVE procedure.

Figure F.6 describes the 2D_SD procedure.

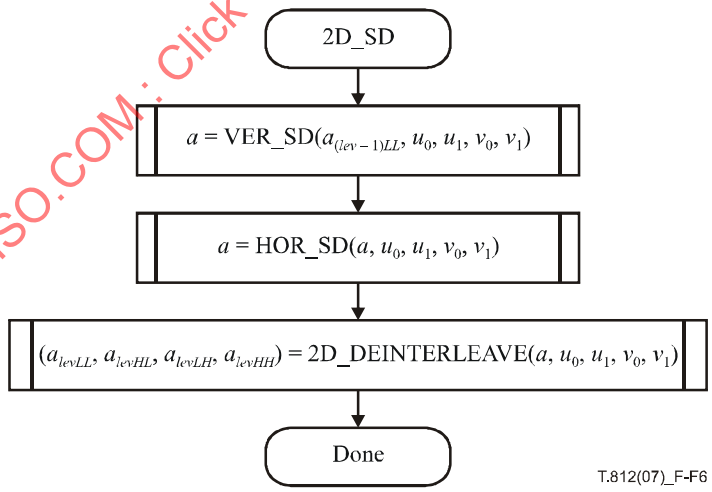

Figure F.6 – The 2D_SD procedure

F.3.3 The VER_SD procedure

The VER_SD procedure performs a vertical sub-band decomposition of a two-dimensional array of coefficients. It takes as input the two-dimensional array $a_{(lev-1)LL}(u, v)$, the horizontal and vertical extent of its coefficients as indicated by $u_0 \le u < u_1$ and $v_0 \le v < v_1$ (see Figure F.7) and produces as output a vertically filtered version $a(u, v)$ of the input array, column by column. The values of u_0, u_1, v_0, v_1 used by the VER_SD procedure are those of tbx_0, tbx_1, tby_0, tby_1 corresponding to sub-band $b = (lev - 1)$LL (see definition in Equation (B-15) in Annex B in ITU-T Rec. T.800 | ISO/IEC 15444-1:2004).

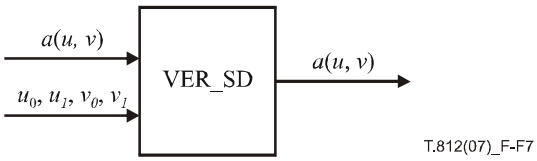

Figure F.7 – Inputs and outputs of the VER_SD procedure

As illustrated in Figure F.8, the VER_SD procedure applies the one-dimensional sub-band decomposition (1D_SD procedure) to each column of the input array $a(u, v)$, and stores the result back into each column.

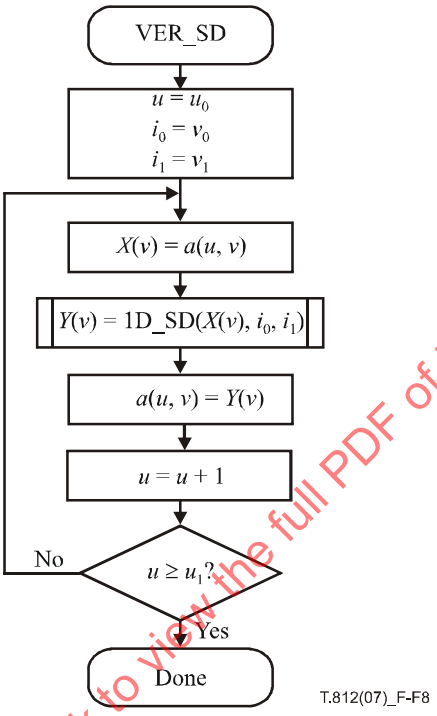

Figure F.8 – The VER_SD procedure

F.3.4 The HOR_SD procedure

The HOR_SD procedure performs a horizontal sub-band decomposition of a two-dimensional array of coefficients. It takes as input a two-dimensional array $a(u, v)$, the horizontal and vertical extent of its coefficients as indicated by $u_0 \le u < u_1$ and $v_0 \le v < v_1$ (see Figure F.9) and produces as output a horizontally filtered version of the input array, row by row.

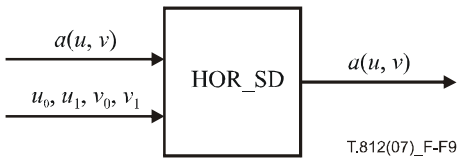

Figure F.9 – Inputs and outputs of the HOR_SD procedure

As illustrated in Figure F.10, the HOR_SD procedure applies the one-dimensional sub-band decomposition (1D_SD procedure) to each row of the input array $a(u, v)$ and stores the result back in each row.

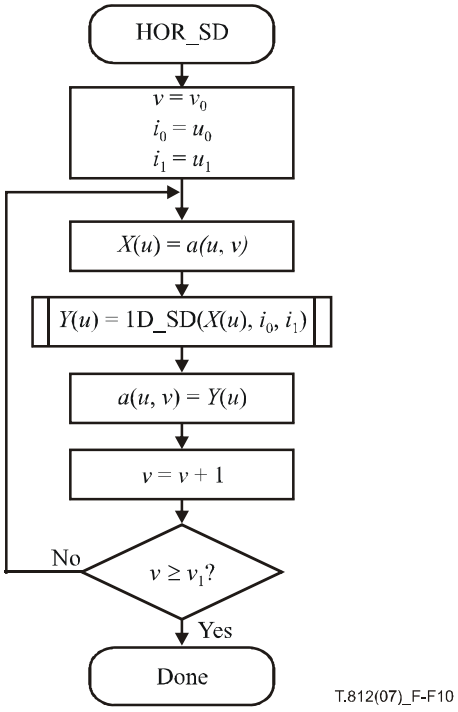

Figure F.10 – The HOR_SD procedure

F.3.5 The 2D_DEINTERLEAVE procedure

As illustrated in Figure F.11, the 2D_DEINTERLEAVE procedure deinterleaves the coefficients of $a(u, v)$ into four sub-bands. The arrangement is dependent on the coordinates (u_0, v_0) of the first coefficient of $a(u, v)$.

The way these sub-bands are formed from the output $a(u, v)$ of the HOR_SD procedure is described by the 2D_DEINTERLEAVE procedure illustrated in Figure F.12.

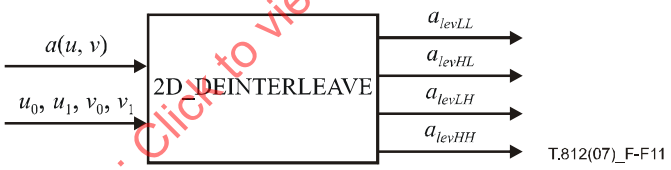

Figure F.11 – Parameters of 2D_DEINTERLEAVE procedure

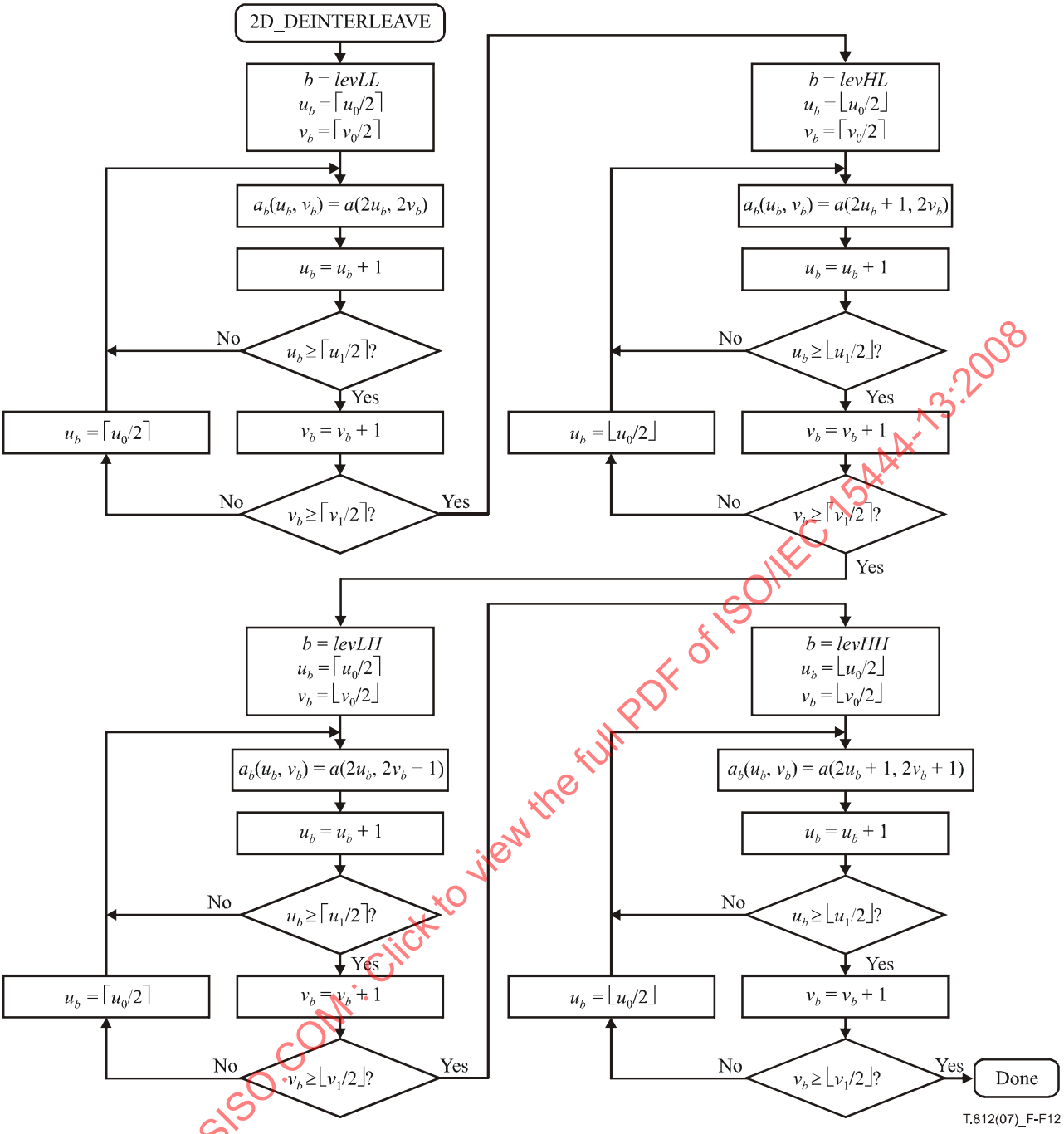

Figure F.12 – The 2D_DEINTERLEAVE procedure

F.3.6 The 1D_SD procedure

As illustrated in Figure F.13, the 1D_SD procedure takes as input a one-dimensional array *X(i)*, the extent of its coefficients as indicated by $i_0 \leq i < i_1$. It produces as output an array *Y(i)*, with the same indices (i_0, i_1).

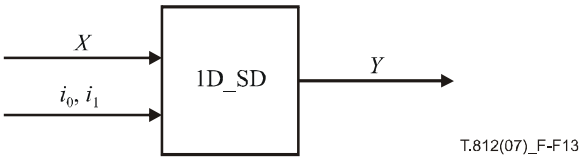

Figure F.13 – Parameters of the 1D_SD procedure