

INTERNATIONAL
STANDARD

ISO/IEC
10918-1

First edition
1994-02-15

**Information technology — Digital
compression and coding of continuous-tone
still images: Requirements and guidelines**

*Technologies de l'information — Compression numérique et codage des
images fixes de nature photographique: Prescriptions et lignes directrices*



Reference number
ISO/IEC 10918-1:1994(E)

AVA Documents
Document Sharing Hub

Contents

Page

1	Scope.....	1
2	Normative references.....	1
3	Definitions, abbreviations and symbols.....	1
4	General.....	12
5	Interchange format requirements.....	23
6	Encoder requirements	23
7	Decoder requirements	23

Annexes

A	Mathematical definitions	24
B	Compressed data formats	31
C	Huffman table specification	50
D	Arithmetic coding	54
E	Encoder and decoder control procedures.....	77
F	Sequential DCT-based mode operation	87
G	Progressive DCT-based mode of operation	119
H	Lossless mode of operation.....	132
J	Hierarchical mode of operation.....	137
K	Examples and guidelines.....	143
L	Patents	179
M	Bibliography.....	181

© ISO/IEC 1994

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

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. 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.

International Standard ISO/IEC 10918-1 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 the CCITT. The identical text is published as CCITT Recommendation T.81.

ISO/IEC 10918 consists of the following parts, under the general title *Information technology — Digital compression and coding of continuous-tone still images*:

- *Part 1: Requirements and guidelines*
- *Part 2: Compliance testing*

Annexes A, B, C, D, E, F, G, H and J form an integral part of this part of ISO/IEC 10918. Annexes K, L and M are for information only.

Patents

During the preparation of this part of ISO/IEC 10918, information was gathered on patents upon which application of the standard might depend. Relevant patents were identified as belonging to the patent holders listed in annex L. However, ISO/IEC cannot give authoritative or comprehensive information about evidence, validity or scope of patent and like rights. The patent holders have stated that licences will be granted under reasonable terms. Communications on this subject should be addressed to the patent holders (see annex L).

Introduction

This part of ISO/IEC 10918 was prepared by CCITT Study Group VIII and the Joint Photographic Experts Group (JPEG) of ISO/IEC JTC 1/SC 29/WG 10. This Experts Group was formed in 1986 to establish a standard for the sequential progressive encoding of continuous-tone grayscale and colour images.

This part of ISO/IEC 10918 sets out requirements and implementation guidelines for continuous-tone still image encoding and decoding processes, and for the coded representation of compressed image data for interchange between applications. These processes and representations are intended to be generic, that is, to be applicable to a broad range of applications for colour and grayscale still images within communications and computer systems. ISO/IEC 10918-2 sets out tests for determining whether implementations comply with the requirements for the various encoding and decoding processes specified in this part of ISO/IEC 10918.

The requirements which these processes must satisfy to be useful for specific image communications applications such as facsimile, Videotex and audiographic conferencing are defined in CCITT Recommendation T.80. The intent is that the generic processes of Recommendation T.80 will be incorporated into the various CCITT Recommendations for terminal equipment for these applications.

In addition to the applications addressed by the CCITT and ISO/IEC, the JPEG committee has developed a compression standard to meet the needs of other applications as well, including desktop publishing, graphic arts, medical imaging and scientific imaging.

This Specification aims to follow the guidelines of CCITT and ISO/IEC JTC 1 on *Rules for presentation of CCITT \ ISO/IEC common text*.

INTERNATIONAL STANDARD

CCITT RECOMMENDATION

INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF CONTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES

1 Scope

This CCITT Recommendation | International Standard is applicable to continuous-tone – grayscale or colour – digital still image data. It is applicable to a wide range of applications which require use of compressed images. It is not applicable to bi-level image data.

This Specification

- specifies processes for converting source image data to compressed image data;
- specifies processes for converting compressed image data to reconstructed image data;
- gives guidance on how to implement these processes in practice;
- specifies coded representations for compressed image data.

NOTE – This Specification does not specify a complete coded image representation. Such representations may include certain parameters, such as aspect ratio, component sample registration, and colour space designation, which are application-dependent.

2 Normative references

The following CCITT Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this CCITT 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 CCITT 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 CCITT Secretariat maintains a list of currently valid CCITT Recommendations.

- *CCITT Recommendation T.80 (1992), Common components for image compression and communication – Basic principles.*

3 Definitions, abbreviations and symbols

3.1 Definitions and abbreviations

For the purposes of this Specification, the following definitions apply.

3.1.1 abbreviated format: A representation of compressed image data which is missing some or all of the table specifications required for decoding, or a representation of table-specification data without frame headers, scan headers, and entropy-coded segments.

3.1.2 AC coefficient: Any DCT coefficient for which the frequency is not zero in at least one dimension.

3.1.3 (adaptive) (binary) arithmetic decoding: An entropy decoding procedure which recovers the sequence of symbols from the sequence of bits produced by the arithmetic encoder.

3.1.4 (adaptive) (binary) arithmetic encoding: An entropy encoding procedure which codes by means of a recursive subdivision of the probability of the sequence of symbols coded up to that point.

3.1.5 application environment: The standards for data representation, communication, or storage which have been established for a particular application.

- 3.1.6 arithmetic decoder:** An embodiment of arithmetic decoding procedure.
- 3.1.7 arithmetic encoder:** An embodiment of arithmetic encoding procedure.
- 3.1.8 baseline (sequential):** A particular sequential DCT-based encoding and decoding process specified in this Specification, and which is required for all DCT-based decoding processes.
- 3.1.9 binary decision:** Choice between two alternatives.
- 3.1.10 bit stream:** Partially encoded or decoded sequence of bits comprising an entropy-coded segment.
- 3.1.11 block:** An 8×8 array of samples or an 8×8 array of DCT coefficient values of one component.
- 3.1.12 block-row:** A sequence of eight contiguous component lines which are partitioned into 8×8 blocks.
- 3.1.13 byte:** A group of 8 bits.
- 3.1.14 byte stuffing:** A procedure in which either the Huffman coder or the arithmetic coder inserts a zero byte into the entropy-coded segment following the generation of an encoded hexadecimal X'FF' byte.
- 3.1.15 carry bit:** A bit in the arithmetic encoder code register which is set if a carry-over in the code register overflows the eight bits reserved for the output byte.
- 3.1.16 ceiling function:** The mathematical procedure in which the greatest integer value of a real number is obtained by selecting the smallest integer value which is greater than or equal to the real number.
- 3.1.17 class (of coding process):** Lossy or lossless coding processes.
- 3.1.18 code register:** The arithmetic encoder register containing the least significant bits of the partially completed entropy-coded segment. Alternatively, the arithmetic decoder register containing the most significant bits of a partially decoded entropy-coded segment.
- 3.1.19 coder:** An embodiment of a coding process.
- 3.1.20 coding:** Encoding or decoding.
- 3.1.21 coding model:** A procedure used to convert input data into symbols to be coded.
- 3.1.22 (coding) process:** A general term for referring to an encoding process, a decoding process, or both.
- 3.1.23 colour image:** A continuous-tone image that has more than one component.
- 3.1.24 columns:** Samples per line in a component.
- 3.1.25 component:** One of the two-dimensional arrays which comprise an image.
- 3.1.26 compressed data:** Either compressed image data or table specification data or both.
- 3.1.27 compressed image data:** A coded representation of an image, as specified in this Specification.
- 3.1.28 compression:** Reduction in the number of bits used to represent source image data.
- 3.1.29 conditional exchange:** The interchange of MPS and LPS probability intervals whenever the size of the LPS interval is greater than the size of the MPS interval (in arithmetic coding).
- 3.1.30 (conditional) probability estimate:** The probability value assigned to the LPS by the probability estimation state machine (in arithmetic coding).
- 3.1.31 conditioning table:** The set of parameters which select one of the defined relationships between prior coding decisions and the conditional probability estimates used in arithmetic coding.
- 3.1.32 context:** The set of previously coded binary decisions which is used to create the index to the probability estimation state machine (in arithmetic coding).
- 3.1.33 continuous-tone image:** An image whose components have more than one bit per sample.
- 3.1.34 data unit:** An 8×8 block of samples of one component in DCT-based processes; a sample in lossless processes.

- 3.1.35 DC coefficient:** The DCT coefficient for which the frequency is zero in both dimensions.
- 3.1.36 DC prediction:** The procedure used by DCT-based encoders whereby the quantized DC coefficient from the previously encoded 8×8 block of the same component is subtracted from the current quantized DC coefficient.
- 3.1.37 (DCT) coefficient:** The amplitude of a specific cosine basis function – may refer to an original DCT coefficient, to a quantized DCT coefficient, or to a dequantized DCT coefficient.
- 3.1.38 decoder:** An embodiment of a decoding process.
- 3.1.39 decoding process:** A process which takes as its input compressed image data and outputs a continuous-tone image.
- 3.1.40 default conditioning:** The values defined for the arithmetic coding conditioning tables at the beginning of coding of an image.
- 3.1.41 dequantization:** The inverse procedure to quantization by which the decoder recovers a representation of the DCT coefficients.
- 3.1.42 differential component:** The difference between an input component derived from the source image and the corresponding reference component derived from the preceding frame for that component (in hierarchical mode coding).
- 3.1.43 differential frame:** A frame in a hierarchical process in which differential components are either encoded or decoded.
- 3.1.44 (digital) reconstructed image (data):** A continuous-tone image which is the output of any decoder defined in this Specification.
- 3.1.45 (digital) source image (data):** A continuous-tone image used as input to any encoder defined in this Specification.
- 3.1.46 (digital) (still) image:** A set of two-dimensional arrays of integer data.
- 3.1.47 discrete cosine transform; DCT:** Either the forward discrete cosine transform or the inverse discrete cosine transform.
- 3.1.48 downsampling (filter):** A procedure by which the spatial resolution of an image is reduced (in hierarchical mode coding).
- 3.1.49 encoder:** An embodiment of an encoding process.
- 3.1.50 encoding process:** A process which takes as its input a continuous-tone image and outputs compressed image data.
- 3.1.51 entropy-coded (data) segment:** An independently decodable sequence of entropy encoded bytes of compressed image data.
- 3.1.52 (entropy-coded segment) pointer:** The variable which points to the most recently placed (or fetched) byte in the entropy encoded segment.
- 3.1.53 entropy decoder:** An embodiment of an entropy decoding procedure.
- 3.1.54 entropy decoding:** A lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder.
- 3.1.55 entropy encoder:** An embodiment of an entropy encoding procedure.
- 3.1.56 entropy encoding:** A lossless procedure which converts a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols.
- 3.1.57 extended (DCT-based) process:** A descriptive term for DCT-based encoding and decoding processes in which additional capabilities are added to the baseline sequential process.
- 3.1.58 forward discrete cosine transform; FDCT:** A mathematical transformation using cosine basis functions which converts a block of samples into a corresponding block of original DCT coefficients.

- 3.1.59 frame:** A group of one or more scans (all using the same DCT-based or lossless process) through the data of one or more of the components in an image.
- 3.1.60 frame header:** A marker segment that contains a start-of-frame marker and associated frame parameters that are coded at the beginning of a frame.
- 3.1.61 frequency:** A two-dimensional index into the two-dimensional array of DCT coefficients.
- 3.1.62 (frequency) band:** A contiguous group of coefficients from the zig-zag sequence (in progressive mode coding).
- 3.1.63 full progression:** A process which uses both spectral selection and successive approximation (in progressive mode coding).
- 3.1.64 grayscale image:** A continuous-tone image that has only one component.
- 3.1.65 hierarchical:** A mode of operation for coding an image in which the first frame for a given component is followed by frames which code the differences between the source data and the reconstructed data from the previous frame for that component. Resolution changes are allowed between frames.
- 3.1.66 hierarchical decoder:** A sequence of decoder processes in which the first frame for each component is followed by frames which decode an array of differences for each component and adds it to the reconstructed data from the preceding frame for that component.
- 3.1.67 hierarchical encoder:** The mode of operation in which the first frame for each component is followed by frames which encode the array of differences between the source data and the reconstructed data from the preceding frame for that component.
- 3.1.68 horizontal sampling factor:** The relative number of horizontal data units of a particular component with respect to the number of horizontal data units in the other components.
- 3.1.69 Huffman decoder:** An embodiment of a Huffman decoding procedure.
- 3.1.70 Huffman decoding:** An entropy decoding procedure which recovers the symbol from each variable length code produced by the Huffman encoder.
- 3.1.71 Huffman encoder:** An embodiment of a Huffman encoding procedure.
- 3.1.72 Huffman encoding:** An entropy encoding procedure which assigns a variable length code to each input symbol.
- 3.1.73 Huffman table:** The set of variable length codes required in a Huffman encoder and Huffman decoder.
- 3.1.74 image data:** Either source image data or reconstructed image data.
- 3.1.75 interchange format:** The representation of compressed image data for exchange between application environments.
- 3.1.76 interleaved:** The descriptive term applied to the repetitive multiplexing of small groups of data units from each component in a scan in a specific order.
- 3.1.77 inverse discrete cosine transform; IDCT:** A mathematical transformation using cosine basis functions which converts a block of dequantized DCT coefficients into a corresponding block of samples.
- 3.1.78 Joint Photographic Experts Group; JPEG:** The informal name of the committee which created this Specification. The "joint" comes from the CCITT and ISO/IEC collaboration.
- 3.1.79 latent output:** Output of the arithmetic encoder which is held, pending resolution of carry-over (in arithmetic coding).
- 3.1.80 less probable symbol; LPS:** For a binary decision, the decision value which has the smaller probability.
- 3.1.81 level shift:** A procedure used by DCT-based encoders and decoders whereby each input sample is either converted from an unsigned representation to a two's complement representation or from a two's complement representation to an unsigned representation.

- 3.1.82 lossless:** A descriptive term for encoding and decoding processes and procedures in which the output of the decoding procedure(s) is identical to the input to the encoding procedure(s).
- 3.1.83 lossless coding:** The mode of operation which refers to any one of the coding processes defined in this Specification in which all of the procedures are lossless (see Annex H).
- 3.1.84 lossy:** A descriptive term for encoding and decoding processes which are not lossless.
- 3.1.85 marker:** A two-byte code in which the first byte is hexadecimal FF (X'FF') and the second byte is a value between 1 and hexadecimal FE (X'FE').
- 3.1.86 marker segment:** A marker and associated set of parameters.
- 3.1.87 MCU-row:** The smallest sequence of MCU which contains at least one line of samples or one block-row from every component in the scan.
- 3.1.88 minimum coded unit; MCU:** The smallest group of data units that is coded.
- 3.1.89 modes (of operation):** The four main categories of image coding processes defined in this Specification.
- 3.1.90 more probable symbol; MPS:** For a binary decision, the decision value which has the larger probability.
- 3.1.91 non-differential frame:** The first frame for any components in a hierarchical encoder or decoder. The components are encoded or decoded without subtraction from reference components. The term refers also to any frame in modes other than the hierarchical mode.
- 3.1.92 non-interleaved:** The descriptive term applied to the data unit processing sequence when the scan has only one component.
- 3.1.93 parameters:** Fixed length integers 4, 8 or 16 bits in length, used in the compressed data formats.
- 3.1.94 point transform:** Scaling of a sample or DCT coefficient.
- 3.1.95 precision:** Number of bits allocated to a particular sample or DCT coefficient.
- 3.1.96 predictor:** A linear combination of previously reconstructed values (in lossless mode coding).
- 3.1.97 probability estimation state machine:** An interlinked table of probability values and indices which is used to estimate the probability of the LPS (in arithmetic coding).
- 3.1.98 probability interval:** The probability of a particular sequence of binary decisions within the ordered set of all possible sequences (in arithmetic coding).
- 3.1.99 (probability) sub-interval:** A portion of a probability interval allocated to either of the two possible binary decision values (in arithmetic coding).
- 3.1.100 procedure:** A set of steps which accomplishes one of the tasks which comprise an encoding or decoding process.
- 3.1.101 process:** See coding process.
- 3.1.102 progressive (coding):** One of the DCT-based processes defined in this Specification in which each scan typically improves the quality of the reconstructed image.
- 3.1.103 progressive DCT-based:** The mode of operation which refers to any one of the processes defined in Annex G.
- 3.1.104 quantization table:** The set of 64 quantization values used to quantize the DCT coefficients.
- 3.1.105 quantization value:** An integer value used in the quantization procedure.
- 3.1.106 quantize:** The act of performing the quantization procedure for a DCT coefficient.
- 3.1.107 reference (reconstructed) component:** Reconstructed component data which is used in a subsequent frame of a hierarchical encoder or decoder process (in hierarchical mode coding).

- 3.1.108 renormalization:** The doubling of the probability interval and the code register value until the probability interval exceeds a fixed minimum value (in arithmetic coding).
- 3.1.109 restart interval:** The integer number of MCUs processed as an independent sequence within a scan.
- 3.1.110 restart marker:** The marker that separates two restart intervals in a scan.
- 3.1.111 run (length):** Number of consecutive symbols of the same value.
- 3.1.112 sample:** One element in the two-dimensional array which comprises a component.
- 3.1.113 sample-interleaved:** The descriptive term applied to the repetitive multiplexing of small groups of samples from each component in a scan in a specific order.
- 3.1.114 scan:** A single pass through the data for one or more of the components in an image.
- 3.1.115 scan header:** A marker segment that contains a start-of-scan marker and associated scan parameters that are coded at the beginning of a scan.
- 3.1.116 sequential (coding):** One of the lossless or DCT-based coding processes defined in this Specification in which each component of the image is encoded within a single scan.
- 3.1.117 sequential DCT-based:** The mode of operation which refers to any one of the processes defined in Annex F.
- 3.1.118 spectral selection:** A progressive coding process in which the zig-zag sequence is divided into bands of one or more contiguous coefficients, and each band is coded in one scan.
- 3.1.119 stack counter:** The count of X'FF' bytes which are held, pending resolution of carry-over in the arithmetic encoder.
- 3.1.120 statistical conditioning:** The selection, based on prior coding decisions, of one estimate out of a set of conditional probability estimates (in arithmetic coding).
- 3.1.121 statistical model:** The assignment of a particular conditional probability estimate to each of the binary arithmetic coding decisions.
- 3.1.122 statistics area:** The array of statistics bins required for a coding process which uses arithmetic coding.
- 3.1.123 statistics bin:** The storage location where an index is stored which identifies the value of the conditional probability estimate used for a particular arithmetic coding binary decision.
- 3.1.124 successive approximation:** A progressive coding process in which the coefficients are coded with reduced precision in the first scan, and precision is increased by one bit with each succeeding scan.
- 3.1.125 table specification data:** The coded representation from which the tables used in the encoder and decoder are generated and their destinations specified.
- 3.1.126 transcoder:** A procedure for converting compressed image data of one encoder process to compressed image data of another encoder process.
- 3.1.127 (uniform) quantization:** The procedure by which DCT coefficients are linearly scaled in order to achieve compression.
- 3.1.128 upsampling (filter):** A procedure by which the spatial resolution of an image is increased (in hierarchical mode coding).
- 3.1.129 vertical sampling factor:** The relative number of vertical data units of a particular component with respect to the number of vertical data units in the other components in the frame.
- 3.1.130 zero byte:** The X'00' byte.
- 3.1.131 zig-zag sequence:** A specific sequential ordering of the DCT coefficients from (approximately) lowest spatial frequency to highest.
- 3.1.132 3-sample predictor:** A linear combination of the three nearest neighbor reconstructed samples to the left and above (in lossless mode coding).

3.2 Symbols

The symbols used in this Specification are listed below.

A	probability interval
AC	AC DCT coefficient
AC_{ji}	AC coefficient predicted from DC values
A_h	successive approximation bit position, high
A_l	successive approximation bit position, low
A_{p_i}	<i>i</i> th 8-bit parameter in APP _n segment
APP_n	marker reserved for application segments
B	current byte in compressed data
B2	next byte in compressed data when B = X'FF'
BE	counter for buffered correction bits for Huffman coding in the successive approximation process
BITS	16-byte list containing number of Huffman codes of each length
BP	pointer to compressed data
BPST	pointer to byte before start of entropy-coded segment
BR	counter for buffered correction bits for Huffman coding in the successive approximation process
B_x	byte modified by a carry-over
C	value of bit stream in code register
C_i	component identifier for frame
C_u	horizontal frequency dependent scaling factor in DCT
C_v	vertical frequency dependent scaling factor in DCT
CE	conditional exchange
C-low	low order 16 bits of the arithmetic decoder code register
C_{m_i}	<i>i</i> th 8-bit parameter in COM segment
CNT	bit counter in NEXTBYTE procedure
CODE	Huffman code value
CODESIZE(V)	code size for symbol V
COM	comment marker
C_s	conditioning table value
C_{s_i}	component identifier for scan
CT	renormalization shift counter
C_x	high order 16 bits of arithmetic decoder code register
CX	conditional exchange
d_{ji}	data unit from horizontal position <i>i</i> , vertical position <i>j</i>
d_{ji^k}	d _{ji} for component <i>k</i>
D	decision decoded

Da	in DC coding, the DC difference coded for the previous block from the same component; in lossless coding, the difference coded for the sample immediately to the left
DAC	define-arithmetic-coding-conditioning marker
Db	the difference coded for the sample immediately above
DC	DC DCT coefficient
DC _i	DC coefficient for <i>i</i> th block in component
DC _k	<i>k</i> th DC value used in prediction of AC coefficients
DHP	define hierarchical progression marker
DHT	define-Huffman-tables marker
DIFF	difference between quantized DC and prediction
DNL	define-number-of-lines marker
DQT	define-quantization-tables marker
DRI	define restart interval marker
E	exponent in magnitude category upper bound
EC	event counter
ECS	entropy-coded segment
ECS _i	<i>i</i> th entropy-coded segment
Eh	horizontal expansion parameter in EXP segment
EHUFCO	Huffman code table for encoder
EHUFSI	encoder table of Huffman code sizes
EOB	end-of-block for sequential; end-of-band for progressive
EOB _n	run length category for EOB runs
EOB _x	position of EOB in previous successive approximation scan
EOB ₀ , EOB ₁ , ..., EOB ₁₄	run length categories for EOB runs
EOI	end-of-image marker
Ev	vertical expansion parameter in EXP segment
EXP	expand reference components marker
FREQ(V)	frequency of occurrence of symbol V
H _i	horizontal sampling factor for <i>i</i> th component
H _{max}	largest horizontal sampling factor
HUFFCODE	list of Huffman codes corresponding to lengths in HUFFSIZE
HUFFSIZE	list of code lengths
HUFFVAL	list of values assigned to each Huffman code
i	subscript index
I	integer variable
Index(S)	index to probability estimation state machine table for context index S
j	subscript index
J	integer variable

JPG	marker reserved for JPEG extensions
JPG _n	marker reserved for JPEG extensions
k	subscript index
K	integer variable
Kmin	index of 1st AC coefficient in band (1 for sequential DCT)
Kx	conditioning parameter for AC arithmetic coding model
L	DC and lossless coding conditioning lower bound parameter
L _i	element in BITS list in DHT segment
L _i (t)	element in BITS list in the DHT segment for Huffman table t
La	length of parameters in APP _n segment
LASTK	largest value of K
Lc	length of parameters in COM segment
Ld	length of parameters in DNL segment
Le	length of parameters in EXP segment
Lf	length of frame header parameters
Lh	length of parameters in DHT segment
Lp	length of parameters in DAC segment
LPS	less probable symbol (in arithmetic coding)
Lq	length of parameters in DQT segment
Lr	length of parameters in DRI segment
Ls	length of scan header parameters
LSB	least significant bit
m	modulo 8 counter for RST _m marker
m _t	number of V _{i,j} parameters for Huffman table t
M	bit mask used in coding magnitude of V
Mn	n th statistics bin for coding magnitude bit pattern category
MAXCODE	table with maximum value of Huffman code for each code length
MCU	minimum coded unit
MCU _i	i th MCU
MCUR	number of MCU required to make up one MCU-row
MINCODE	table with minimum value of Huffman code for each code length
MPS	more probable symbol (in arithmetic coding)
MPS(S)	more probable symbol for context-index S
MSB	most significant bit
M2, M3, M4, ..., M15	designation of context-indices for coding of magnitude bits in the arithmetic coding models
n	integer variable
N	data unit counter for MCU coding
N/A	not applicable

Nb	number of data units in MCU
Next_Index_LPS	new value of Index(S) after a LPS renormalization
Next_Index_MPS	new value of Index(S) after a MPS renormalization
Nf	number of components in frame
NL	number of lines defined in DNL segment
Ns	number of components in scan
OTHERS(V)	index to next symbol in chain
P	sample precision
Pq	quantizer precision parameter in DQT segment
Pq(t)	quantizer precision parameter in DQT segment for quantization table t
PRED	quantized DC coefficient from the most recently coded block of the component
Pt	point transform parameter
Px	calculated value of sample
Q _{ji}	quantizer value for coefficient AC _{ji}
Q _{vu}	quantization value for DCT coefficient S _{vu}
Q ₀₀	quantizer value for DC coefficient
QAC _{ji}	quantized AC coefficient predicted from DC values
QDC _k	kth quantized DC value used in prediction of AC coefficients
Qe	LPS probability estimate
Qe(S)	LPS probability estimate for context index S
Qk	kth element of 64 quantization elements in DQT segment
r _{vu}	reconstructed image sample
R	length of run of zero amplitude AC coefficients
R _{vu}	dequantized DCT coefficient
Ra	reconstructed sample value
Rb	reconstructed sample value
Rc	reconstructed sample value
Rd	rounding in prediction calculation
RES	reserved markers
Ri	restart interval in DRI segment
RRRR	4-bit value of run length of zero AC coefficients
RS	composite value used in Huffman coding of AC coefficients
RST _m	restart marker number m
s _{yx}	reconstructed value from IDCT
S	context index
S _{vu}	DCT coefficient at horizontal frequency u, vertical frequency v

SC	context-index for coding of correction bit in successive approximation coding
Se	end of spectral selection band in zig-zag sequence
SE	context-index for coding of end-of-block or end-of-band
SI	Huffman code size
SIGN	1 if decoded sense of sign is negative and 0 if decoded sense of sign is positive
SIZE	length of a Huffman code
SLL	shift left logical operation
SLL α β	logical shift left of α by β bits
SN	context-index for coding of first magnitude category when V is negative
SOF ₀	baseline DCT process frame marker
SOF ₁	extended sequential DCT frame marker, Huffman coding
SOF ₂	progressive DCT frame marker, Huffman coding
SOF ₃	lossless process frame marker, Huffman coding
SOF ₅	differential sequential DCT frame marker, Huffman coding
SOF ₆	differential progressive DCT frame marker, Huffman coding
SOF ₇	differential lossless process frame marker, Huffman coding
SOF ₉	sequential DCT frame marker, arithmetic coding
SOF ₁₀	progressive DCT frame marker, arithmetic coding
SOF ₁₁	lossless process frame marker, arithmetic coding
SOF ₁₃	differential sequential DCT frame marker, arithmetic coding
SOF ₁₄	differential progressive DCT frame marker, arithmetic coding
SOF ₁₅	differential lossless process frame marker, arithmetic coding
SOI	start-of-image marker
SOS	start-of-scan marker
SP	context-index for coding of first magnitude category when V is positive
Sq _{vu}	quantized DCT coefficient
SRL	shift right logical operation
SRL α β	logical shift right of α by β bits
Ss	start of spectral selection band in zig-zag sequence
SS	context-index for coding of sign decision
SSSS	4-bit size category of DC difference or AC coefficient amplitude
ST	stack counter
Switch_MPS	parameter controlling inversion of sense of MPS
Sz	parameter used in coding magnitude of V
S0	context-index for coding of V = 0 decision
t	summation index for parameter limits computation
T	temporary variable

Ta_j	AC entropy table destination selector for j th component in scan
Tb	arithmetic conditioning table destination identifier
Tc	Huffman coding or arithmetic coding table class
Td_j	DC entropy table destination selector for j th component in scan
TEM	temporary marker
Th	Huffman table destination identifier in DHT segment
Tq	quantization table destination identifier in DQT segment
Tq_i	quantization table destination selector for i th component in frame
U	DC and lossless coding conditioning upper bound parameter
V	symbol or value being either encoded or decoded
V_i	vertical sampling factor for i th component
V_{ij}	j th value for length i in HUFFVAL
V_{max}	largest vertical sampling factor
V_t	temporary variable
VALPTR	list of indices for first value in HUFFVAL for each code length
V1	symbol value
V2	symbol value
x_i	number of columns in i th component
X	number of samples per line in component with largest horizontal dimension
X_i	i th statistics bin for coding magnitude category decision
$X1, X2, X3, \dots, X15$	designation of context-indices for coding of magnitude categories in the arithmetic coding models
XHUFÇO	extended Huffman code table
XHUFSl	table of sizes of extended Huffman codes
X' values'	values within the quotes are hexadecimal
y_i	number of lines in i th component
Y	number of lines in component with largest vertical dimension
ZRL	value in HUFFVAL assigned to run of 16 zero coefficients
$ZZ(K)$	K th element in zig-zag sequence of quantized DCT coefficients
$ZZ(0)$	quantized DC coefficient in zig-zag sequence order

4 General

The purpose of this clause is to give an informative overview of the elements specified in this Specification. Another purpose is to introduce many of the terms which are defined in clause 3. These terms are printed in *italics* upon first usage in this clause.

4.1 Elements specified in this Specification

There are three elements specified in this Specification:

- An *encoder* is an embodiment of an *encoding process*. As shown in Figure 1, an encoder takes as input *digital source image data* and *table specifications*, and by means of a specified set of *procedures* generates as output *compressed image data*.
- A *decoder* is an embodiment of a *decoding process*. As shown in Figure 2, a decoder takes as input *compressed image data* and *table specifications*, and by means of a specified set of *procedures* generates as output *digital reconstructed image data*.
- The *interchange format*, shown in Figure 3, is a compressed image data representation which includes all table specifications used in the encoding process. The interchange format is for exchange between *application environments*.

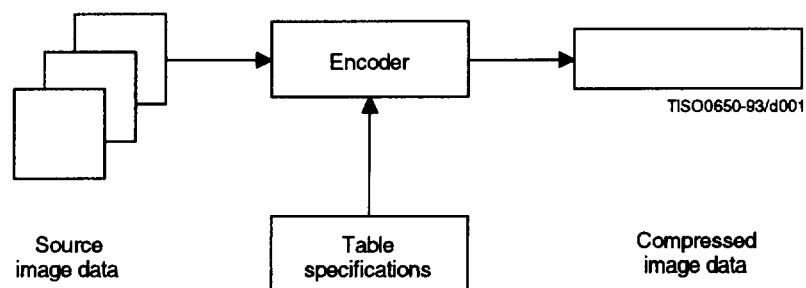


Figure 1 – Encoder

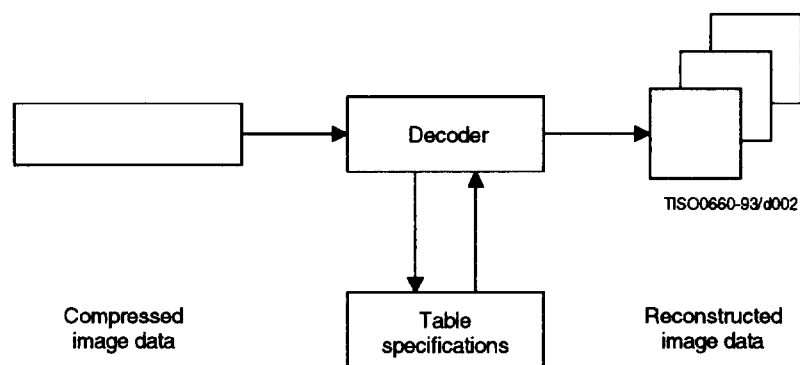


Figure 2 – Decoder

Figures 1 and 2 illustrate the general case for which the *continuous-tone* source and reconstructed image data consist of multiple *components*. (A *colour* image consists of multiple components; a *grayscale* image consists only of a single component.) A significant portion of this Specification is concerned with how to handle multiple-component images in a flexible, application-independent way.

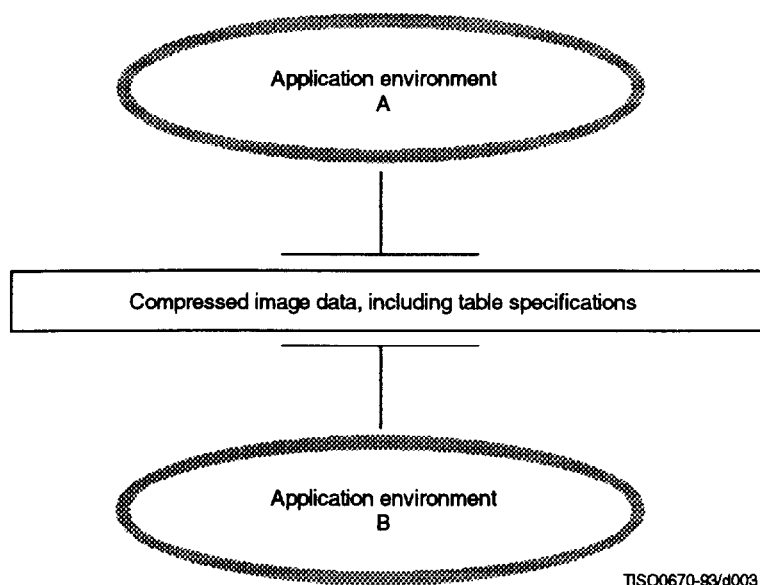


Figure 3 – Interchange format for compressed image data

These figures are also meant to show that the same tables specified for an encoder to use to compress a particular image must be provided to a decoder to reconstruct that image. However, this Specification does not specify how applications should associate tables with compressed image data, nor how they should represent source image data generally within their specific environments.

Consequently, this Specification also specifies the interchange format shown in Figure 3, in which table specifications are included within compressed image data. An image compressed with a specified encoding process within one application environment, A, is passed to a different environment, B, by means of the interchange format. The interchange format does not specify a complete coded image representation. Application-dependent information, e.g. colour space, is outside the scope of this Specification.

4.2 Lossy and lossless compression

This Specification specifies two *classes* of encoding and decoding processes, *lossy* and *lossless* processes. Those based on the *discrete cosine transform* (DCT) are lossy, thereby allowing substantial *compression* to be achieved while producing a reconstructed image with high visual fidelity to the encoder's source image.

The simplest DCT-based *coding process* is referred to as the *baseline sequential* process. It provides a capability which is sufficient for many applications. There are additional DCT-based processes which extend the baseline sequential process to a broader range of applications. In any decoder using *extended DCT-based decoding processes*, the baseline decoding process is required to be present in order to provide a default decoding capability.

The second class of coding processes is not based upon the DCT and is provided to meet the needs of applications requiring lossless compression. These lossless encoding and decoding processes are used independently of any of the DCT-based processes.

A table summarizing the relationship among these lossy and lossless coding processes is included in 4.11.

The amount of compression provided by any of the various processes is dependent on the characteristics of the particular image being compressed, as well as on the picture quality desired by the application and the desired speed of compression and decompression.

4.3 DCT-based coding

Figure 4 shows the main procedures for all encoding processes based on the DCT. It illustrates the special case of a single-component image; this is an appropriate simplification for overview purposes, because all processes specified in this Specification operate on each image component independently.

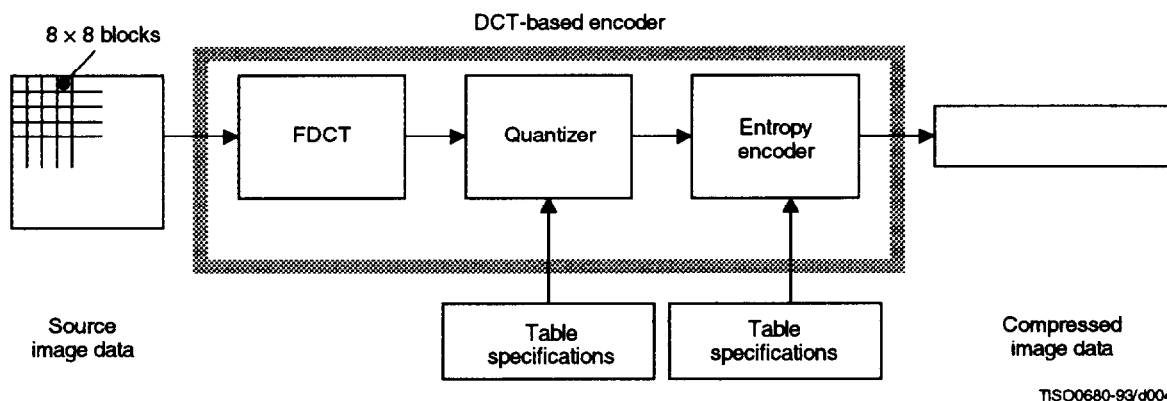


Figure 4 – DCT-based encoder simplified diagram

In the encoding process the input component's *samples* are grouped into 8×8 *blocks*, and each block is transformed by the *forward DCT* (FDCT) into a set of 64 values referred to as *DCT coefficients*. One of these values is referred to as the *DC coefficient* and the other 63 as the *AC coefficients*.

Each of the 64 coefficients is then *quantized* using one of 64 corresponding values from a *quantization table* (determined by one of the table specifications shown in Figure 4). No default values for quantization tables are specified in this Specification; applications may specify values which customize picture quality for their particular image characteristics, display devices, and viewing conditions.

After quantization, the DC coefficient and the 63 AC coefficients are prepared for *entropy encoding*, as shown in Figure 5. The previous quantized DC coefficient is used to predict the current quantized DC coefficient, and the difference is encoded. The 63 quantized AC coefficients undergo no such differential encoding, but are converted into a one-dimensional *zig-zag sequence*, as shown in Figure 5.

The quantized coefficients are then passed to an entropy encoding procedure which compresses the data further. One of two entropy coding procedures can be used, as described in 4.6. If *Huffman encoding* is used, *Huffman table specifications* must be provided to the encoder. If *arithmetic encoding* is used, *arithmetic coding conditioning table specifications* may be provided, otherwise the default conditioning table specifications shall be used.

Figure 6 shows the main procedures for all DCT-based decoding processes. Each step shown performs essentially the inverse of its corresponding main procedure within the encoder. The entropy decoder decodes the zig-zag sequence of quantized DCT coefficients. After *dequantization* the DCT coefficients are transformed to an 8×8 block of samples by the *inverse DCT* (IDCT).

4.4 Lossless coding

Figure 7 shows the main procedures for the lossless encoding processes. A *predictor* combines the reconstructed values of up to three neighbourhood samples at positions a, b, and c to form a prediction of the sample at position x as shown in Figure 8. This prediction is then subtracted from the actual value of the sample at position x, and the difference is losslessly entropy-coded by either Huffman or arithmetic coding.

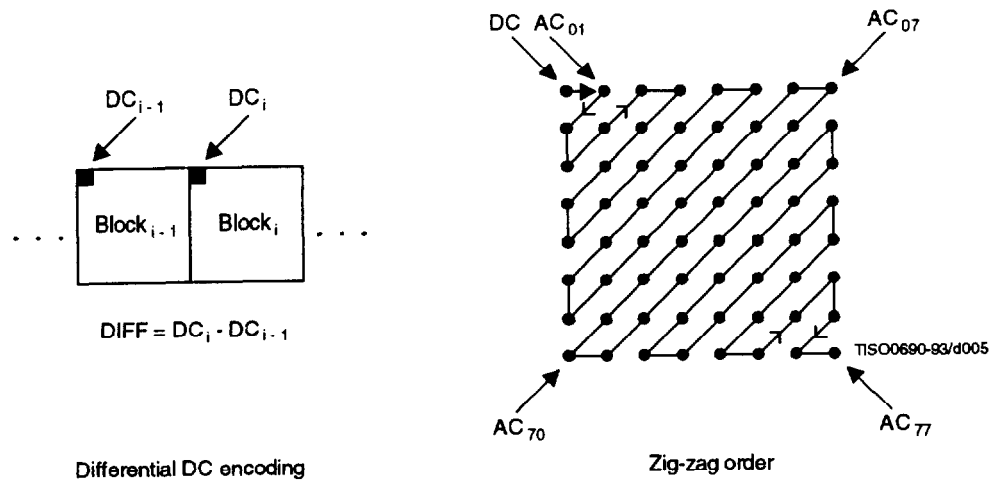


Figure 5 – Preparation of quantized coefficients for entropy encoding

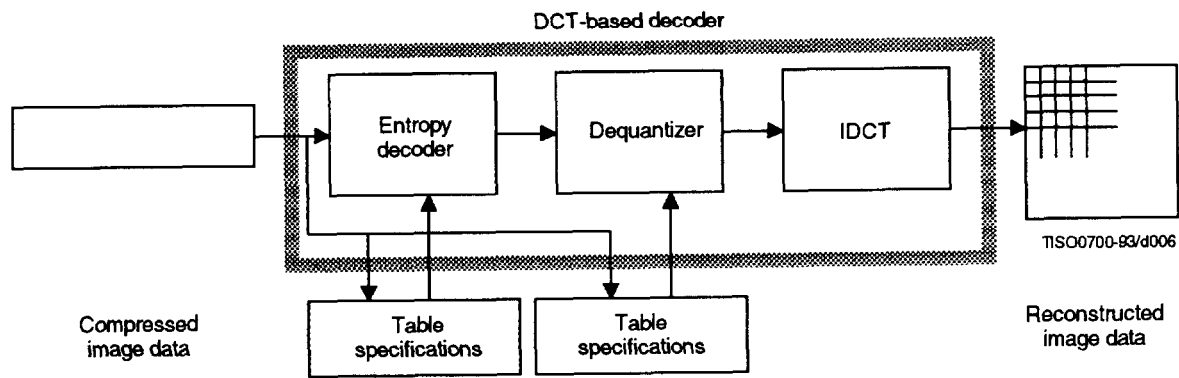


Figure 6 – DCT-based decoder simplified diagram

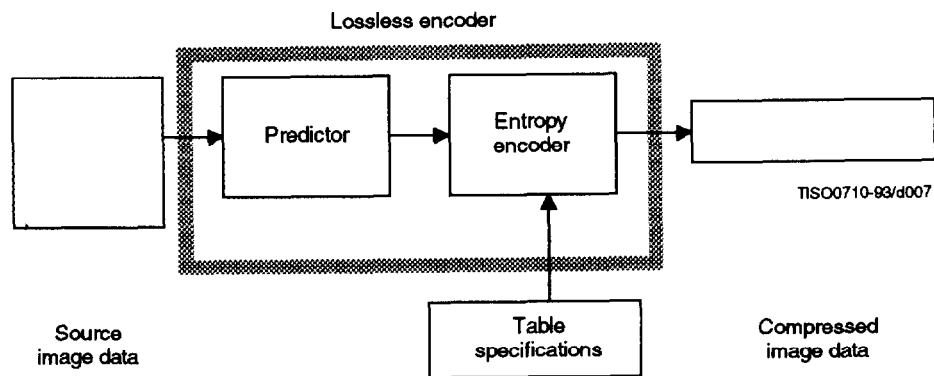


Figure 7 – Lossless encoder simplified diagram

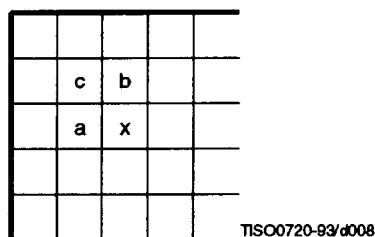


Figure 8 – 3-sample prediction neighbourhood

This encoding process may also be used in a slightly modified way, whereby the *precision* of the input samples is reduced by one or more bits prior to the lossless coding. This achieves higher compression than the lossless process (but lower compression than the DCT-based processes for equivalent visual fidelity), and limits the reconstructed image's worst-case sample error to the amount of input precision reduction.

4.5 Modes of operation

There are four distinct *modes of operation* under which the various coding processes are defined: *sequential DCT-based*, *progressive DCT-based*, *lossless*, and *hierarchical*. (Implementations are not required to provide all of these.) The lossless mode of operation was described in 4.4. The other modes of operation are compared as follows.

For the sequential DCT-based mode, 8×8 sample blocks are typically input block by block from left to right, and block-row by block-row from top to bottom. After a block has been transformed by the forward DCT, quantized and prepared for entropy encoding, all 64 of its quantized DCT coefficients can be immediately entropy encoded and output as part of the compressed image data (as was described in 4.3), thereby minimizing coefficient storage requirements.

For the progressive DCT-based mode, 8×8 blocks are also typically encoded in the same order, but in multiple *scans* through the image. This is accomplished by adding an image-sized coefficient memory buffer (not shown in Figure 4) between the quantizer and the entropy encoder. As each block is transformed by the forward DCT and quantized, its coefficients are stored in the buffer. The DCT coefficients in the buffer are then partially encoded in each of multiple scans. The typical sequence of image presentation at the output of the decoder for sequential versus progressive modes of operation is shown in Figure 9.

There are two procedures by which the quantized coefficients in the buffer may be partially encoded within a scan. First, only a specified *band* of coefficients from the zig-zag sequence need be encoded. This procedure is called *spectral selection*, because each band typically contains coefficients which occupy a lower or higher part of the *frequency* spectrum for that 8×8 block. Secondly, the coefficients within the current band need not be encoded to their full (quantized) accuracy within each scan. Upon a coefficient's first encoding, a specified number of most significant bits is encoded first. In subsequent scans, the less significant bits are then encoded. This procedure is called *successive approximation*. Either procedure may be used separately, or they may be mixed in flexible combinations.

In hierarchical mode, an image is encoded as a sequence of *frames*. These frames provide *reference reconstructed components* which are usually needed for prediction in subsequent frames. Except for the first frame for a given component, *differential frames* encode the difference between source components and reference reconstructed components. The coding of the differences may be done using only DCT-based processes, only lossless processes, or DCT-based processes with a final lossless process for each component. *Downsampling* and *upsampling filters* may be used to provide a pyramid of spatial resolutions as shown in Figure 10. Alternatively, the hierarchical mode can be used to improve the quality of the reconstructed components at a given spatial resolution.

Hierarchical mode offers a progressive presentation similar to the progressive DCT-based mode but is useful in environments which have multi-resolution requirements. Hierarchical mode also offers the capability of progressive coding to a final lossless stage.

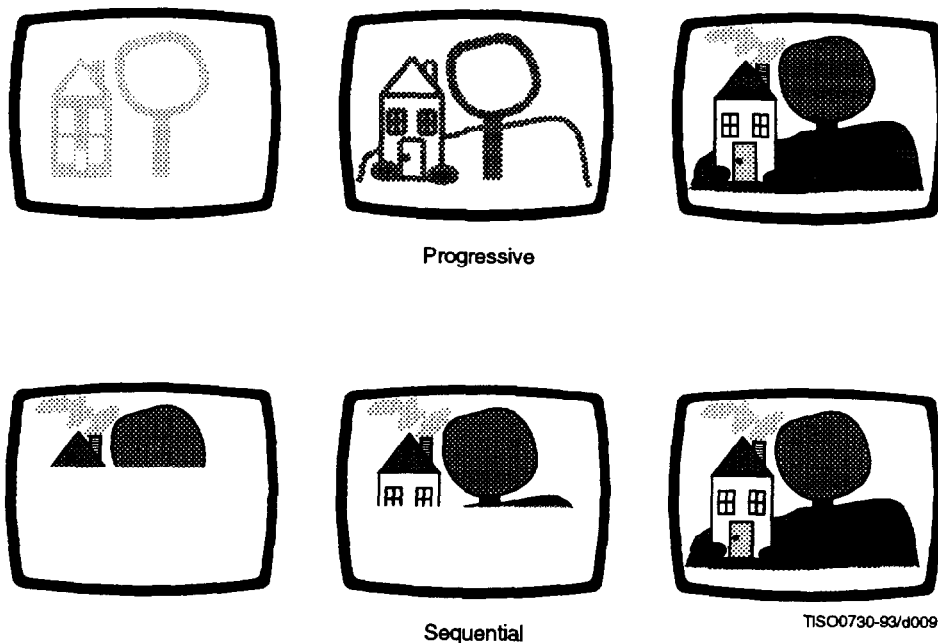


Figure 9 – Progressive versus sequential presentation

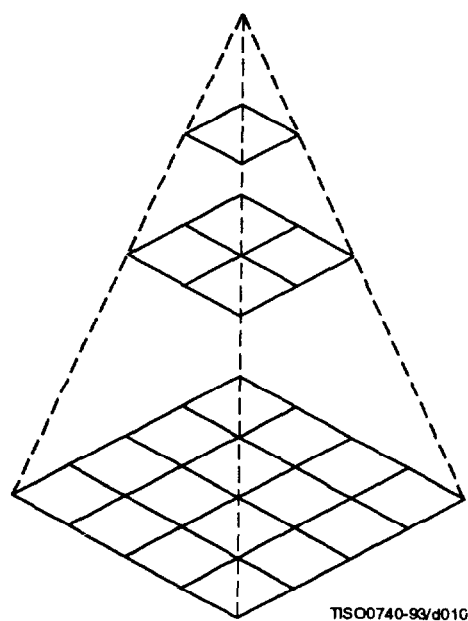


Figure 10 – Hierarchical multi-resolution encoding

4.6 Entropy coding alternatives

Two alternative entropy coding procedures are specified: Huffman coding and arithmetic coding. Huffman coding procedures use Huffman tables, determined by one of the table specifications shown in Figures 1 and 2. Arithmetic coding procedures use arithmetic coding conditioning tables, which may also be determined by a table specification. No default values for Huffman tables are specified, so that applications may choose tables appropriate for their own environments. Default tables are defined for the arithmetic coding conditioning.

The baseline sequential process uses Huffman coding, while the extended DCT-based and lossless processes may use either Huffman or arithmetic coding.

4.7 Sample precision

For DCT-based processes, two alternative sample precisions are specified: either 8 bits or 12 bits per sample. Applications which use samples with other precisions can use either 8-bit or 12-bit precision by shifting their source image samples appropriately. The baseline process uses only 8-bit precision. DCT-based implementations which handle 12-bit source image samples are likely to need greater computational resources than those which handle only 8-bit source images. Consequently in this Specification separate normative requirements are defined for 8-bit and 12-bit DCT-based processes.

For lossless processes the sample precision is specified to be from 2 to 16 bits.

4.8 Multiple-component control

Subclauses 4.3 and 4.4 give an overview of one major part of the encoding and decoding processes – those which operate on the sample values in order to achieve compression. There is another major part as well – the procedures which control the order in which the image data from multiple components are processed to create the compressed data, and which ensure that the proper set of table data is applied to the proper *data units* in the image. (A data unit is a sample for lossless processes and an 8×8 block of samples for DCT-based processes.)

4.8.1 Interleaving multiple components

Figure 11 shows an example of how an encoding process selects between multiple source image components as well as multiple sets of table data, when performing its encoding procedures. The source image in this example consists of the three components A, B and C, and there are two sets of table specifications. (This simplified view does not distinguish between the quantization tables and entropy coding tables.)

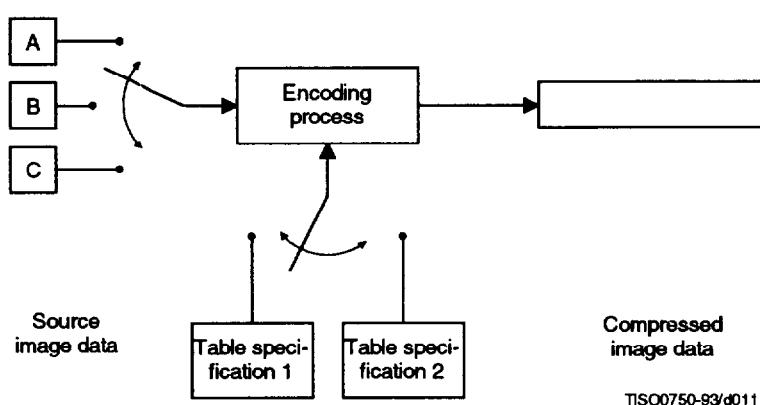


Figure 11 – Component-interleave and table-switching control

In sequential mode, encoding is *non-interleaved* if the encoder compresses all image data units in component A before beginning component B, and then in turn all of B before C. Encoding is *interleaved* if the encoder compresses a data unit from A, a data unit from B, a data unit from C, then back to A, etc. These alternatives are illustrated in Figure 12, which shows a case in which all three image components have identical dimensions: X *columns* by Y *lines*, for a total of n data units each.

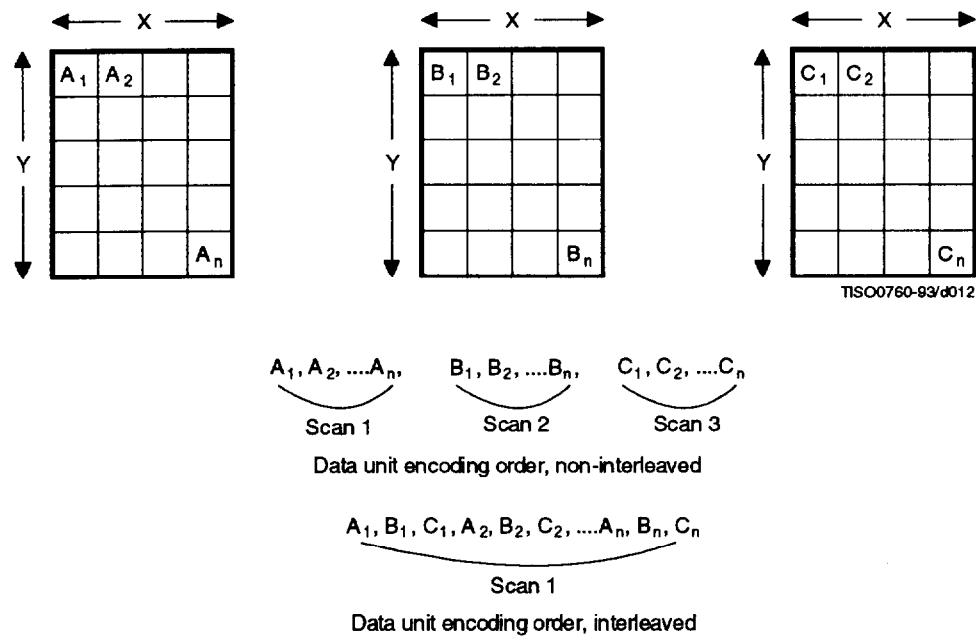


Figure 12 – Interleaved versus non-interleaved encoding order

These control procedures are also able to handle cases in which the source image components have different dimensions. Figure 13 shows a case in which two of the components, B and C, have half the number of horizontal samples relative to component A. In this case, two data units from A are interleaved with one each from B and C. Cases in which components of an image have more complex relationships, such as different horizontal and vertical dimensions, can be handled as well. (See Annex A.)

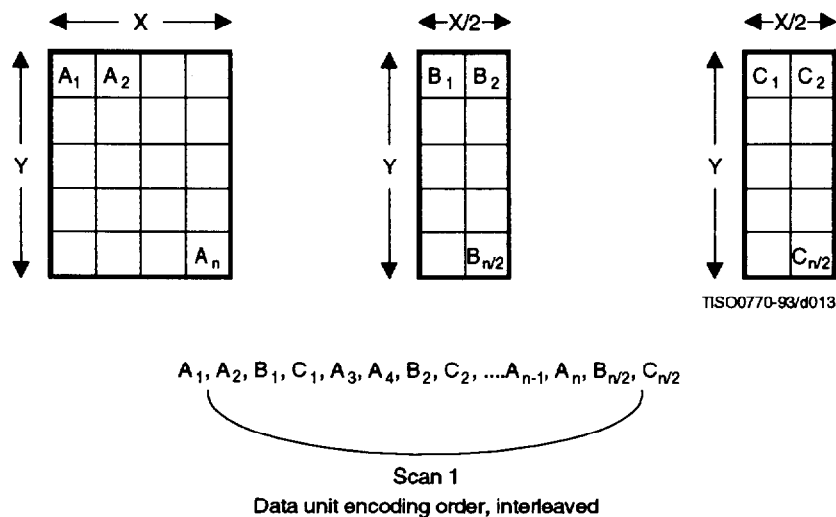


Figure 13 – Interleaved order for components with different dimensions

4.8.2 Minimum coded unit

Related to the concepts of multiple-component interleave is the *minimum coded unit* (MCU). If the compressed image data is non-interleaved, the MCU is defined to be one data unit. For example, in Figure 12 the MCU for the non-interleaved case is a single data unit. If the compressed data is interleaved, the MCU contains one or more data units from each component. For the interleaved case in Figure 12, the (first) MCU consists of the three interleaved data units A_1 , B_1 , C_1 . In the example of Figure 13, the (first) MCU consists of the four data units A_1 , A_2 , B_1 , C_1 .

4.9 Structure of compressed data

Figures 1, 2, and 3 all illustrate slightly different views of compressed image data. Figure 1 shows this data as the output of an encoding process, Figure 2 shows it as the input to a decoding process, and Figure 3 shows compressed image data in the interchange format, at the interface between applications.

Compressed image data are described by a uniform structure and set of *parameters* for both classes of encoding processes (lossy or lossless), and for all modes of operation (sequential, progressive, lossless, and hierarchical). The various parts of the compressed image data are identified by special two-byte codes called *markers*. Some markers are followed by particular sequences of parameters, as in the case of table specifications, *frame header*, or *scan header*. Others are used without parameters for functions such as marking the start-of-image and end-of-image. When a marker is associated with a particular sequence of parameters, the marker and its parameters comprise a *marker segment*.

The data created by the entropy encoder are also segmented, and one particular marker – the *restart marker* – is used to isolate *entropy-coded data segments*. The encoder outputs the restart markers, intermixed with the entropy-coded data, at regular *restart intervals* of the source image data. Restart markers can be identified without having to decode the compressed data to find them. Because they can be independently decoded, they have application-specific uses, such as parallel encoding or decoding, isolation of data corruptions, and semi-random access of entropy-coded segments.

There are three compressed data formats:

- a) the interchange format;
- b) the *abbreviated format* for compressed image data;
- c) the abbreviated format for table-specification data.

4.9.1 Interchange format

In addition to certain required marker segments and the entropy-coded segments, the interchange format shall include the marker segments for all quantization and entropy-coding table specifications needed by the decoding process. This guarantees that a compressed image can cross the boundary between application environments, regardless of how each environment internally associates tables with compressed image data.

4.9.2 Abbreviated format for compressed image data

The abbreviated format for compressed image data is identical to the interchange format, except that it does not include all tables required for decoding. (It may include some of them.) This format is intended for use within applications where alternative mechanisms are available for supplying some or all of the table-specification data needed for decoding.

4.9.3 Abbreviated format for table-specification data

This format contains only table-specification data. It is a means by which the application may install in the decoder the tables required to subsequently reconstruct one or more images.

4.10 Image, frame, and scan

Compressed image data consists of only one image. An image contains only one frame in the cases of sequential and progressive coding processes; an image contains multiple frames for the hierarchical mode.

A frame contains one or more scans. For sequential processes, a scan contains a complete encoding of one or more image components. In Figures 12 and 13, the frame consists of three scans when non-interleaved, and one scan if all three components are interleaved together. The frame could also consist of two scans: one with a non-interleaved component, the other with two components interleaved.

For progressive processes, a scan contains a partial encoding of all data units from one or more image components. Components shall not be interleaved in progressive mode, except for the DC coefficients in the first scan for each component of a progressive frame.

4.11 Summary of coding processes

Table 1 provides a summary of the essential characteristics of the various coding processes specified in this Specification. The full specification of these processes is contained in Annexes F, G, H, and J.

Table 1 – Summary: Essential characteristics of coding processes

Baseline process (required for all DCT-based decoders)
• DCT-based process • Source image: 8-bit samples within each component • Sequential • Huffman coding: 2 AC and 2 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans
Extended DCT-based processes
• DCT-based process • Source image: 8-bit or 12-bit samples • Sequential or progressive • Huffman or arithmetic coding: 4 AC and 4 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans
Lossless processes
• Predictive process (not DCT-based) • Source image: P-bit samples ($2 \leq P \leq 16$) • Sequential • Huffman or arithmetic coding: 4 DC tables • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans
Hierarchical processes
• Multiple frames (non-differential and differential) • Uses extended DCT-based or lossless processes • Decoders shall process scans with 1, 2, 3, and 4 components • Interleaved and non-interleaved scans

5 Interchange format requirements

The interchange format is the coded representation of compressed image data for exchange between application environments.

The interchange format requirements are that any compressed image data represented in interchange format shall comply with the syntax and code assignments appropriate for the decoding process selected, as specified in Annex B.

Tests for whether compressed image data comply with these requirements are specified in Part 2 of this Specification.

6 Encoder requirements

An encoding process converts source image data to compressed image data. Each of Annexes F, G, H, and J specifies a number of distinct encoding processes for its particular mode of operation.

An encoder is an embodiment of one (or more) of the encoding processes specified in Annexes F, G, H, or J. In order to comply with this Specification, an encoder shall satisfy at least one of the following two requirements.

An encoder shall

- a) with appropriate accuracy, convert source image data to compressed image data which comply with the interchange format syntax specified in Annex B for the encoding process(es) embodied by the encoder;
- b) with appropriate accuracy, convert source image data to compressed image data which comply with the abbreviated format for compressed image data syntax specified in Annex B for the encoding process(es) embodied by the encoder.

For each of the encoding processes specified in Annexes F, G, H, and J, the compliance tests for the above requirements are specified in Part 2 of this Specification.

NOTE – There is **no requirement** in this Specification that any encoder which embodies one of the encoding processes specified in Annexes F, G, H, or J shall be able to operate for all ranges of the parameters which are allowed for that process. An encoder is only required to meet the compliance tests specified in Part 2, and to generate the compressed data format according to Annex B for those parameter values which it does use.

7 Decoder requirements

A decoding process converts compressed image data to reconstructed image data. Each of Annexes F, G, H, and J specifies a number of distinct decoding processes for its particular mode of operation.

A decoder is an embodiment of one (or more) of the decoding processes specified in Annexes F, G, H, or J. In order to comply with this Specification, a decoder shall satisfy all three of the following requirements.

A decoder shall

- a) with appropriate accuracy, convert to reconstructed image data any compressed image data with parameters within the range supported by the application, and which comply with the interchange format syntax specified in Annex B for the decoding process(es) embodied by the decoder;
- b) accept and properly store any table-specification data which comply with the abbreviated format for table-specification data syntax specified in Annex B for the decoding process(es) embodied by the decoder;
- c) with appropriate accuracy, convert to reconstructed image data any compressed image data which comply with the abbreviated format for compressed image data syntax specified in Annex B for the decoding process(es) embodied by the decoder, provided that the table-specification data required for decoding the compressed image data has previously been installed into the decoder.

Additionally, any DCT-based decoder, if it embodies any DCT-based decoding process other than baseline sequential, shall also embody the baseline sequential decoding process.

For each of the decoding processes specified in Annexes F, G, H, and J, the compliance tests for the above requirements are specified in Part 2 of this Specification.

Annex A

Mathematical definitions

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

A.1 Source image

Source images to which the encoding processes specified in this Specification can be applied are defined in this annex.

A.1.1 Dimensions and sampling factors

As shown in Figure A.1, a source image is defined to consist of N_f components. Each component, with unique identifier C_i , is defined to consist of a rectangular array of samples of x_i columns by y_i lines. The component dimensions are derived from two parameters, X and Y , where X is the maximum of the x_i values and Y is the maximum of the y_i values for all components in the frame. For each component, sampling factors H_i and V_i are defined relating component dimensions x_i and y_i to maximum dimensions X and Y , according to the following expressions:

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and } y_i = \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil,$$

where H_{max} and V_{max} are the maximum sampling factors for all components in the frame, and $\lceil \cdot \rceil$ is the ceiling function.

As an example, consider an image having 3 components with maximum dimensions of 512 lines and 512 samples per line, and with the following sampling factors:

Component 0	$H_0 = 4, V_0 = 1$
Component 1	$H_1 = 2, V_1 = 2$
Component 2	$H_2 = 1, V_2 = 1$

Then $X = 512$, $Y = 512$, $H_{max} = 4$, $V_{max} = 2$, and x_i and y_i for each component are

Component 0	$x_0 = 512, y_0 = 256$
Component 1	$x_1 = 256, y_1 = 512$
Component 2	$x_2 = 128, y_2 = 256$

NOTE – The X , Y , H_i , and V_i parameters are contained in the frame header of the compressed image data (see B.2.2), whereas the individual component dimensions x_i and y_i are derived by the decoder. Source images with x_i and y_i dimensions which do not satisfy the expressions above cannot be properly reconstructed.

A.1.2 Sample precision

A sample is an integer with precision P bits, with any value in the range 0 through $2^P - 1$. All samples of all components within an image shall have the same precision P . Restrictions on the value of P depend on the mode of operation, as specified in B.2 to B.7.

A.1.3 Data unit

A data unit is a sample in lossless processes and an 8×8 block of contiguous samples in DCT-based processes. The left-most 8 samples of each of the top-most 8 rows in the component shall always be the top-left-most block. With this top-left-most block as the reference, the component is partitioned into contiguous data units to the right and to the bottom (as shown in Figure A.4).

A.1.4 Orientation

Figure A.1 indicates the orientation of an image component by the terms top, bottom, left, and right. The order by which the data units of an image component are input to the compression encoding procedures is defined to be left-to-right and top-to-bottom within the component. (This ordering is precisely defined in A.2.) Applications determine which edges of a source image are defined as top, bottom, left, and right.

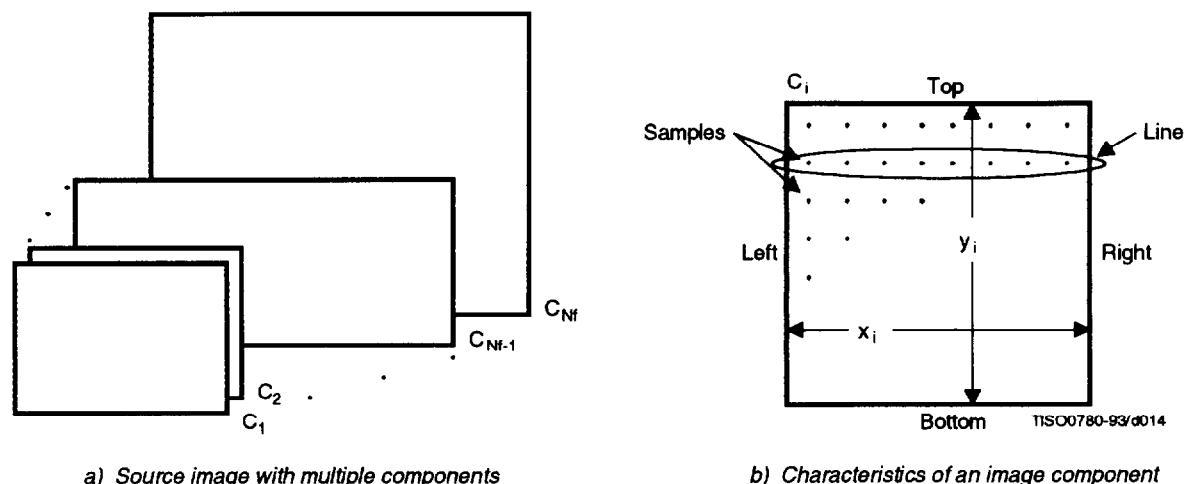


Figure A.1 – Source image characteristics

A.2 Order of source image data encoding

The scan header (see B.2.3) specifies the order by which source image data units shall be encoded and placed within the compressed image data. For a given scan, if the scan header parameter $N_s = 1$, then data from only one source component – the component specified by parameter C_{s1} – shall be present within the scan. This data is non-interleaved by definition. If $N_s > 1$, then data from the N_s components C_{s1} through C_{sN_s} shall be present within the scan. This data shall always be interleaved. The order of components in a scan shall be according to the order specified in the frame header.

The ordering of data units and the construction of minimum coded units (MCU) is defined as follows.

A.2.1 Minimum coded unit (MCU)

For non-interleaved data the MCU is one data unit. For interleaved data the MCU is the sequence of data units defined by the sampling factors of the components in the scan.

A.2.2 Non-interleaved order ($N_s = 1$)

When $N_s = 1$ (where N_s is the number of components in a scan), the order of data units within a scan shall be left-to-right and top-to-bottom, as shown in Figure A.2. This ordering applies whenever $N_s = 1$, regardless of the values of H_1 and V_1 .

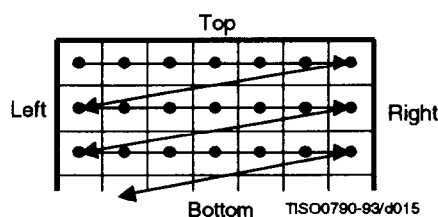


Figure A.2 – Non-interleaved data ordering

A.2.3 Interleaved order ($N_s > 1$)

When $N_s > 1$, each scan component Cs_i is partitioned into small rectangular arrays of H_k horizontal data units by V_k vertical data units. The subscripts k indicate that H_k and V_k are from the position in the frame header component-specification for which $C_k = Cs_i$. Within each H_k by V_k array, data units are ordered from left-to-right and top-to-bottom. The arrays in turn are ordered from left-to-right and top-to-bottom within each component.

As shown in the example of Figure A.3, $N_s = 4$, and MCU_1 consists of data units taken first from the top-left-most region of Cs_1 , followed by data units from the corresponding region of Cs_2 , then from Cs_3 and then from Cs_4 . MCU_2 follows the same ordering for data taken from the next region to the right for the four components.

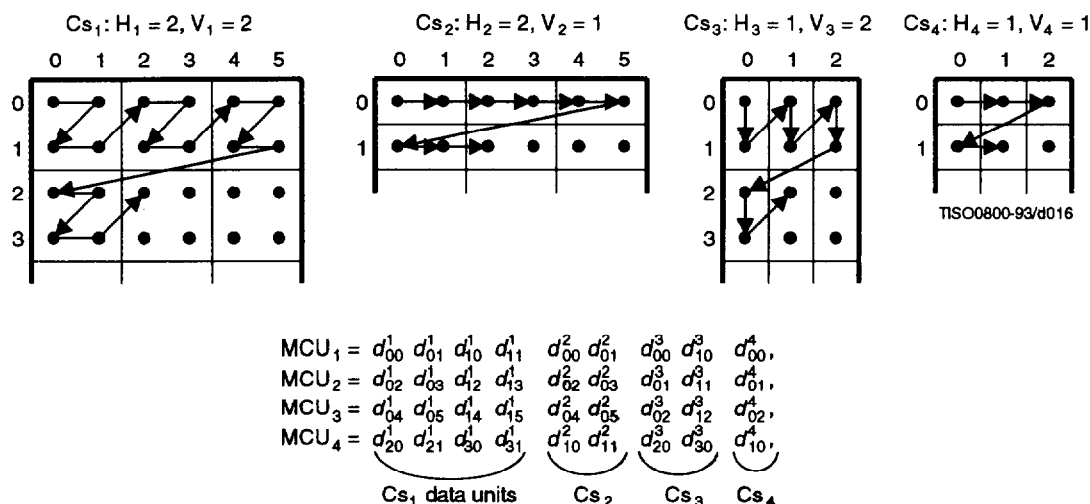


Figure A.3 – Interleaved data ordering example

A.2.4 Completion of partial MCU

For DCT-based processes the data unit is a block. If x_i is not a multiple of 8, the encoding process shall extend the number of columns to complete the right-most sample blocks. If the component is to be interleaved, the encoding process shall also extend the number of samples by one or more additional blocks, if necessary, so that the number of blocks is an integer multiple of H_i . Similarly, if y_i is not a multiple of 8, the encoding process shall extend the number of lines to complete the bottom-most block-row. If the component is to be interleaved, the encoding process shall also extend the number of lines by one or more additional block-rows, if necessary, so that the number of block-rows is an integer multiple of V_i .

NOTE – It is recommended that any incomplete MCUs be completed by replication of the right-most column and the bottom line of each component.

For lossless processes the data unit is a sample. If the component is to be interleaved, the encoding process shall extend the number of samples, if necessary, so that the number is a multiple of H_i . Similarly, the encoding process shall extend the number of lines, if necessary, so that the number of lines is a multiple of V_i .

Any sample added by an encoding process to complete partial MCUs shall be removed by the decoding process.

A.3 DCT compression

A.3.1 Level shift

Before a non-differential frame encoding process computes the FDCT for a block of source image samples, the samples shall be level shifted to a signed representation by subtracting 2^{P-1} , where P is the precision parameter specified in B.2.2. Thus, when $P = 8$, the level shift is by 128; when $P = 12$, the level shift is by 2048.

After a non-differential frame decoding process computes the IDCT and produces a block of reconstructed image samples, an inverse level shift shall restore the samples to the unsigned representation by adding 2^{P-1} and clamping the results to the range 0 to 2^P-1 .

A.3.2 Orientation of samples for FDCT computation

Figure A.4 shows an image component which has been partitioned into 8×8 blocks for the FDCT computations. Figure A.4 also defines the orientation of the samples within a block by showing the indices used in the FDCT equation of A.3.3.

The definitions of block partitioning and sample orientation also apply to any DCT decoding process and the output reconstructed image. Any sample added by an encoding process to complete partial MCUs shall be removed by the decoding process.

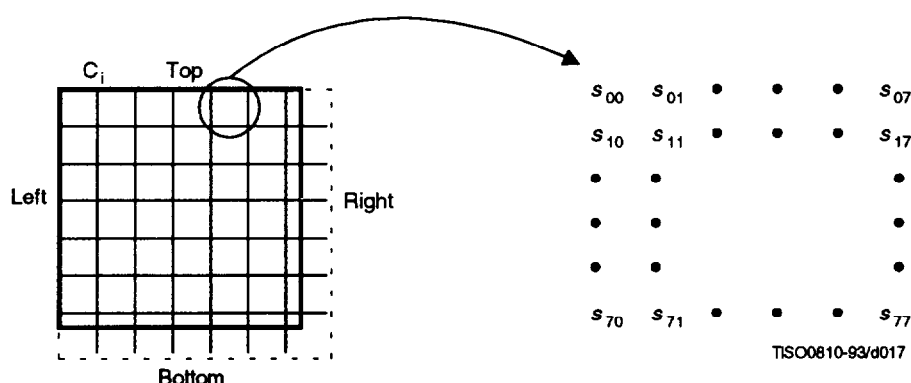


Figure A.4 – Partition and orientation of 8×8 sample blocks

A.3.3 FDCT and IDCT (informative)

The following equations specify the ideal functional definition of the FDCT and the IDCT.

NOTE – These equations contain terms which cannot be represented with perfect accuracy by any real implementation. The accuracy requirements for the combined FDCT and quantization procedures are specified in Part 2 of this Specification. The accuracy requirements for the combined dequantization and IDCT procedures are also specified in Part 2 of this Specification.

$$\text{FDCT:} \quad S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^7 \sum_{y=0}^7 s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

$$\text{IDCT:} \quad s_{yx} = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C_u C_v S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

where

$$C_u, C_v = 1/\sqrt{2} \text{ for } u, v = 0$$

$$C_u, C_v = 1 \text{ otherwise}$$

otherwise.

A.3.4 DCT coefficient quantization (informative) and dequantization (normative)

After the FDCT is computed for a block, each of the 64 resulting DCT coefficients is quantized by a uniform quantizer. The quantizer step size for each coefficient S_{vu} is the value of the corresponding element Q_{vu} from the quantization table specified by the frame parameter Tq_i (see B.2.2).

The uniform quantizer is defined by the following equation. Rounding is to the nearest integer:

$$Sq_{vu} = \text{round} \left(\frac{S_{vu}}{Q_{vu}} \right)$$

Sq_{vu} is the quantized DCT coefficient, normalized by the quantizer step size.

NOTE – This equation contains a term which may not be represented with perfect accuracy by any real implementation. The accuracy requirements for the combined FDCT and quantization procedures are specified in Part 2 of this Specification.

At the decoder, this normalization is removed by the following equation, which defines dequantization:

$$R_{vu} = Sq_{vu} \times Q_{vu}$$

NOTE – Depending on the rounding used in quantization, it is possible that the dequantized coefficient may be outside the expected range.

The relationship among samples, DCT coefficients, and quantization is illustrated in Figure A.5.

A.3.5 Differential DC encoding

After quantization, and in preparation for entropy encoding, the quantized DC coefficient Sq_{00} is treated separately from the 63 quantized AC coefficients. The value that shall be encoded is the difference (DIFF) between the quantized DC coefficient of the current block (DC_i which is also designated as Sq_{00}) and that of the previous block of the same component (PRED):

$$DIFF = DC_i - PRED$$

A.3.6 Zig-zag sequence

After quantization, and in preparation for entropy encoding, the quantized AC coefficients are converted to the zig-zag sequence. The quantized DC coefficient (coefficient zero in the array) is treated separately, as defined in A.3.5. The zig-zag sequence is specified in Figure A.6.

A.4 Point transform

For various procedures data may be optionally divided by a power of 2 by a point transform prior to coding. There are three processes which require a point transform: lossless coding, lossless differential frame coding in the hierarchical mode, and successive approximation coding in the progressive DCT mode.

In the lossless mode of operation the point transform is applied to the input samples. In the difference coding of the hierarchical mode of operation the point transform is applied to the difference between the input component samples and the reference component samples. In both cases the point transform is an integer divide by 2^{Pt} , where Pt is the value of the point transform parameter (see B.2.3).

In successive approximation coding the point transform for the AC coefficients is an integer divide by 2^{Al} , where Al is the successive approximation bit position, low (see B.2.3). The point transform for the DC coefficients is an arithmetic-shift-right by Al bits. This is equivalent to dividing by 2^{Pt} before the level shift (see A.3.1).

The output of the decoder is rescaled by multiplying by 2^{Pt} . An example of the point transform is given in K.10.

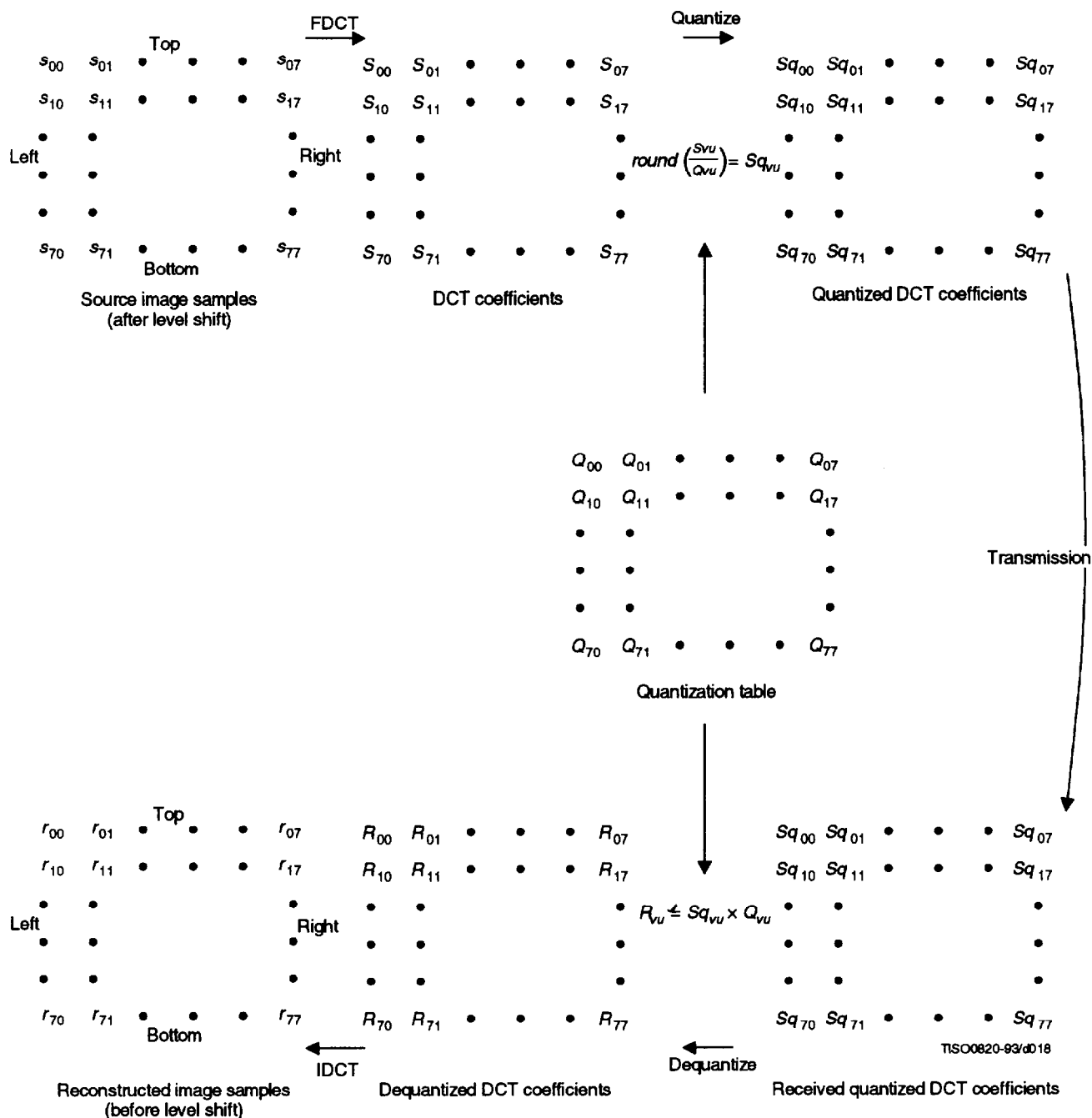


Figure A.5 – Relationship between 8 × 8-block samples and DCT coefficients

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

Figure A.6 – Zig-zag sequence of quantized DCT coefficients

A.5 Arithmetic procedures in lossless and hierarchical modes of operation

In the lossless mode of operation predictions are calculated with full precision and without clamping of either overflow or underflow beyond the range of values allowed by the precision of the input. However, the division by two which is part of some of the prediction calculations shall be approximated by an arithmetic-shift-right by one bit.

The two's complement differences which are coded in either the lossless mode of operation or the differential frame coding in the hierarchical mode of operation are calculated modulo 65 536, thereby restricting the precision of these differences to a maximum of 16 bits. The modulo values are calculated by performing the logical AND operation of the two's complement difference with X'FFFF'. For purposes of coding, the result is still interpreted as a 16 bit two's complement difference. Modulo 65 536 arithmetic is also used in the decoder in calculating the output from the sum of the prediction and this two's complement difference.

Annex B

Compressed data formats

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

This annex specifies three compressed data formats:

- a) the interchange format, specified in B.2 and B.3;
- b) the abbreviated format for compressed image data, specified in B.4;
- c) the abbreviated format for table-specification data, specified in B.5.

B.1 describes the constituent parts of these formats. B.1.3 and B.1.4 give the conventions for symbols and figures used in the format specifications.

B.1 General aspects of the compressed data format specifications

Structurally, the compressed data formats consist of an ordered collection of parameters, markers, and entropy-coded data segments. Parameters and markers in turn are often organized into marker segments. Because all of these constituent parts are represented with byte-aligned codes, each compressed data format consists of an ordered sequence of 8-bit bytes. For each byte, a most significant bit (MSB) and a least significant bit (LSB) are defined.

B.1.1 Constituent parts

This subclause gives a general description of each of the constituent parts of the compressed data format.

B.1.1.1 Parameters

Parameters are integers, with values specific to the encoding process, source image characteristics, and other features selectable by the application. Parameters are assigned either 4-bit, 1-byte, or 2-byte codes. Except for certain optional groups of parameters, parameters encode critical information without which the decoding process cannot properly reconstruct the image.

The code assignment for a parameter shall be an unsigned integer of the specified length in bits with the particular value of the parameter.

For parameters which are 2 bytes (16 bits) in length, the most significant byte shall come first in the compressed data's ordered sequence of bytes. Parameters which are 4 bits in length always come in pairs, and the pair shall always be encoded in a single byte. The first 4-bit parameter of the pair shall occupy the most significant 4 bits of the byte. Within any 16-, 8-, or 4-bit parameter, the MSB shall come first and LSB shall come last.

B.1.1.2 Markers

Markers serve to identify the various structural parts of the compressed data formats. Most markers start marker segments containing a related group of parameters; some markers stand alone. All markers are assigned two-byte codes: an X'FF' byte followed by a byte which is not equal to 0 or X'FF' (see Table B.1). Any marker may optionally be preceded by any number of fill bytes, which are bytes assigned code X'FF'.

NOTE – Because of this special code-assignment structure, markers make it possible for a decoder to parse the compressed data and locate its various parts without having to decode other segments of image data.

B.1.1.3 Marker assignments

All markers shall be assigned two-byte codes: a X'FF' byte followed by a second byte which is not equal to 0 or X'FF'. The second byte is specified in Table B.1 for each defined marker. An asterisk (*) indicates a marker which stands alone, that is, which is not the start of a marker segment.

Table B.1 – Marker code assignments

Code Assignment	Symbol	Description
Start Of Frame markers, non-differential, Huffman coding		
X'FFC0' X'FFC1' X'FFC2' X'FFC3'	SOF ₀ SOF ₁ SOF ₂ SOF ₃	Baseline DCT Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, Huffman coding		
X'FFC5' X'FFC6' X'FFC7'	SOF ₅ SOF ₆ SOF ₇	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Start Of Frame markers, non-differential, arithmetic coding		
X'FFC8' X'FFC9' X'FFCA' X'FFCB'	JPG SOF ₉ SOF ₁₀ SOF ₁₁	Reserved for JPEG extensions Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, arithmetic coding		
X'FFCD' X'FFCE' X'FFCF'	SOF ₁₃ SOF ₁₄ SOF ₁₅	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Huffman table specification		
X'FFC4'	DHT	Define Huffman table(s)
Arithmetic coding conditioning specification		
X'FFCC'	DAC	Define arithmetic coding conditioning(s)
Restart interval termination		
X'FFD0' through X'FFD7'	RST _m *	Restart with modulo 8 count "m"
Other markers		
X'FFD8' X'FFD9' X'FFDA' X'FFDB' X'FFDC' X'FFDD' X'FFDE' X'FFDF' X'FFE0' through X'FFEF' X'FFF0' through X'FFFD' X'FFFE'	SOI* EOI* SOS DQT DNL DRI DHP EXP APP _n JPG _n COM	Start of image End of image Start of scan Define quantization table(s) Define number of lines Define restart interval Define hierarchical progression Expand reference component(s) Reserved for application segments Reserved for JPEG extensions Comment
Reserved markers		
X'FF01' X'FF02' through X'FFBF'	TEM* RES	For temporary private use in arithmetic coding Reserved

B.1.1.4 Marker segments

A marker segment consists of a marker followed by a sequence of related parameters. The first parameter in a marker segment is the two-byte length parameter. This length parameter encodes the number of bytes in the marker segment, including the length parameter and excluding the two-byte marker. The marker segments identified by the SOF and SOS marker codes are referred to as headers: the frame header and the scan header respectively.

B.1.1.5 Entropy-coded data segments

An entropy-coded data segment contains the output of an entropy-coding procedure. It consists of an integer number of bytes, whether the entropy-coding procedure used is Huffman or arithmetic.

NOTES

1 Making entropy-coded segments an integer number of bytes is performed as follows: for Huffman coding, 1-bits are used, if necessary, to pad the end of the compressed data to complete the final byte of a segment. For arithmetic coding, byte alignment is performed in the procedure which terminates the entropy-coded segment (see D.1.8).

2 In order to ensure that a marker does not occur within an entropy-coded segment, any X'FF' byte generated by either a Huffman or arithmetic encoder, or an X'FF' byte that was generated by the padding of 1-bits described in NOTE 1 above, is followed by a "stuffed" zero byte (see D.1.6 and F.1.2.3).

B.1.2 Syntax

In B.2 and B.3 the interchange format syntax is specified. For the purposes of this Specification, the syntax specification consists of:

- the required ordering of markers, parameters, and entropy-coded segments;
- identification of optional or conditional constituent parts;
- the name, symbol, and definition of each marker and parameter;
- the allowed values of each parameter;
- any restrictions on the above which are specific to the various coding processes.

The ordering of constituent parts and the identification of which are optional or conditional is specified by the syntax figures in B.2 and B.3. Names, symbols, definitions, allowed values, conditions, and restrictions are specified immediately below each syntax figure.

B.1.3 Conventions for syntax figures

The syntax figures in B.2 and B.3 are a part of the interchange format specification. The following conventions, illustrated in Figure B.1, apply to these figures:

- **parameter/marker indicator:** A thin-lined box encloses either a marker or a single parameter;
- **segment indicator:** A thick-lined box encloses either a marker segment, an entropy-coded data segment, or combinations of these;
- **parameter length indicator:** The width of a thin-lined box is proportional to the parameter length (4, 8, or 16 bits, shown as E, B, and D respectively in Figure B.1) of the marker or parameter it encloses; the width of thick-lined boxes is not meaningful;
- **optional/conditional indicator:** Square brackets indicate that a marker or marker segment is only optionally or conditionally present in the compressed image data;
- **ordering:** In the interchange format a parameter or marker shown in a figure precedes all of those shown to its right, and follows all of those shown to its left;
- **entropy-coded data indicator:** Angled brackets indicate that the entity enclosed has been entropy encoded.

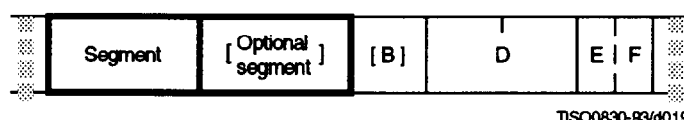


Figure B.1 – Syntax notation conventions

B.1.4 Conventions for symbols, code lengths, and values

Following each syntax figure in B.2 and B.3, the symbol, name, and definition for each marker and parameter shown in the figure are specified. For each parameter, the length and allowed values are also specified in tabular form.

The following conventions apply to symbols for markers and parameters:

- all marker symbols have three upper-case letters, and some also have a subscript. Examples: SOI, SOF_n;
- all parameter symbols have one upper-case letter; some also have one lower-case letter and some have subscripts. Examples: Y, Nf, H_j, Tq_i.

B.2 General sequential and progressive syntax

This clause specifies the interchange format syntax which applies to all coding processes for sequential DCT-based, progressive DCT-based, and lossless modes of operation.

B.2.1 High-level syntax

Figure B.2 specifies the order of the high-level constituent parts of the interchange format for all non-hierarchical encoding processes specified in this Specification.

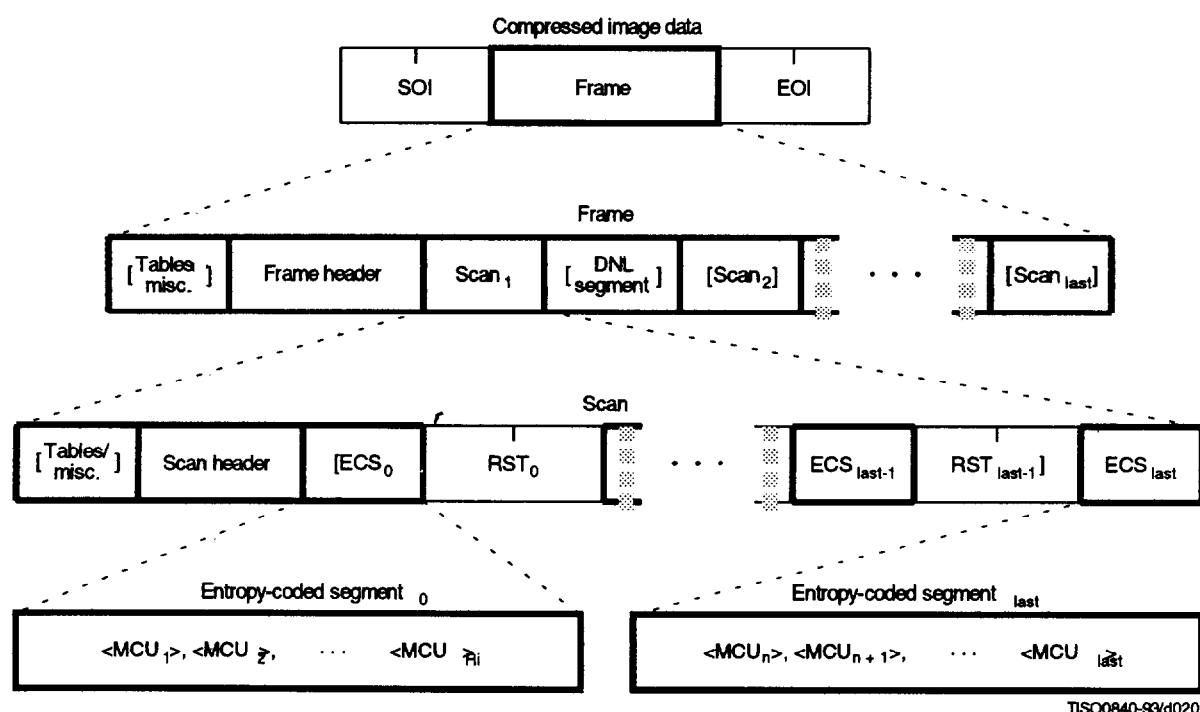


Figure B.2 – Syntax for sequential DCT-based, progressive DCT-based, and lossless modes of operation

The three markers shown in Figure B.2 are defined as follows:

SOI: Start of image marker – Marks the start of a compressed image represented in the interchange format or abbreviated format.

EOI: End of image marker – Marks the end of a compressed image represented in the interchange format or abbreviated format.

RST_m: Restart marker – A conditional marker which is placed between entropy-coded segments only if restart is enabled. There are 8 unique restart markers ($m = 0 - 7$) which repeat in sequence from 0 to 7, starting with zero for each scan, to provide a modulo 8 restart interval count.

The top level of Figure B.2 specifies that the non-hierarchical interchange format shall begin with an SOI marker, shall contain one frame, and shall end with an EOI marker.

The second level of Figure B.2 specifies that a frame shall begin with a frame header and shall contain one or more scans. A frame header may be preceded by one or more table-specification or miscellaneous marker segments as specified in B.2.4. If a DNL segment (see B.2.5) is present, it shall immediately follow the first scan.

For sequential DCT-based and lossless processes each scan shall contain from one to four image components. If two to four components are contained within a scan, they shall be interleaved within the scan. For progressive DCT-based processes each image component is only partially contained within any one scan. Only the first scan(s) for the components (which contain only DC coefficient data) may be interleaved.

The third level of Figure B.2 specifies that a scan shall begin with a scan header and shall contain one or more entropy-coded data segments. Each scan header may be preceded by one or more table-specification or miscellaneous marker segments. If restart is not enabled, there shall be only one entropy-coded segment (the one labeled "last"), and no restart markers shall be present. If restart is enabled, the number of entropy-coded segments is defined by the size of the image and the defined restart interval. In this case, a restart marker shall follow each entropy-coded segment except the last one.

The fourth level of Figure B.2 specifies that each entropy-coded segment is comprised of a sequence of entropy-coded MCUs. If restart is enabled and the restart interval is defined to be R_i , each entropy-coded segment except the last one shall contain R_i MCUs. The last one shall contain whatever number of MCUs completes the scan.

Figure B.2 specifies the locations where table-specification segments **may** be present. However, this Specification hereby specifies that the interchange format **shall** contain all table-specification data necessary for decoding the compressed image. Consequently, the required table-specification data **shall** be present at one or more of the allowed locations.

B.2.2 Frame header syntax

Figure B.3 specifies the frame header which shall be present at the start of a frame. This header specifies the source image characteristics (see A.1), the components in the frame, and the sampling factors for each component, and specifies the destinations from which the quantized tables to be used with each component are retrieved.

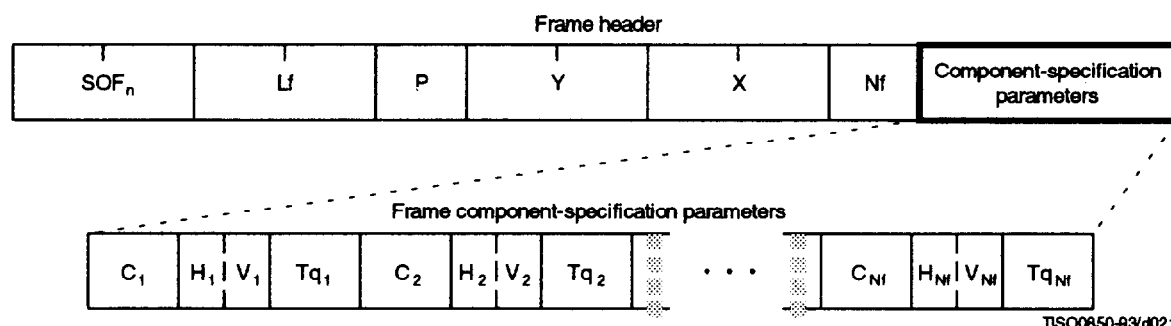


Figure B.3 – Frame header syntax

The markers and parameters shown in Figure B.3 are defined below. The size and allowed values of each parameter are given in Table B.2. In Table B.2 (and similar tables which follow), value choices are separated by commas (e.g. 8, 12) and inclusive bounds are separated by dashes (e.g. 0 - 3).

SOF_n : Start of frame marker – Marks the beginning of the frame parameters. The subscript n identifies whether the encoding process is baseline sequential, extended sequential, progressive, or lossless, as well as which entropy encoding procedure is used.

SOF_0 : Baseline DCT

SOF_1 : Extended sequential DCT, Huffman coding

SOF_2 : Progressive DCT, Huffman coding

SOF₃: Lossless (sequential), Huffman coding

SOF₉: Extended sequential DCT, arithmetic coding

SOF₁₀: Progressive DCT, arithmetic coding

SOF₁₁: Lossless (sequential), arithmetic coding

Lf: Frame header length – Specifies the length of the frame header shown in Figure B.3 (see B.1.1.4).

P: Sample precision – Specifies the precision in bits for the samples of the components in the frame.

Y: Number of lines – Specifies the maximum number of lines in the source image. This shall be equal to the number of lines in the component with the maximum number of vertical samples (see A.1.1). Value 0 indicates that the number of lines shall be defined by the DNL marker and parameters at the end of the first scan (see B.2.5).

X: Number of samples per line – Specifies the maximum number of samples per line in the source image. This shall be equal to the number of samples per line in the component with the maximum number of horizontal samples (see A.1.1).

Nf: Number of image components in frame – Specifies the number of source image components in the frame. The value of Nf shall be equal to the number of sets of frame component specification parameters (C_i , H_i , V_i , and Tq_i) present in the frame header.

C_i : Component identifier – Assigns a unique label to the i th component in the sequence of frame component specification parameters. These values shall be used in the scan headers to identify the components in the scan. The value of C_i shall be different from the values of C_1 through C_{i-1} .

H_i : Horizontal sampling factor – Specifies the relationship between the component horizontal dimension and maximum image dimension X (see A.1.1); also specifies the number of horizontal data units of component C_i in each MCU, when more than one component is encoded in a scan.

V_i : Vertical sampling factor – Specifies the relationship between the component vertical dimension and maximum image dimension Y (see A.1.1); also specifies the number of vertical data units of component C_i in each MCU, when more than one component is encoded in a scan.

Tq_i : Quantization table destination selector – Specifies one of four possible quantization table destinations from which the quantization table to use for dequantization of DCT coefficients of component C_i is retrieved. If the decoding process uses the dequantization procedure, this table shall have been installed in this destination by the time the decoder is ready to decode the scan(s) containing component C_i . The destination shall not be re-specified, or its contents changed, until all scans containing C_i have been completed.

Table B.2 – Frame header parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lf	16	8 + 3 × Nf			
P	8	8	8, 12	8, 12	2-16
Y	16	0-65 535			
X	16	1-65 535			
Nf	8	1-255	1-255	1-4	1-255
C _i	8	0-255			
H _i	4	1-4			
V _i	4	1-4			
Tq _i	8	0-3	0-3	0-3	0

B.2.3 Scan header syntax

Figure B.4 specifies the scan header which shall be present at the start of a scan. This header specifies which component(s) are contained in the scan, specifies the destinations from which the entropy tables to be used with each component are retrieved, and (for the progressive DCT) which part of the DCT quantized coefficient data is contained in the scan. For lossless processes the scan parameters specify the predictor and the point transform.

NOTE – If there is only one image component present in a scan, that component is, by definition, non-interleaved. If there is more than one image component present in a scan, the components present are, by definition, interleaved.

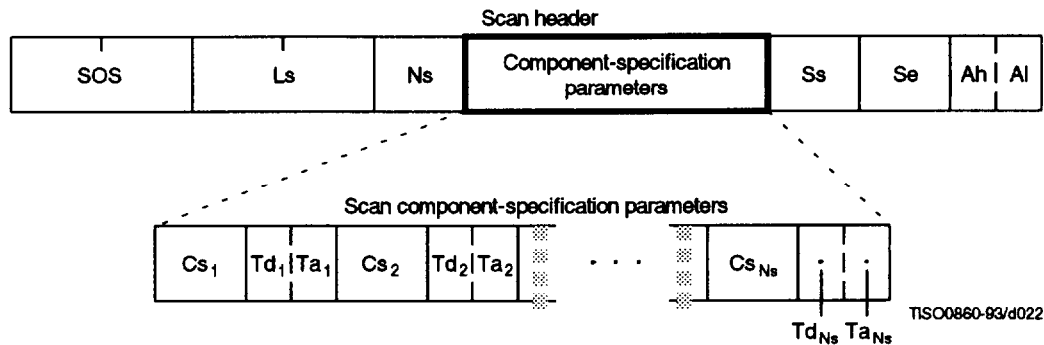


Figure B.4 – Scan header syntax

The marker and parameters shown in Figure B.4 are defined below. The size and allowed values of each parameter are given in Table B.3.

SOS: Start of scan marker – Marks the beginning of the scan parameters.

Ls: Scan header length – Specifies the length of the scan header shown in Figure B.4 (see B.1.1.4).

Ns: Number of image components in scan – Specifies the number of source image components in the scan. The value of Ns shall be equal to the number of sets of scan component specification parameters (Cs_j, Td_j, and Ta_j) present in the scan header.

Cs_j: Scan component selector – Selects which of the N_f image components specified in the frame parameters shall be the jth component in the scan. Each Cs_j shall match one of the C_i values specified in the frame header, and the ordering in the scan header shall follow the ordering in the frame header. If Ns > 1, the order of interleaved components in the MCU is Cs₁ first, Cs₂ second, etc. If Ns > 1, the following restriction shall be placed on the image components contained in the scan:

$$\sum_{j=1}^{N_s} H_j \times V_j \leq 10,$$

where H_j and V_j are the horizontal and vertical sampling factors for scan component j. These sampling factors are specified in the frame header for component i, where i is the frame component specification index for which frame component identifier C_i matches scan component selector Cs_j.

As an example, consider an image having 3 components with maximum dimensions of 512 lines and 512 samples per line, and with the following sampling factors:

Component 0	$H_0 = 4,$	$V_0 = 1$
Component 1	$H_1 = 1,$	$V_1 = 2$
Component 2	$H_2 = 2$	$V_2 = 2$

Then the summation of H_j × V_j is (4 × 1) + (1 × 2) + (2 × 2) = 10.

The value of Cs_j shall be different from the values of Cs₁ to Cs_{j-1}.

Td_j: DC entropy coding table destination selector – Specifies one of four possible DC entropy coding table destinations from which the entropy table needed for decoding of the DC coefficients of component C_{s_j} is retrieved. The DC entropy table shall have been installed in this destination (see B.2.4.2 and B.2.4.3) by the time the decoder is ready to decode the current scan. This parameter specifies the entropy coding table destination for the lossless processes.

Ta_j: AC entropy coding table destination selector – Specifies one of four possible AC entropy coding table destinations from which the entropy table needed for decoding of the AC coefficients of component C_{s_j} is retrieved. The AC entropy table selected shall have been installed in this destination (see B.2.4.2 and B.2.4.3) by the time the decoder is ready to decode the current scan. This parameter is zero for the lossless processes.

Ss: Start of spectral or predictor selection – In the DCT modes of operation, this parameter specifies the first DCT coefficient in each block in zig-zag order which shall be coded in the scan. This parameter shall be set to zero for the sequential DCT processes. In the lossless mode of operations this parameter is used to select the predictor.

Se: End of spectral selection – Specifies the last DCT coefficient in each block in zig-zag order which shall be coded in the scan. This parameter shall be set to 63 for the sequential DCT processes. In the lossless mode of operations this parameter has no meaning. It shall be set to zero.

Ah: Successive approximation bit position high – This parameter specifies the point transform used in the preceding scan (i.e. successive approximation bit position low in the preceding scan) for the band of coefficients specified by Ss and Se. This parameter shall be set to zero for the first scan of each band of coefficients. In the lossless mode of operations this parameter has no meaning. It shall be set to zero.

Al: Successive approximation bit position low or point transform – In the DCT modes of operation this parameter specifies the point transform, i.e. bit position low, used before coding the band of coefficients specified by Ss and Se. This parameter shall be set to zero for the sequential DCT processes. In the lossless mode of operations, this parameter specifies the point transform, Pt.

The entropy coding table destination selectors, Td_j and Ta_j, specify either Huffman tables (in frames using Huffman coding) or arithmetic coding tables (in frames using arithmetic coding). In the latter case the entropy coding table destination selector specifies both an arithmetic coding conditioning table destination and an associated statistics area.

Table B.3 – Scan header parameter size and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Ls	16	6 + 2 × Ns			
Ns	8	1-4			
Csj	8	0-255a)			
Tdj	4	0-1	0-3	0-3	0-3
Taj	4	0-1	0-3	0-3	0
Ss	8	0	0	0-63	1-7b)
Se	8	63	63	Ss-63c)	0
Ah	4	0	0	0-13	0
Al	4	0	0	0-13	0-15
a) Csj shall be a member of the set of Ci specified in the frame header.					
b) 0 for lossless differential frames in the hierarchical mode (see B.3).					
c) 0 if Ss equals zero.					

B.2.4 Table-specification and miscellaneous marker segment syntax

Figure B.5 specifies that, at the places indicated in Figure B.2, any of the table-specification segments or miscellaneous marker segments specified in B.2.4.1 through B.2.4.6 may be present in any order and with no limit on the number of segments.

If any table specification for a particular destination occurs in the compressed image data, it shall replace any previous table specified for this destination, and shall be used whenever this destination is specified in the remaining scans in the frame or subsequent images represented in the abbreviated format for compressed image data. If a table specification for a given destination occurs more than once in the compressed image data, each specification shall replace the previous specification. The quantization table specification shall not be altered between progressive DCT scans of a given component.

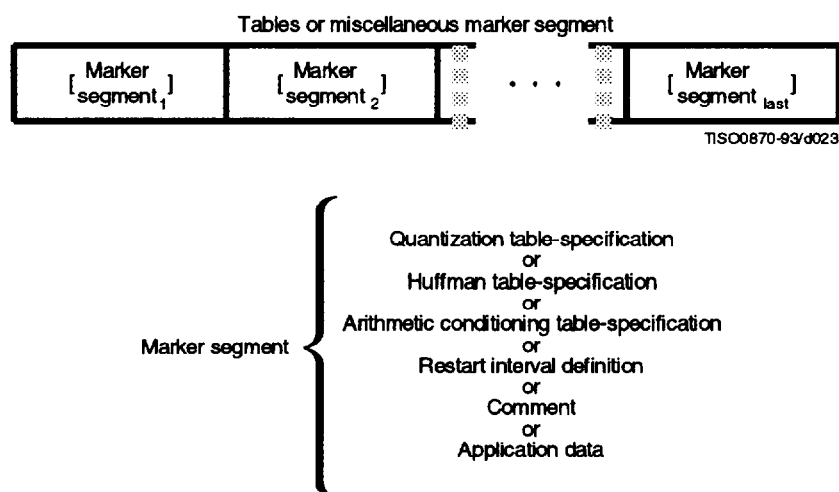


Figure B.5 – Tables/miscellaneous marker segment syntax

B.2.4.1 Quantization table-specification syntax

Figure B.6 specifies the marker segment which defines one or more quantization tables.

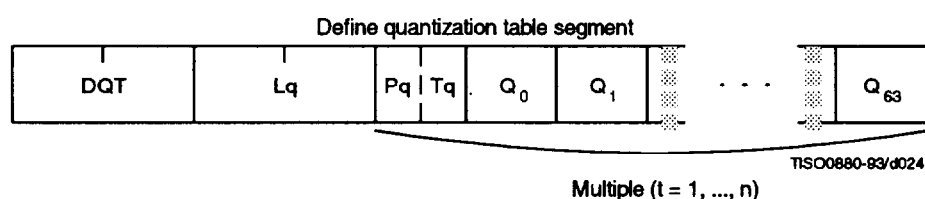


Figure B.6 – Quantization table syntax

The marker and parameters shown in Figure B.6 are defined below. The size and allowed values of each parameter are given in Table B.4.

DQT: Define quantization table marker – Marks the beginning of quantization table-specification parameters.

Lq: Quantization table definition length – Specifies the length of all quantization table parameters shown in Figure B.6 (see B.1.1.4).

P_q: Quantization table element precision – Specifies the precision of the Q_k values. Value 0 indicates 8-bit Q_k values; value 1 indicates 16-bit Q_k values. P_q shall be zero for 8 bit sample precision P (see B.2.2).

T_q: Quantization table destination identifier – Specifies one of four possible destinations at the decoder into which the quantization table shall be installed.

Q_k: Quantization table element – Specifies the *k*th element out of 64 elements, where *k* is the index in the zig-zag ordering of the DCT coefficients. The quantization elements shall be specified in zig-zag scan order.

Table B.4 – Quantization table-specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lq	16	$2 + \sum_{t=1}^n (65 + 64 \times Pq(t))$			Undefined
Pq	4	0	0, 1	0, 1	Undefined
Tq	4	0-3			Undefined
Qk	8, 16	1-255, 1-65 535			Undefined

The value *n* in Table B.4 is the number of quantization tables specified in the DQT marker segment.

Once a quantization table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used, when referenced, in the remaining scans of the current image and in subsequent images represented in the abbreviated format for compressed image data. If a table has never been defined for a particular destination, then when this destination is specified in a frame header, the results are unpredictable.

An 8-bit DCT-based process shall not use a 16-bit precision quantization table.

B.2.4.2 Huffman table-specification syntax

Figure B.7 specifies the marker segment which defines one or more Huffman table specifications.

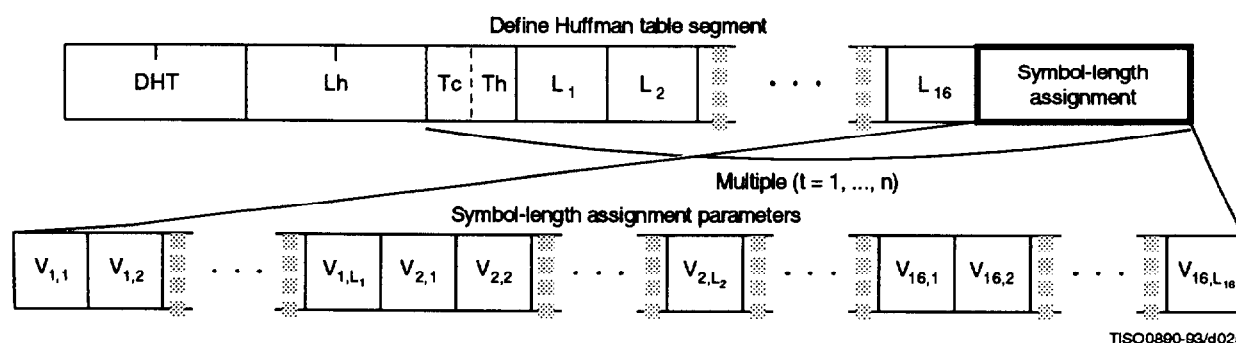


Figure B.7 – Huffman table syntax

The marker and parameters shown in Figure B.7 are defined below. The size and allowed values of each parameter are given in Table B.5.

DHT: Define Huffman table marker – Marks the beginning of Huffman table definition parameters.

Lh: Huffman table definition length – Specifies the length of all Huffman table parameters shown in Figure B.7 (see B.1.1.4).

Tc: Table class – 0 = DC table or lossless table, 1 = AC table.

Th: Huffman table destination identifier – Specifies one of four possible destinations at the decoder into which the Huffman table shall be installed.

L_i: Number of Huffman codes of length *i* – Specifies the number of Huffman codes for each of the 16 possible lengths allowed by this Specification. *L_i*'s are the elements of the list BITS.

V_{i,j}: Value associated with each Huffman code – Specifies, for each *i*, the value associated with each Huffman code of length *i*. The meaning of each value is determined by the Huffman coding model. The *V_{i,j}*'s are the elements of the list HUFFVAL.

Table B.5 – Huffman table specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lh	16	$2 + \sum_{t=1}^n (17 + m_t)$			
Tc	4	0, 1			0
Th	4	0, 1	0-3		
Li	8	0-255			
V _{i,j}	8	0-255			

The value *n* in Table B.5 is the number of Huffman tables specified in the DHT marker segment. The value *m_t* is the number of parameters which follow the 16 *L_i(t)* parameters for Huffman table *t*, and is given by:

$$m_t = \sum_{i=1}^{16} L_i$$

In general, *m_t* is different for each table.

Once a Huffman table has been defined for a particular destination, it replaces the previous tables stored in that destination and shall be used when referenced, in the remaining scans of the current image and in subsequent images represented in the abbreviated format for compressed image data. If a table has never been defined for a particular destination, then when this destination is specified in a scan header, the results are unpredictable.

B.2.4.3 Arithmetic conditioning table-specification syntax

Figure B.8 specifies the marker segment which defines one or more arithmetic coding conditioning table specifications. These replace the default arithmetic coding conditioning tables established by the SOI marker for arithmetic coding processes. (See F.1.4.4.1.4 and F.1.4.4.2.1.)

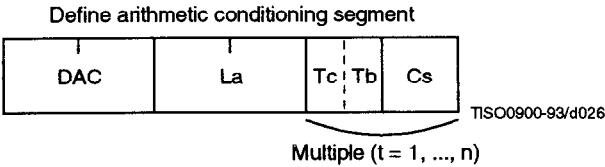


Figure B.8 – Arithmetic conditioning table-specification syntax

The marker and parameters shown in Figure B.8 are defined below. The size and allowed values of each parameter are given in Table B.6.

DAC: Define arithmetic coding conditioning marker – Marks the beginning of the definition of arithmetic coding conditioning parameters.

La: Arithmetic coding conditioning definition length – Specifies the length of all arithmetic coding conditioning parameters shown in Figure B.8 (see B.1.1.4).

Tc: Table class – 0 = DC table or lossless table, 1 = AC table.

Tb: Arithmetic coding conditioning table destination identifier – Specifies one of four possible destinations at the decoder into which the arithmetic coding conditioning table shall be installed.

Cs: Conditioning table value – Value in either the AC or the DC (and lossless) conditioning table. A single value of Cs shall follow each value of Tb. For AC conditioning tables Tc shall be one and Cs shall contain a value of Kx in the range $1 \leq Kx \leq 63$. For DC (and lossless) conditioning tables Tc shall be zero and Cs shall contain two 4-bit parameters, U and L. U and L shall be in the range $0 \leq L \leq U \leq 15$ and the value of Cs shall be $L + 16 \times U$.

The value n in Table B.6 is the number of arithmetic coding conditioning tables specified in the DAC marker segment. The parameters L and U are the lower and upper conditioning bounds used in the arithmetic coding procedures defined for DC coefficient coding and lossless coding. The separate value range 1-63 listed for DCT coding is the Kx conditioning used in AC coefficient coding.

Table B.6 – Arithmetic coding conditioning table-specification parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
La	16	Undefined	$2 + 2 \times n$		
Tc	4	Undefined	0, 1		0
Tb	4	Undefined	0-3		
Cs	8	Undefined	0-255 (Tc = 0), 1-63 (Tc = 1)		0-255

B.2.4.4 Restart interval definition syntax

Figure B.9 specifies the marker segment which defines the restart interval.

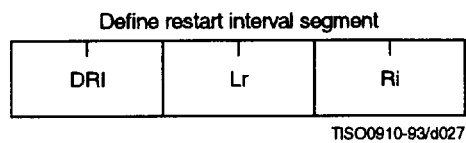


Figure B.9 – Restart interval definition syntax

The marker and parameters shown in Figure B.9 are defined below. The size and allowed values of each parameter are given in Table B.7.

DRI: Define restart interval marker – Marks the beginning of the parameters which define the restart interval.

Lr: Define restart interval segment length – Specifies the length of the parameters in the DRI segment shown in Figure B.9 (see B.1.1.4).

Ri: Restart interval – Specifies the number of MCU in the restart interval.

In Table B.7 the value n is the number of rows of MCU in the restart interval. The value MCUR is the number of MCU required to make up one line of samples of each component in the scan. The SOI marker disables the restart intervals. A DRI marker segment with Ri nonzero shall be present to enable restart interval processing for the following scans. A DRI marker segment with Ri equal to zero shall disable restart intervals for the following scans.

Table B.7 – Define restart interval segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lr	16	4			
Ri	16	0-65 535			n × MCUR

B.2.4.5 Comment syntax

Figure B.10 specifies the marker segment structure for a comment segment.

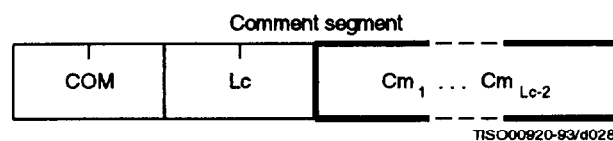


Figure B.10 – Comment segment syntax

The marker and parameters shown in Figure B.10 are defined below. The size and allowed values of each parameter are given in Table B.8.

COM: Comment marker – Marks the beginning of a comment.

Lc: Comment segment length – Specifies the length of the comment segment shown in Figure B.10 (see B.1.1.4).

Cm_i: Comment byte – The interpretation is left to the application.

Table B.8 – Comment segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lc	16	2-65 535			
Cm _i	8	0-255			

B.2.4.6 Application data syntax

Figure B.11 specifies the marker segment structure for an application data segment.

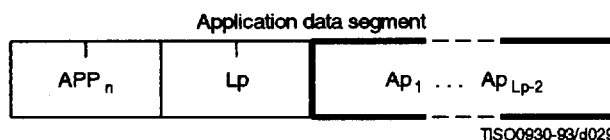


Figure B.11 – Application data syntax

The marker and parameters shown in Figure B.11 are defined below. The size and allowed values of each parameter are given in Table B.9.

APP_n: Application data marker – Marks the beginning of an application data segment.

Lp: Application data segment length – Specifies the length of the application data segment shown in Figure B.11 (see B.1.1.4).

Ap_i: Application data byte – The interpretation is left to the application.

The APP_n (Application) segments are reserved for application use. Since these segments may be defined differently for different applications, they should be removed when the data are exchanged between application environments.

Table B.9 – Application data segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Lp	16	2-65 535			
Ap _i	8	0-255			

B.2.5 Define number of lines syntax

Figure B.12 specifies the marker segment for defining the number of lines. The DNL (Define Number of Lines) segment provides a mechanism for defining or redefining the number of lines in the frame (the Y parameter in the frame header) at the end of the first scan. The value specified shall be consistent with the number of MCU-rows encoded in the first scan. This segment, if used, shall only occur at the end of the first scan, and only after coding of an integer number of MCU-rows. This marker segment is mandatory if the number of lines (Y) specified in the frame header has the value zero.

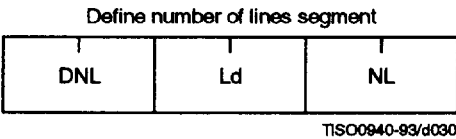


Figure B.12 – Define number of lines syntax

The marker and parameters shown in Figure B.12 are defined below. The size and allowed values of each parameter are given in Table B.10.

- DNL:** Define number of lines marker – Marks the beginning of the define number of lines segment.
- Ld:** Define number of lines segment length – Specifies the length of the define number of lines segment shown in Figure B.12 (see B.1.1.4).
- NL:** Number of lines – Specifies the number of lines in the frame (see definition of Y in B.2.2).

Table B.10 – Define number of lines segment parameter sizes and values

Parameter	Size (bits)	Values			
		Sequential DCT		Progressive DCT	Lossless
		Baseline	Extended		
Ld	16	4			
NL	16	1-65 535 ^{a)}			
^{a)} The value specified shall be consistent with the number of lines coded at the point where the DNL segment terminates the compressed data segment.					

B.3 Hierarchical syntax

B.3.1 High level hierarchical mode syntax

Figure B.13 specifies the order of the high level constituent parts of the interchange format for hierarchical encoding processes.

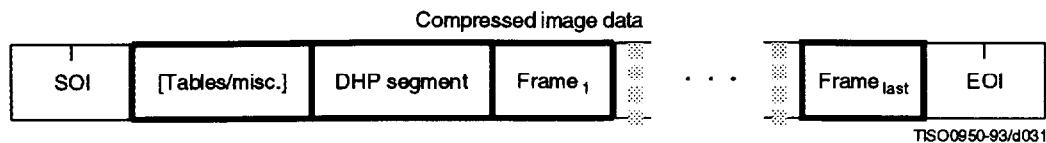


Figure B.13 – Syntax for the hierarchical mode of operation

Hierarchical mode syntax requires a DHP marker segment that appears before the non-differential frame or frames. The hierarchical mode compressed image data may include EXP marker segments and differential frames which shall follow the initial non-differential frame. The frame structure in hierarchical mode is identical to the frame structure in non-hierarchical mode.

The non-differential frames in the hierarchical sequence shall use one of the coding processes specified for SOF_n markers: SOF_0 , SOF_1 , SOF_2 , SOF_3 , SOF_9 , SOF_{10} and SOF_{11} . The differential frames shall use one of the processes specified for SOF_5 , SOF_6 , SOF_7 , SOF_{13} , SOF_{14} and SOF_{15} . The allowed combinations of SOF markers within one hierarchical sequence are specified in Annex J.

The sample precision (P) shall be constant for all frames and have the identical value as that coded in the DHP marker segment. The number of samples per line (X) for all frames shall not exceed the value coded in the DHP marker segment. If the number of lines (Y) is non-zero in the DHP marker segment, then the number of lines for all frames shall not exceed the value in the DHP marker segment.

B.3.2 DHP segment syntax

The DHP segment defines the image components, size, and sampling factors for the completed hierarchical sequence of frames. The DHP segment shall precede the first frame; a single DHP segment shall occur in the compressed image data.

The DHP segment structure is identical to the frame header syntax, except that the DHP marker is used instead of the SOF_n marker. The figures and description of B.2.2 then apply, except that the quantization table destination selector parameter shall be set to zero in the DHP segment.

B.3.3 EXP segment syntax

Figure B.14 specifies the marker segment structure for the EXP segment. The EXP segment shall be present if (and only if) expansion of the reference components is required either horizontally or vertically. The EXP segment parameters apply only to the next frame (which shall be a differential frame) in the image. If required, the EXP segment shall be one of the table-specification segments or miscellaneous marker segments preceding the frame header; the EXP segment shall not be one of the table-specification segments or miscellaneous marker segments preceding a scan header or a DHP marker segment.

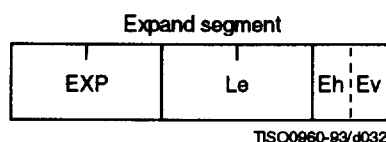


Figure B.14 – Syntax of the expand segment