Document AVC-449
March 5, 1993

CCITT SGXV
Working Party XV/1
Experts Group for ATM Video Coding
(Rapporteur's Group on Part of Q.3/XV)

INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC1/SC29/WG11

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC/JTC1/SC29/WG11

March 5, 1993

Title : MPEG-2 Working Draft (Video Part)

Status: Version 3

Source: Working Draft Editing Committee

NOTES

This document has been restructured since the last distribution. The new structure is intended to support the requirements of MPEG-2. However, much of the work required to get the technical content of the document in line with the current state of the MPEG video work has yet to be done. The working draft editing committee will continue to work on the document until the next MPEG meeting in Sydney. This distribution is therefore a "snapshot" of the working draft in its current state.

It is the goal of the working draft editing committee to have the working draft in broad agreement with the current test model by the start of the Sydney meeting. This should allow preparation of a working draft that accurately reflects the MPEG algorithm by the end of this meeting.

In its current state the working draft documents some features that are still the subject of experiments where final decisions have not yet been made. Please be assured that there is no intention to pre-empt the outcome of these decisions. The preparation of text describing the various options should help us to document the decisions made in Sydney in a timely fashion.

Throughout the text editors' notes will be found. These are enclosed in braces (i.e. { and }) and are set in italics.

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

Recommendation H.26x

ISO/IEC 11172-6

Draft of: March 5, 1993

CONTENTS

[The contents list will be reduced in size (by limiting the depth of headings included) when the editing is complete. In this verbose form it serves as a useful at-a-glance summary of the document structure.]

Foreword

The CCITT (the International Telegraph and Telephone Consultative Committee) is the permanent organ of the International Telecommunications Union (ITU). CCITT is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on worldwide basis.

The Plenary Assembly of CCITT which meets every four years, establishes the topics for study and approves Recommendations prepared by its Study Groups. The approval of Recommendations by members of CCITT between Plenary Assemblies is covered by the procedure laid down in CCITT Resolution No.2 (Melbourne, 1988).

[Editor's note: the above two paragraphs should be modified according to the outcome of the World Telecommunication Standardization Conference held in March 1993 in Helsinki.]

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, 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 JTC1. 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.

This Specification is a committee draft that is submitting for approval to CCITT, ISO-IEC/JTC1 SC29. It was prepared by SC29/WG11 also known as MPEG (Moving Pictures Expert Group). MPEG was formed in 1988 to establish a standard for coding of moving pictures and associated audio for various applications such as digital storage media, distribution and communication.

In this Specification Annex A, Annex B and Annex C contain normative requirements and are an integral part of this Specification. Annex D and Annex E are informative and contain no normative requirements.

CCITT

This Recommendation is published along with several related recommendations:

????? systems —	specifies the system coding layer of the Specification. It defines a multiplexed structure for combining audio and video data and means of representing the timing information needed to replay synchronized sequences in real-time.
????? audio —	specifies the coded representation of audio data.
????? conformance —	specifies the procedures for determining the characteristics of coded bit streams and for testing compliance with the requirements stated in ?????, H.26x and ?????.

ISO/IEC

This International Standard is published in four Parts.

11172-5 systems —	specifies the system coding layer of the Specification. It defines a multiplexed structure for combining audio and video data and means of representing the timing information needed to replay synchronized sequences in real-time.
11172-6 video —	specifies the coded representation of video data and the decoding process required to reconstruct pictures.
11172-7 audio —	specifies the coded representation of audio data.
11172-8 conformance—	specifies the procedures for determining the characteristics of coded bit streams and for testing compliance with the requirements stated in 11172-5, 11172-6 and 11172-7.

0 Introduction

0.1 Purpose

This Part of this Specification was developed in response to the growing need for a generic coding method of moving pictures and of associated sound for various applications such as digital storage media, television broadcasting and communication. The use of this Specification means that motion video can be manipulated as a form of computer data and can be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

0.2 Application

The applications of this Specification cover, but are not limited to, such areas as listed below:

CATV Cable TV Distribution on optical networks, copper, etc.

CDAD Cable Digital Audio Distribution

DAB Digital Audio Broadcasting (terrestrial and satellite broadcasting)

DTTB Digital Terrestrial Television Broadcast

EC Electronic Cinema

ENG Electronic News Gathering (including SNG, Satellite News Gathering)

HTT Home Television Theatre

IPC InterPersonal Communications (videoconferencing, videophone, etc.)

ISM Interactive Storage Media (optical disks, etc.)

NCA News and Current Affairs

NDB Networked Database Service (via ATM, etc.)

RVS Remote Video Surveillance

SSM Serial Storage Media (digital VTR, etc.)

STV Satellite TV Broadcasting

TTV Terrestrial TV Broadcasting

0.3 The "toolkit" approach

{Introducing the idea of a toolkit from which can be taken appropriate techniques for the particular application being addressed.

Introduces Annex D.}

0.4 Extensions beyond 11172-2

{A description is necessary of the facilities provided beyond MPEG-1. To me (Adrian) the following headings convey the most important features:}

0.4.1 Coding interlaced pictures

0.4.2 Scalable extensions

{Introduce the concept of scalable bitstreams. Includes a diagram showing two (or more) independent bitstreams (demultiplexed outside the realm of MPEG-2 video) being decoded by a suitable decoder.}

0.4.21 Spatial scalable extension

{Including a discussion of compatibility}

0.4.22 Frequency scalable extension

{Including a discussion of partitioning and error resilience in ATM applications}

0.4.3 4:2:2 and 4:4:4 Chrominance

{Explains that two approaches are supported. One in which a scalable higher level bitstream codes the additional chrominance. A second where the higher resolution chrominance is coded in a spatial manner embedded in the bit stream (ie. 8 block and 12 block macroblocks).}

0.5 Overview of the algorithm

{This section needs reviewing in the context of MPEG-2}

The coded representation defined in this Specification achieves a high compression ratio while preserving good picture quality. The algorithm is not lossless as the exact pixel values are not preserved during coding. The choice of the techniques is based on the need to balance a high picture quality and compression ratio with the requirement to make random access to the coded bit stream. Obtaining good picture quality at the bitrates of interest demands very high compression, which is not achievable with intraframe coding alone. The need for random access, however, is best satisfied with pure intraframe coding. This requires a careful balance between intra- and interframe coding and between recursive and non-recursive temporal redundancy reduction.

A number of techniques are used to achieve high compression. The first, which is almost independent from this Specification, is to select an appropriate spatial resolution for the signal. The algorithm then uses block-based motion compensation to reduce the temporal redundancy. Motion compensation is used both for causal prediction of the current picture from a previous picture, and for non-causal, interpolative prediction from past and future pictures. Motion vectors are defined for each 16-pixel by 16-line region of the image. The difference signal, i.e., the prediction error, is further compressed using the discrete cosine transform (DCT) to remove spatial correlation before it is quantised in an irreversible process that discards the less important information. Finally, the motion vectors are combined with the residual DCT information, and transmitted using variable length codes.

0.5.1 Temporal processing

{This section needs reviewing in the context of MPEG-2}

Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intra coded pictures (I-Pictures) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only moderate compression. Predictive coded pictures (P-Pictures) are coded more efficiently using motion compensated prediction from a past intra or predictive coded picture and are generally used as a reference for further prediction. Bidirectionally-predictive coded pictures (B-Pictures) provide the highest degree of compression but require both past and future reference pictures for motion compensation. Bidirectionally-predictive coded pictures are never used as references for prediction. The organisation of the three picture types in a sequence is very flexible. The choice is left to the encoder and will depend on the requirements of the application. Figure 0-1 illustrates the relationship among the three different picture types.

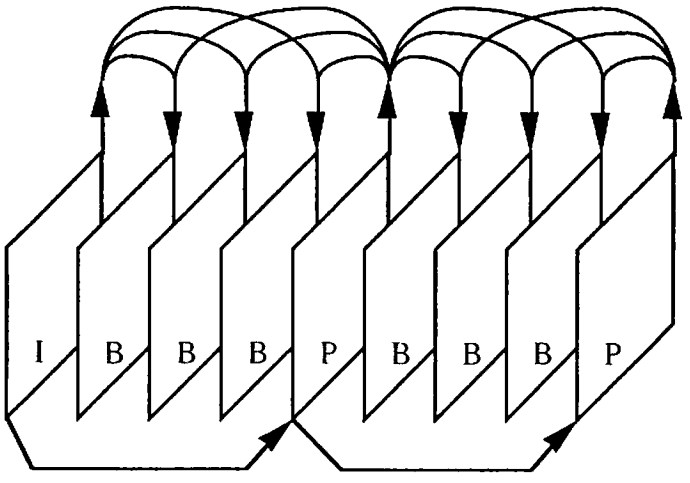


Figure 0-1 Example of temporal picture structure

The fourth picture type defined in the Specification, the D-picture, is provided to allow a simple, but limited quality, fast-forward mode.

0.5.2 Motion representation - macroblocks

[This section needs reviewing in the context of MPEG-2]

The choice of 16 by 16 macroblocks for the motion-compensation unit is a result of the trade-off between the coding gain provided by using motion information and the overhead needed to store it. Each macroblock can be one of a number of different types. For example, intra-coded, forward-predictive-coded, backward-predictive coded, and bidirectionally-predictive-coded macroblocks are permitted in bidirectionally-predictive coded pictures. Depending on the type of the macroblock, motion vector information and other side information is stored with the compressed prediction error signal in each macroblock. The motion vectors are encoded differentially with respect to the last transmitted motion vector, using variable length codes. The maximum length of the vectors that may be represented can be programmed, on a picture-by-picture basis, so that the most demanding applications can be met without compromising the performance of the system in more normal situations.

It is the responsibility of the encoder to calculate appropriate motion vectors. The Specification does not specify how this should be done.

0.5.3 Spatial redundancy reduction

[This section needs reviewing in the context of MPEG-2]

Both original pictures and prediction error signals have high spatial redundancy. This Specification uses a block-based DCT method with visually weighted quantisation and run-length coding. After motion compensated prediction or interpolation, the residual picture is split into 8 by 8 blocks. These are transformed into the DCT domain where they are weighted before being quantised. After quantisation many of the coefficients are zero in value and so two-dimensional run-length and variable length coding is used to encode the remaining coefficients efficiently.

INTERNATIONAL STANDARD 11172-6

CCITT RECOMMENDATION H.26x

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

1 Scope

This Recommendation | International Standard specifies the coded representation of picture information for digital storage media and digital video communication and specifies the decoding process. The representation supports constant bitrate transmission, variable bitrate transmission, random access, channel hopping, scalable decoding, bit stream editing, as well as special functions such as fast play, fast reverse play, normal speed reverse playback, slow motion, pause and still pictures. This Recommendation | International Standard is compatible with MPEG-1/H.261 and upward or downward compatible with EDTV, HDTV, SDTV formats.

This Recommendation | International Standard is primarily applicable to digital storage media, video broadcast and communication. The storage media may be directly connected to the decoder, or via communications means such as busses, LANs, or telecommunications links.

2 Normative references

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

- Recommendations and reports of the CCIR, 1990

  XVIIth Plenary Assembly, Dusseldorf, 1990 Volume XI - Part 1

  Broadcasting Service (Television) Rec. 601-2 "Encoding parameters of digital television for studios"

- CCIR Volume X and XI Part 3 Recommendation 648: Recording of audio signals.

- CCIR Volume X and XI Part 3 Report 955-2: Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of advanced digital system II.

- IS 11172 "Coding of moving picture and associated audio -- for digital storage media at up to about 1.5 Mbit/s," Nov. 5, 1992.

- IEEE Draft Standard "Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform", P1180/D2, July 18, 1990.

- IEC Publication 908:198, "CD Digital Audio System"

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

AC coefficient [video]: Any DCT coefficient for which the frequency in one or both dimensions is non-zero

access unit: in the case of compressed audio an access unit is an Audio Access Unit. In the case of compressed video an access unit is the coded representation of a picture.

backward motion vector [video]: A motion vector that is used for motion compensation from a reference picture at a later time in display order.

bitrate: The rate at which the compressed bit stream is delivered from the storage medium to the input of a decoder.

block [video]: An 8-row by 8-column orthogonal block of pixels.

byte aligned: A bit in a coded bit stream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.

chrominance (component) [video]: A matrix, block or sample of pixels representing one of the two colour difference signals related to the primary colours in the manner defined in CCIR Rec. 601. The symbols used for the colour difference signals are Cr and Cb.

coded order [video]: The order in which the pictures are stored and decoded. This order is not necessarily the same as the display order.

coded pictures [video]: A coded representation of one or more pictures as specified in this Specification.

coded representation: A data element as represented in its encoded form.

coded video bit stream [video]: A coded representation of a series of one or more pictures as specified in this Specification.

coding parameters [video]: The set of user-definable parameters that characterise a coded video bit stream. Bit-streams are characterised by coding parameters. Decoders are characterised by the bit streams that they are capable of decoding.

component [video]: A matrix, block or sample of pixel data from one of the three matrices (luminance and two chrominance) that make up a picture.

compression: Reduction in the number of bits used to represent an item of data.

constant bitrate coded video [video]: A compressed video bit stream with a constant average bitrate.

constant bitrate: Operation where the bitrate is constant from start to finish of the compressed bit stream.

Constrained Parameters [video]: *In the case of the video specification, the values of the set of coding parameters defined in Part 2 Clause 2.4.3.2.*

data element: An item of data as represented before encoding and after decoding.

DC-coefficient [video]: The DCT coefficient for which the frequency is zero in both dimensions.

DC-coded picture; D-picture [video]: A picture that is coded using only information from itself. Of the DCT coefficients in the coded representation, only the DC-coefficients are present.

DCT coefficient: The amplitude of a specific cosine basis function.

decoded stream: The decoded reconstruction of a compressed bit stream.

decoder input buffer [video]: The first-in first-out (FIFO) buffer specified in the video buffering verifier.

decoder input rate [video]: The data rate specified in the video buffering verifier and encoded in the coded video bit stream.

decoder: An embodiment of a decoding process.

decoding process: The process defined in this Specification that reads an input coded bit stream and outputs decoded pictures or audio samples.

dequantisation [video]: The process of rescaling the quantised DCT coefficients after their representation in the bit stream has been decoded and before they are presented to the inverse DCT digital storage media; DSM: A digital storage or transmission device or system.

discrete cosine transform; DCT [video]: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation. The inverse DCT is defined in Annex A of H.26xl11172-6

display order [video]: The order in which the decoded pictures should be displayed. Normally this is the same order in which they were presented at the input of the encoder.

editing: The process by which one or more compressed bit streams are manipulated to produce a new compressed bit stream. Conforming edited bit streams must meet the requirements defined in this Specification.

encoder: An embodiment of an encoding process.

encoding process: A process, not specified in this Specification, that reads a stream of input pictures or audio samples and produces a valid coded bit stream as defined in this Specification.

Entropy coding: Variable length noiseless coding of the digital representation of a signal to reduce redundancy.

fast forward [video]: The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.

FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).

field [video]: For interlaced source signal, a "field" is the assembly of alternate lines of a frame. Therefore. an interlaced frame is composed of two fields. namely Field1 and Field2. These two fields are merged in one frame.

Field1 [video]: Field1 consists the lines that start from the first time instant of an interlaced frame. In this Specification, the top-most active line of Field1 is defined as line 1 of an interlaced frame. Field1 shall precede Field2 in time

Field2 [video]: Field2 consists of lines that start from the second time instant of an interlaced frame. Field2 shall follow Field1 in time.

forbidden: The term 'forbidden" when used in the clauses defining the coded bit stream indicates that the value shall never be used. This is usually to avoid emulation of start codes.

forced updating [video]: The process by which macroblocks are intra-coded from time-to-time to ensure that mismatch errors between the inverse DCT processes in encoders and decoders cannot build up excessively.

forward motion vector [video]: A motion vector that is used for motion compensation from a reference picture at an earlier time in display order.

frame [video]: A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and the lines are numbered, starting from 1 at the top-most active line of a frame, continuously from the top to the bottom . For interlaced video, these lines contain samples starting from two time instants and two vertically adjacent positions and the lines are numbered, starting from 1 at the top-most line of Field1, continuously from the top to the bottom of a frame.

future reference picture [video]: The future reference picture is the reference picture that occurs at a later time than the current picture in display order.

group of pictures [video]: A series of one or more coded pictures intended to assist random access. The group of pictures is one of the layers in the coding syntax defined in Part 2 of this Specification.

Huffman coding: A specific method for entropy coding.

interlace [video]: The property of conventional television pictures where alternating lines of the picture represent different instances in time.

intra coding [video]: Compression coding of a block or picture that uses information only from that block or picture.

intra-coded picture; I-picture [video]: A picture coded using information only from itself.

layer [video and systems]: One of the levels in the data hierarchy of the video and system specifications defined in Parts 1 and 2 of this Specification.

luminance (component) [video]: A matrix, block or sample of pixels representing a monochrome representation of the signal and related to the primary colours in the manner defined in CCIR Rec. 601. The symbol used for luminance is Y.

macroblock [video]: The four 8 by 8 blocks of luminance data and the two corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the pixel data and sometimes to the coded representation of the pixel and other data elements defined in the macroblock layer of the syntax defined in Part 2 of this Specification. The usage is clear from the context.

motion compensation [video]: The use of motion vectors to improve the efficiency of the prediction of pixel values. The prediction uses motion vectors to provide offsets into the past and/or future reference frames containing previously decoded pixels that are used to form the prediction and the error difference signal.

motion estimation [video]: The process of estimating motion vectors during the encoding process.

motion vector [video]: A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture to the coordinates in a reference picture.

non-intra coding [video]: Coding of a block or picture that uses information both from itself and from blocks and pictures occurring at other times.

Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.

past reference picture [video]: The past reference picture is the reference picture that occurs at an earlier time than the current picture in display order.

pixel aspect ratio [video]: The ratio of the nominal vertical height of pixel on the display to its nominal horizontal width.

pixel [video]: An 8-bit sample of luminance or chrominance data.

picture [video]: Source or reconstructed image data. A picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. The Picture layer is one of the layers in the coding syntax defined in H.26xl11172-6 For progressive source, a picture is identical to a frame, while for interlaced video, a picture can refer to a frame, or Field1 or Field2 of the frame depending on the context.

picture period [video]: The reciprocal of the picture rate.

picture rate [video]: The nominal rate at which pictures should be output from the decoding process.

prediction [video]: The use of predictor to provide an estimate of the pixel or data element currently being decoded.

predictive-coded picture; P-picture [video]: A picture that is coded using motion compensated prediction from the past reference picture.

predictor [video]: A linear combination of previously decoded pixels or data elements.

presentation unit; PU: A decoded audio access unit or a decoded picture.

progressive [video]: The term "progressive" when used in the clauses defining the video source indicates that the picture is scanned line by line continuously at one instance in time from the top to the bottom of the picture.

quantisation matrix [video]: A set of sixty-four 8-bit scaling values used by the dequantiser.

quantised DCT coefficients: DCT coefficients before dequantisation. A variable length coded representation of quantised DCT coefficients is stored as part of the compressed video bit stream.

quantiser scale factor: A data element represented in the bit stream and used by the decoding process to scale the dequantisation.

random access: The process of beginning to read and decode the coded bit stream at an arbitrary point.

reference picture [video]: Reference pictures are the nearest adjacent I- or P-pictures to the current picture in display order.

reorder buffer [video]: A buffer in the system target decoder for storage of a reconstructed I-picture or a reconstructed P-picture.

reserved: The term "reserved" when used in the clauses defining the coded bit stream indicates that the value may be used in the future for CCITT/ISO/IEC defined extensions.

reverse play [video]: The process of displaying the picture sequence in the reverse of display order.

scalability [video]: Scalability implies the ability of decoder to ignore some portion of a total bit stream and produce useful video output from the portion which is decoded.

sequence header [video]: A block of data in the coded bit stream containing the coded representation of a number of data elements. It is one of the layers of the coding syntax defined in Part 2 of this Specification.

Side information: Information in the bit stream necessary for controlling the decoder.

skipped macroblock [video]: A macroblock for which no data is stored.

slice [video]: A series of macroblocks. It is one of the layers of the coding syntax defined in Part 2 of this Specification.

source stream: A single non-multiplexed stream of samples before compression coding.

start codes: 3 bit codes embedded in that coded bit stream that are unique. They are used for several purposes including identifying some of the layers in the coding syntax.

stuffing (bits); stuffing (bytes) [video]: Code-words that may be inserted into the compressed bit stream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream.

time-stamp: A term that indicates the time of an event.

Tonal component [audio]: A sinusoid-like component of an audio signal.

variable bitrate: Operation where the bitrate varies with time during the decoding of a compressed bit stream.

variable length coding; VLC: A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.

video buffering verifier; VBV [video]: A hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an encoder or editing process may produce.

video sequence [video]: A series of one or more groups of pictures. It is one of the layer of the coding syntax defined in part 2 of this Specification.

zig-zag scanning order [video]: A specific sequential ordering of the DCT coefficients from (approximately) the lowest spatial frequency to the highest.

4 Abbreviations and symbols

The mathematical operators used to describe this Specification are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin from zero.

{ I don't think this should be necessary ..."The bitwise operators are defined assuming two's-complement representation of integers. " and so should be deleted. And twos complement isn't spelt with an apostrophe!}

4.1 Arithmetic operators

I think it might be necessary to add ABS(). Adrian.

+	Addition.
-	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
- -	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, 7/4 and -7/-4 are truncated to 1 and -7/4 and 7/-4 are truncated to -1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and -3//2 is rounded to -2.
DIV	Integer division with truncation of the result toward - (infinitive).
%	Modulus operator. Defined only for positive numbers.
Sign( )	$Sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x == 0 \\ -1 & x < 0 \end{cases}$
NINT( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.
√	Square root.
log10	Logarithm to base ten.
loge	Logarithm to base e.

4.2 Logical operators

|	Logical OR.
&&	Logical AND.
!	Logical NOT.

4.3 Relational operators

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.

==	Equal to.
!=	Not equal to.
max [,...,]	the maximum value in the argument list.
min [,...,]	the minimum value in the argument list.

4.4 Bitwise operators

&	AND
|	OR
>>	Shift right with sign extension.
<<	Shift left with zero fill.

4.5 Assignment

=	Assignment operator.

4.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in the Specification. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.

uimsbf Unsigned integer, most significant bit first.

vlclbf Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written. The byte order of multi-byte words is most significant byte first.

4.7 Constants

π	3.14159265359...
e	2.71828182845...

5 Conventions

5.1 Structure of the coded bit stream

This Specification specifies a syntax for a compressed bit stream. This syntax contains six layers, each of which either supports a signal processing or a system function: as described in Table 5-1.

Table 5-1 --- Structure of the coded bit stream

Layers of the syntax
Sequence layer
Group of pictures layer
Picture layer
Slice layer
Macroblock layer
Block layer

5.2 Method of describing bit stream syntax

The bit stream retrieved by the decoder is described in Clause 6.2. Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in Clause 6.3. The following constructs are used to express the conditions when data elements are present, and are in normal type:

Note this syntax uses the 'C-code' convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

while (condition) {
data_element
...
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do { **data_element** ... } while (condition)	The data element always occurs at least once. The data element is repeated until the condition is not true.

if (condition) {
data_element
...
} else {
data_element
...
}
	If the condition is true, then the first group of data elements occurs next in the data stream.
	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (i = 0; i < n; i++) {
data_element
...
}
	The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} are omitted when only one data element follows.

data_element [n] data_element [n] is the n+1th element of an array of data.

data_element [m][n] data_element [m][n] is the m+1,n+1th element of a two-dimensional array of data.

While the syntax is expressed in procedural terms, it should not be assumed that Clause 6.3 implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bit stream. Actual decoders must include means to look for start codes in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardised.

5.3 Definition of functions

Several utility functions for picture coding algorithm are defined as follows:

5.3.1 Definition of bytealigned function

The function bytealigned () returns 1 if the current position is on a byte boundary, that is the next bit in the bit stream is the first bit in a byte. Otherwise it returns 0.

5.3.2 Definition of nextbits function

The function nextbits () permits comparison of a bit string with the next bits to be decoded in the bit stream.

5.3.3 Definition of next_start_code function

The next_start_code function removes any zero bit and zero byte stuffing and locates the next start code.

next_start_code() {	No. of bits	Mnemonic
while (!bytealigned())		
zero_bit	1	"0"
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
zero_byte	8	"0000 0000"
}		

This function checks whether the current position is byte aligned. If it is not, zero stuffing bits are present. After that any number of zero bytes may be present before the start-code. Therefore start-codes are always byte aligned and may be preceded by any number of zero stuffing bits.

5.4 Reserved, forbidden and marker_bit

5.4.1 Arithmetic precision

In order to reduce discrepancies between implementations of this Specification, the following rules for arithmetic operations are specified.

(a) Where arithmetic precision is not specified, such as in the calculation of DCT transform coefficients, the precision should be sufficient so that significant errors do not occur in the final integer values

(b) Where ranges of values are given by two dots, the end points are included if a bracket is present, and excluded if the 'less then' (<) and 'greater then' (>) characters are used. For example, [a .. b> means from a to b, including a but excluding b.

6 Profiles and levels

This Specification is intended to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities and services. Applications should cover, among other things, digital storage media, television broadcasting and communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax. Hence this Specification will facilitate the bitstream interchange among different applications.

Considering practicality of implementing the full specifications of this Specification at early stages, however, a limited number of subsets are also stipulated by means of "profile" and "level". These and other related terms are formally defined in clause 4 of this Specification.

{ Actually they are not defined at the moment - must remember to fix that: profile, level, parameter, flag}

A "profile" is a defined sub-set of the entire bit stream syntax that is defined by this Specification. Within the bounds impossed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by parameters in the bit stream. For instance it is possible to specify picture sizes as large as (approximately) 2^{16} pels wide by 2^{16} lines high. It is currently neither practical nor economic to implement a decoder capable of dealing with all possible picture sizes.

In order to deal with this "levels" are defined within each profile. A level is a defined set of constraints impossed on parameters in the bit stream. These constraints may be simple limits on numbers. Alternatively they may also take the form of constraints on arithmetic combinations of the parameters (e.g. picture width multiplied by picture height multiplied by picture rate).

Bit streams complying with this Specification use a common syntax. Thus a less demanding profile uses a strict sub-set of the complete syntax. In order to achieve this flags and parameters are included in the bit stream that signal the presence or otherwise of syntactic elements that occur later in the bit stream. In order to specify constraints on the syntax (and hence define a profile) it is thus only necessary to constrain the values of these flags and parameters that specify the presence of later syntactic elements.

All syntactic elements that are not constrained in a given level within a given profile may take all possible values that are allowed by this Specification. Decoders shall be able to deal with all possible values in all syntactic elements that are not constrained by the level and profile to which they conform. Decoders shall be able to deal with all values from the defined constrained set of values in all syntactic elements that are constrained by the level and profile to which they conform.

6.1 Main profile and main level

This Specification specifies a particular profile called the "main profile". Within this profile a particular level called the "main level" is specified.

The main level of the main profile is intended to be suitable for use by the largest possible number of major initial applications of this Specification in terms of both functionality requirements and cost constraints.

Clause 9 of this Specification specifies both the main profile and the main level within that profile.

6.2 Other profiles and levels

At the time of writing (February 93) only the main profile and the main level are defined by this Specification.

It is anticipated that in the future new profiles and new levels will be defined. These may be defined within the CCITT and ISO committees that drafted this Specification. Alternatively they may be defined by other committees or other organisations. In view of this the following guidlines are offered:

1 1 Compliance tests shall be carried out against a defined level of a defined profile.

2 2 Full compliance[1] to the same level of the same profile by two codecs shall be sufficient
3 guarantee of their ability to successfully decode one another's bit streams.

4 3 The number of profiles shall be kept as small as practical. In particular, the development of
5 profiles which are very similar, but not identical, shall be discouraged.

6 4 Wherever practical, a new profile shall be defined to be either a superset or a subset of each
7 existing profile. Particular attention shall be paid to try to make this true for the main profile.

8 5 The number of levels within a profile shall be kept small and, wherever practical, levels will
9 be defined so that higher level conforming decoders can decode lower level bit streams.

10 6 Wherever practical, the meaning of levels will be harmonised between different profiles.

11 7 The bit stream defined by this Specification shall support the indication of the profile and
12 level which will be required to decode the bit stream.

13 *[Editors note (Adrian) I feel a certain confusion at the present time. Both profiles and levels are*
14 *defined by constraining syntactic elements in the bit stream. The question arises which can be*
15 *constrained in profiles and which in levels? At the extremes things are clear: horizontal size can be*
16 *constrained in a level and not a profile, "fscalable" in a profile but not a level. However in the middle*
17 *there is a "fuzzy area" in which it is rather difficult to decide. For instance changes in the chroma*
18 *sampling structure currently represent syntactic changes so I would expect that it can only be*
19 *constrained in a profile. But in practical terms I can see that we might want to constrain this in a*
20 *level.*

21 *I suggest the following rule but I am not sure that this is correct: Whereever in the "C" pseudo-code*
22 *syntax there is an "if" statement the syntactic elements which are tested (or the syntactic elements*
23 *from which the tested variables are derived) may be constrained in a profile and may not be*
24 *constrained in a level.]*

---

1 *The meaning of "full compliance" is to be defined.*

7 Video bit stream syntax and semantics

This clause describes the way in which the syntax is organised (i.e. in layers) and then goes on to describe the syntax. and then the semantics.

7.1 Layered structure of video data

In general the highest level of the coded video bit stream is the video sequence. Note however that when a scalable extension is used two or more separate bit streams are decoded in a single decoder. Each of these bit streams complies with the description in this clause. See Recommendation H.22x | International Standard 11172-5 for a description of the way these separate video bitstreams may be multiplexed together. See clause 9 of this Specification for a description of the decoding process for scalable extensions.

7.1.1 Video sequence

A coded video sequence commences with a sequence header and is followed by one or more groups of pictures and is ended by a sequence_end_code. Immediately before each of the groups of pictures there may be a sequence header. Within each sequence, pictures shall be decoded continuously.

In each of these repeated sequence headers all of the data elements with the permitted exception of those defining the quantisation matrices (load_intra_quantiser_matrix, load_non_intra_quantiser_matrix and optionally intra_quantiser_matrix and non_intra_quantiser_matrix) shall have the same values as in the first sequence header. The quantisation matrices may be redefined each time that a sequence header occurs in the bit stream. Thus the data elements load_intra_quantiser_matrix, load_non_intra_quantiser_matrix and optionally intra_quantiser_matrix and non_intra_quantiser_matrix may have any (non-forbidden) values.

Repeating the sequence header allows the data elements of the initial sequence header to be repeated in order that random access into the video sequence is possible. In addition the quantisation matrices may be changed inside the video sequence as required.

7.1.2 Sequence header

A video sequence header commences with a sequence_header_code and is followed by a series of data elements.

7.1.3 Group of pictures

A group of pictures is a series of one or more consecutive pictures intended to assist random access into the sequence. In the coded bitstream, the first coded picture in a group of pictures is an I-picture. The order of the pictures in the coded bitstream is the order in which the decoder processes them. In display order, the last picture in a group of pictures is always an I-Picture or a P-Picture, and the first is either an I-Picture or the first B-Picture of the consecutive series of B-Pictures which immediately precedes the first I-Picture.

The following is an example of groups of pictures taken from the beginning of a video sequence. In this example the first group of pictures contains seven pictures and subsequent groups of pictures contain nine pictures. There are two B-pictures between successive P-pictures and also two B-pictures between successive I- and P-pictures. Picture '1I' is used to form a prediction for picture '4P'. Pictures '4P' and '1I' are both used to form predictions for pictures '2B' and '3B'. Therefore the order of pictures in the coded sequence shall be '1I', '4P', '2B', '3B'. However, the decoder should display them in the order '1I', '2B', '3B', '4P'.

At the encoder input,

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I

At the encoder output, in the coded bitstream, and at the decoder input,

1	4	2	3	7	5	6	10	8	9	13	11	12	16	14	15	19	17	18	22	20	21	25	23	24	28	26	27
I	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B

where the double vertical bars mark the group of pictures boundaries. Note that in this example, the first group of pictures is two pictures shorter than in subsequent groups of pictures, since at the beginning of video coding there are no B-pictures preceding the first I-Picture. However, in general, in

display order, there may be B-Pictures preceding the first I-Picture in the group of pictures, even for the first group of pictures to be decoded.

At the decoder output,

A group of pictures may be of any length. A group of pictures shall contain one or more I-Pictures. Applications requiring random access, fast-forward playback, or fast and normal reverse playback may use relatively short groups of pictures. Groups of pictures may also be started at scene cuts or other cases where motion compensation is ineffective.

The number of consecutive B-Pictures is variable. Neither B- nor P-Pictures need be present.

A video sequence of groups of pictures input to the decoder may be different from the one at the encoder output due to editing.

7.1.4 Picture

A source or reconstructed picture consists of three rectangular matrices of eight-bit numbers; a luminance matrix (Y), and two chrominance matrices (Cr and Cb). The Y-matrix shall have an even number of rows and columns. The Cr and Cb matrices are dependent to the one of the three chromatic formats.

There are four types of pictures that use different coding methods.

An **Intra-coded (I) picture** is coded using information only from itself.

A **Predictive-coded (P) picture** is a picture which is coded using motion compensated prediction from a past I-Picture or P-Picture.

A **Bidirectionally predictive-coded (B) picture** is a picture which is coded using motion compensated prediction from a past and/or future I-Picture or P-Picture.

A **DC coded (D) picture** is coded using information only from itself. Of the DCT coefficients only the DC ones are present. The D-pictures shall not be in a sequence containing any other picture types.

A picture can represent either a frame or a field as elaborated below:

7.1.41 Field picture

A **field picture** refers to either Field1 or Field2 of a frame. The transmission order of the fields is fixed and same as display order (Field1 first, Field2 second.) in case of the simplified syntax. Each encoded field picture is constrained in one picture layer of the video bitstream.

In field P- and B-pictures, both fields must be P- or B-. The fields of those field pictures are called P-Field and B-Field.

For an intra field picture, however, it is possible (but not required) to use predictive-coding-type to transmit the second field.

7.1.42 Frame picture

A frame picture refers to a complete frame. When the frame consists of two component fields, these shall be merged as specified in 6.2.1 to form a frame picture. Each encoded frame picture is constrained in one picture layer of the video bitstream.

7.1.5 Slice

A **slice** is a series of an arbitrary number of macroblocks with the order of macroblocks starting from the upper-left of the picture and proceeding by raster-scan order from left to right and top to bottom. The first and last macroblocks of a slice shall not be skipped macroblocks (see Clause 6.6.?). Every slice shall contain at least one macroblock. Slices shall not overlap and there shall be no gaps between slices. The position of slices may change from picture to picture. The first slice shall start with the first macroblock in the picture and the last slice shall end with the last macroblock in the picture.

A frame is divided into a number of contiguous macroblock slices. The number of Macro block Slices (MBS) differs by the source formats. These MBSs cover the significant pixel area.

7.1.6 Macroblock

A **macroblock** contains a 16-pel by 16-line section of luminance component and the spatially corresponding chrominance component. Macroblock can either refer to source and decoded data or to

the corresponding coded data elements. A skipped macroblock is one for which no information is stored (see Clause 6.6.?). There are three chroma formats for a macroblock, namely, 4:2:0, 4:2:2 and 4:4:4 formats. The orders of blocks in a macroblock are different for different chroma formats and are illustrated below:

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and the block order is depicted in figure 7-1a.

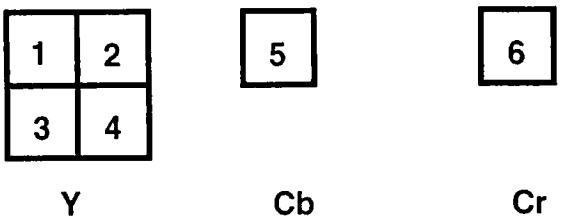


Figure 7-1a 4:2:0 Macroblock structure

A 4:2:2 Macroblock consists of 8 blocks. This structure holds 4 Y, 2 Cb and 2 Cr Blocks and the block order is depicted in figure 7-1b.

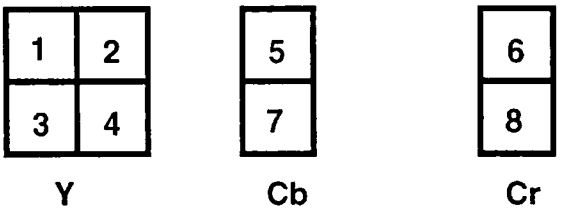


Figure 7-1b 4:2:2 Macroblock structure

A 4:4:4 Macroblock consists of 16 blocks. This structure holds 4 Y, 4 Cb and 4 Cr Blocks and the block order is depicted in figure 7-1c.

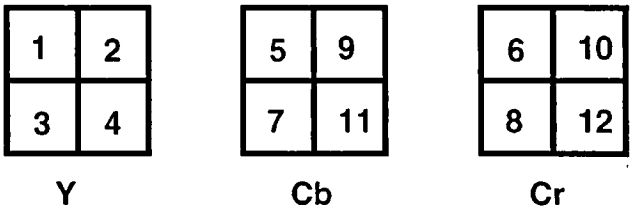


Figure 7-1c 4:4:4 Macroblock structure

The internal organisation within the Macroblock is different for Frame and Field DCT coding, and is depicted for the luminance blocks in figure 7-2 and 7-3. The chrominance block is in frame order for both DCT coding macroblock types.

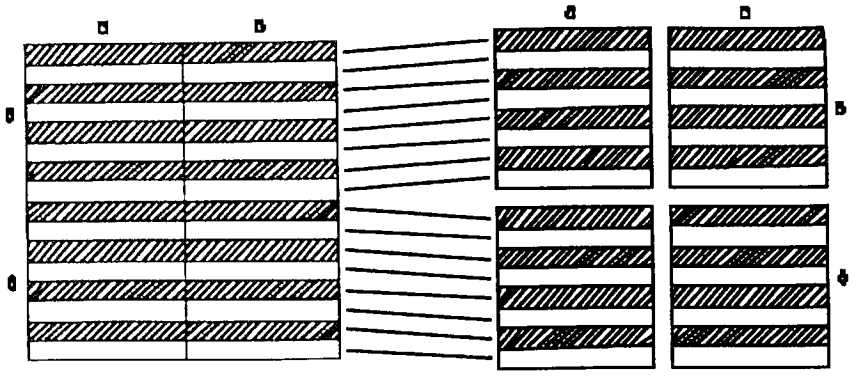


Figure 7-2 Luminance macroblock structure in frame DCT coding

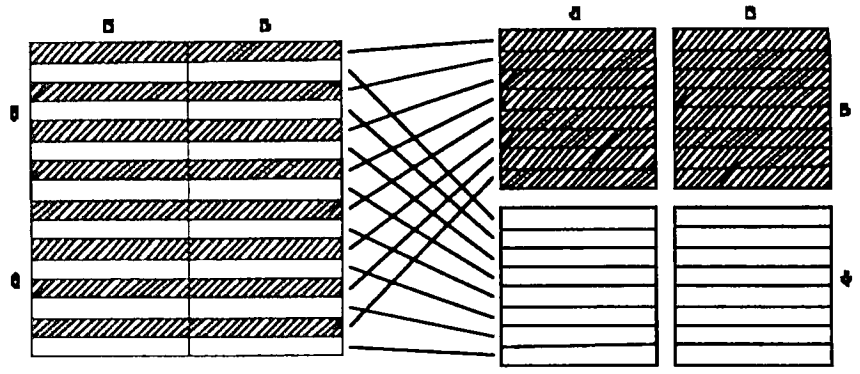


Figure 7-3 Luminance macroblock structure in field DCT coding

7.1.7 Block

A Block consists of an array of 8x8 coefficients. Figure 7-4 shows coefficients in the block in zigzag scanned order.

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

+----------> increasing cycles per width

|

|

V

Increasing cycles per picture height

Figure 7-4 Coefficients in a block

7.2 Video bit stream syntax

{This section specifies the bit stream for the whole of the whole of the MPEG-2 video standard. Since 7.3 concentrates on semantics it is necessary to be clear about what constitutes the syntax. I think we should aim at a definition based on the concept that with only an understanding of the syntax it should be possible to parse the bitstream from beginning to end without loosing ones place (although no meaning could be attributed to the recovered symbols). In practice this will not be possible since MPEG is such a highly context senisitive language. However the goal is still useful.}

7.2.1 Start codes

Start codes are reserved bit patterns that do not otherwise occur in the video stream.

Each start code consists of a start code prefix followed by a start code value. The start code prefix is a string of twenty three bits with the value zero followed by a single bit with the value zero. The start code prefix is thus the bit stream "0000 0000 0000 0000 0000 0001".

The start code value is an eight bit integer which identifies the type of start code. Most types of start code have just one start code value. However slice_start_code is represented by many start code values, in this case the start code value is the slice vertical position for the slice.

All start codes shall be byte aligned. This shall be achieved by inserting bits with the value zero before the start code prefix such that the first bit of the start code prefix is the most significant bit of a byte.

Table 6-8 defines the slice code values for the start codes used in the video bit stream.

Table 7-1 — Start code values

name	start code value (hexadecimal)
picture_start_code	00
slice_start_code	01 through AF
reserved	B0
reserved	B1
user_data_start_code	B2
sequence_header_code	B3
sequence_error_code	B4
extension_start_code	B5
reserved	B6
sequence_end_code	B7
group_start_code	B8
system start codes (see note)	B9 through FF
NOTE - system start codes are defined in Part 1 of this Specification	

The use of the start codes is defined in the following syntax description with the exception of the sequence_error_code. The sequence_error_code has been allocated for use by the digital storage media interface to indicate where uncorrectable errors have been detected.

1 **7.2.2 Video sequence layer**

video_sequence() {	No. of bits	Mnemonic
next_start_code()		
do {		
sequence_header()		
do {		
group_of_pictures()		
} while (nextbits() == group_start_code)		
} while (nextbits() == sequence_header_code)		
sequence_end_code	32	bslbf
}		

2

7.2.3 Sequence header

sequence_header() {	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
pel_aspect_ratio	4	uimsbf
picture_rate	4	uimsbf
bit_rate	18	uimsbf
marker_bit	1	"1"
vbv_buffer_size	10	uimsbf
constrained_parameter_flag	1	
load_intra_quantizer_matrix	1	
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix[64]	8*64	uimsbf
load_non_intra_quantizer_matrix	1	
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix[64]	8*64	uimsbf
next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
spatial_embedded	1	uimsbf
fscalable	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	4	uimsbf
vertical_size_extension	4	uimsbf
video_format	3	uimsbf
intra_vlc_format	1	uimsbf
tcoeff_escape_format *(this was removed in Rome? Adrian)*	*1*	*uimsbf*
if (fscalable) {		
interlaced	1	uimsbf
subband	1	uimsbf
linked_prediction	1	uimsbf
scalable_side_information	1	uimsbf
motion_refinement	1	uimsbf
do {		
fscale_code	8	uimsbf
motion_compensation_loop	1	uimsbf
} while (nextbits != '0000 0111')		
end_of_fscales_code	8	'0000 0111'
}		

if (spatial_embedded) {		
subsampling_ratio	3	uimsbf
compatible_mtype	2	uimsbf
load_prediction_weighting_matrix	1	uimsbf
if (load_prediction_weighting_matrix) {		
prediction_weighting_matrix[8]	4*8	uimsbf
}		
}		
display_horizontal_dimension	16	uimsbf
display_vertical_dimension	16	uimsbf
if (fscalable || spatial_embedded) {		
low_resolution_prediction_horizontal_dimension	17	uimsbf
low_resolution_prediction_vertical_dimension	17	uimsbf
}		
usnb_flag	1	uimsbf
reserved /* byte align */	?	uimsbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
sequence_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
}		

1 **7.2.4 Group of pictures layer**

group_of_pictures() {	No. of bits	Mnemonic
group_start_code	32	bslbf
time_code	25	
closed_gop	1	
broken_link	1	
next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
group_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
do {		
picture()		
} while (nextbits() == picture_start_code)		
}		

2

3 **7.2.5 Picture layer**

picture() {	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if (picture_coding_type == 2 || picture_coding_type == 3) {		
full_pel_forward_vector	*1*	
forward_f_code	*3*	*uimsbf*
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	*1*	
backward_f_code	*3*	*uimsbf*
}		
while (nextbits() == '1') {		
extra_bit_picture	1	"1"
extra_information_picture	8	
}		
extra_bit_picture	1	"0"

4

next_start_code()		
if (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
if (picture_coding_type == 2 || picture_coding_type == 3) {		
forward_vertical_f_code	*3*	*uimsbf*
}		
if (picture_coding_type == 3) {		
backward_vertical_f_code	*3*	*uimsbf*
}		
picture_structure	2	uimsbf
interlace_progressive_flag	2	uimsbf
if ((picture_structure == 'frame_picture') &&		
(interlace_progressive_flag == 'interlace')) {		
top_field_first_flag	1	uimsbf
number_of_fields_displayed_code	3	uimsbf
}		
forward_reference_fields	2	uimsbf
backward_reference_fields	2	uimsbf
if (chroma_format == "01") { /* 4:2:0 */		
chroma_postprocessing_type	1	uimsbf
} else {		
reserved	1	uimsbf
}		
if (video_format != '000') { /* composite input */		
v-axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
for (i=0; i<number_of_fields_displayed; i++) {		
pan_horizontal_left_upper_offset	16 ?	uimsbf
pan_vertical_left_upper_offset	16 ?	uimsbf
}		
if (fscalable || spatial_embedded) {		
overlap_horizontal_left_upper_offset	17	uimsbf
overlap_horizontal_left_upper_offset	17	uimsbf
if (interlaced &&		
(picture_structure == frame_structure)) {		
overlap_horizontal_left_upper_offset	17	uimsbf
overlap_horizontal_left_upper_offset	17	uimsbf
}		
}		

1

intra_dc_precision	2	uimsbf
qscale_type	1	uimsbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
picture_extension_data	8	
}		
next_start_code()		
}		
if (nextbits() == user_data_start_code) {		
user_data_start_code	32	bslbf
while (nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		
if (picture_coding_type != '101') { /* skipped */		
do {		
slice()		
} while (nextbits() == slice_start_code)		
}		
}		

1

2 **7.2.6 Slice layer**

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
quantizer_scale	5	uimsbf
if (fscalable) {		
extra_bit_slice	1	"1"
dct_size	8	uimsbf
}		
while (nextbits() == '1') {		
extra_bit_slice	1	"1"
extra_information_slice	8	
}		
extra_bit_slice	1	"0"
do {		
macroblock()		
} while (nextbits() != '000 0000 0000 0000 0000 0000')		
next_start_code()		
}		

3

7.2.7 Macroblock layer

macroblock() {	No. of bits	Mnemonic
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	vlclbf
macroblock_address_increment	1-11	vlclbf
macroblock_type	1-8	vlclbf
if (macroblock_compatible &&		
(picture_coding_type != 1) && (compatible_mtype != '10'))		
prediction_weight_code	1-3	vlclbf
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == "11") { /* Frame-Picture */		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == "11") && /* Frame-Picture */		
(macroblock_intra || macroblock_pattern) &&		
(interlaced_progressive_flag == 'interlaced'))		
dct_type	1	uimsbf
}		
if (macroblock_quant)		
quantizer_scale	5	uimsbf
if (macroblock_motion_forward)		
forward_motion_vectors()	...	...
if (macroblock_motion_backward)		
backward_motion_vectors()	...	...
if (macroblock_pattern)		
coded_block_pattern()	...	...
for (i=0; i<block_count; i++) {		
block(i)		
}		
if (picture_coding_type == 4)		
end_of_macroblock	1	"1"
}		

forward_motion_vectors () {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if (mv_format == frame) {		
motion_vector()	...	...
} else {		
forward_field_motion_vector()	...	...
}	...	...
} else {		
forward_field_motion_vector()	...	...
forward_field_motion_vector()	...	...
}		
}		

1

backward_motion_vectors () {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if (mv_format == frame) {		
motion_vector()	...	...
} else {		
backward_field_motion_vector()	...	...
}	...	...
} else {		
backward_field_motion_vector()	...	...
backward_field_motion_vector()	...	...
}		
}		

2

motion_vector () {	No. of bits	Mnemonic
motion_horizontal_size	1-13	vlclbf
if (motion_horizontal_size != 0)		
motion_horizontal_mag	1-12	uimsbf
motion_vertical_size	1-13	vlclbf
if (motion_vertical_size != 0)		
motion_vertical_mag	1-12	uimsbf
}		

3

forward_field_motion_vector () {	No. of bits	Mnemonic
if (forward_reference_field == "11")		
motion_vertical_field_select	**1**	**uimsbf**
motion_vector()	...	...
}		

4

backward_field_motion_vector () {	No. of bits	Mnemonic
if (backward_reference_field == "11")		
motion_vertical_field_select	**1**	**uimsbf**
motion_vector()	...	...
}		

coded_block_pattern () {	No. of bits	Mnemonic
coded_block_pattern_420	3-9	vlclbf
if ((chroma_format == 4:4:4) || (chroma_format == 4:2:2))		
coded_block_pattern_1	2	uimsbf
if (chroma_format == 4:4:4) {		
coded_block_pattern_2	2	uimsbf
coded_block_pattern_3	2	uimsbf
}		
}		

7.2.8 Block layer

The detailed syntax for the terms "First DCT coefficient", "Subsequent DCT coefficients" and "End of Block" is described below.

block(i) {	No. of bits	Mnemonic
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	**2-7**	**vlclbf**
if(dct_dc_size_luminance != 0)		
dct_dc_differential	**1-8**	**uimsbf**
} else {		
dct_dc_size_chrominance	**2-8**	**vlclbf**
if(dct_dc_size_chrominance !=0)		
dct_dc_differential	**1-8**	**uimsbf**
}		
} else {		
First DCT coefficient	...	
}		
if (picture_coding_type != 4) {		
while (nextbits() != End of block)		
Subsequent DCT coefficients	...	
End of block	...	
}		
}		
}		

7.2.81 Subsequent DCT coefficients

Two different tables are used for representing the DCT coefficients. One table shown in Table B-11 of Annex B is optimised for coefficients with small amplitudes. The other shown in Table B-12 of Annex B is optimised for coefficients with large amplitudes. Neither table contains entries for all

possible combinations of *run* and *level*. An "escape" mechanism is provided in order to deal with these combinations of *run* and *level*. which occur infrequently. In this case the entry marked as "ESCAPE" in Table B-11 and Table B-12 shall be used. This shall then followed by *run* and then *level* coded as fixed length codes as shown in Table B-13 of Annex B.

7.2.811 DCT coefficient table selection

Selection between the two tables is made dynamically as the decoding progresses. In order to do this a variable, *j*, is maintained which may take the values zero and one. When *j* is zero then Table B-11 shall be used to decode the subsequent coefficient. When *j* is one then Table B-12 shall be used to decode the subsequent coefficient. After each coefficient is decoded the value of *j* is updated depending upon the "run" and "level" just decoded;

```
if ( j == 0) {
    if ( ( run == 0) && (ABS(level) > 1) )
        j = 1;
} else {
    if ( ( run > 1) && (ABS(level) == 1) )
        j = 0;
}
```

If end_of_block is decoded then *j* is set to 0.

7.2.812 DCT coefficient table predictors

j is initially set at the beginning of the block to a value that is predicted from an earlier block. In order to do this four DCT coefficient table predictors are maintained:

- *pred_vlc_intra_luma*
- *pred_vlc_intra_chroma*
- *pred_vlc_inter_luma*
- *pred_vlc_inter_chroma*

pred_vlc_intra_luma and *pred_vlc_intra_chroma* shall be used in macroblocks where **macroblock_intra** is 1. *pred_vlc_inter_luma* and *pred_vlc_inter_chroma* shall be used in macroblocks where **macroblock_intra** is 0. *pred_vlc_intra_luma* and *pred_vlc_inter_luma* shall be used when *i* (the block index) is less than four. *pred_vlc_intra_chroma* and *pred_vlc_inter_chroma* shall be used when *i* (the block index) is greater than three.

At the start of the block *j* shall be initialised to the value stored in the appropriate predictor. After the value of *j* has been updated for the first time in a given block *j* is stored in the appropriate predictor. None of the predictors take any further part in the decoding of that block.

At the start of a slice all four DCT coefficient table predictors shall be reset to theinitial values shown in table 7-2

Table 7-2 — Initial values for predictors

Predictor	Initial value
pred_vlc_intra_luma	1
pred_vlc_intra_chroma	1
pred_vlc_inter_luma	0
pred_vlc_inter_chroma	0

When a macroblock is skipped (i.e. **macroblock_address_increment** indicates an address increment greater than 1) pred_vlc_inter_luma and pred_vlc_inter_chroma shall be reset to zero.

When a block within a macroblock is not coded (i.e. pattern_code[i] is zero) the appropriate DCT coefficient table predictor for that block (pred_vlc_inter_luma or pred_vlc_inter_chroma) shall be reset to zero.

7.2.82 First DCT coefficient

When the very first coefficient in a block (the DC coefficient) is coded using Table B-11 the table is modified slightly. This occurs when the appropriate DCT coefficient table predictor has the value zero

at the start of the block, pattern_code[i] is one and **macroblock_intra** is zero In this case the the table is modified as per Note-2 and Note-3 of Table B-11.

In all other respects the syntax for the "First DCT coefficient" is the same as for "Subsequent DCT coefficients".

7.2.83 End of Block

When required by the syntax for the block level the "End of Block" symbol is used to indicate the end of the block. The "End of Block" entry in either Table B-11 or Table B-12 is used depending upon the value of *j* at the time that it is encoutered.

7.3 Video bit stream semantics

{This section expands on 7.2 to introduce the semantics. Essentially this section defines the meaning associated with the data recovered by the syntax parser. I feel that it would be useful to move some of the information currently in section 8 into this section. A goal would be that at the end of this process of semantic understanding we have a a series of numbers which we understand. In addition the DCT coefficient data has been recovered to the point where the quantizer is ready to deal with it (ie. run's have been expanded out) and motion vectors have been fully recovered into X-Y coordinates.}

7.3.1 Sequence layer

sequence_end_code -- The sequence_end_code is the bit string 000001B7 in hexadecimal. It terminates a video sequence.

7.3.2 Sequence header

sequence_header_code -- The sequence_header_code is the bit string 0000 01B3 in hexadecimal. It identifies the beginning of a sequence header.

horizontal_size_value -- This word forms the 12 least significant bits from horizontal_size.

vertical_size_value -- This word forms the 12 least significant bits from vertical_size.

horizontal_size -- The horizontal_size is a 16 bit unsigned integer, the 12 least significant bits are defined in horizontal_size_value, the 4 most significant bits are defined in horizontal_size_extension. The horizontal_size is the width of the displayable part of each luminance picture in pixels. The width of the encoded luminance picture in macroblocks, mb_width, is (horizontal_size+15)/16. The displayable part of the picture is left-aligned in the encoded picture.

vertical_size -- The vertical_size is a 16 bit unsigned integer, the 12 least significant bits are defined in vertical_size_value, the 4 most significant bits are defined in vertical_size_extension. The vertical_size is the height of the displayable part of each luminance picture in pixels. The height of the encoded luminance picture in macroblocks, mb_height, is (vertical_size+15)/16. The displayable part of the picture is top-aligned in the encoded picture.

pel_aspect_ratio -- This is a four-bit integer defined in the Table 7-3.

Table 7-3--- pel_aspect_ratio

pel_aspect_ratio	height/width	example
0000	forbidden	
0001	1.0000	VGA etc.
0010	0.6735	
0011	0.7031	16:9,625line
0100	0.7615	
0101	0.8055	
0110	0.8437	16:9, 525line
0111	0.8935	
1000	0.9157	CCIR601, 625line
1001	0.9815	
1010	1.0255	
1011	1.0695	
1100	1.0950	CCIR601, 525line
1101	1.1575	
1110	1.2015	
1111	reserved	

picture_rate -- This is a four-bit integer defined in the following Table.6-10.

Table 7-4 --- picture_rate

picture_rate	pictures per second
0000	forbidden
0001	23.976
0010	24
0011	25
0100	29.97
0101	30
0110	50
0111	59.94
1000	60
. . .	reserved
1111	reserved

bit_rate -- This is an integer specifying the bit rate of the bit stream measured in units of 400 bits/second, rounded upwards. The value zero is forbidden. The value 3FFFF (11 1111 1111 1111 1111) identifies variable bit rate operation.

marker_bit -- This is one bit that shall be set to "1". This bit prevents emulation of start codes.

vbv_buffer_size -- This is a 10-bit integer defining the size of the VBV (Video Buffering Verifier, see Annex C) buffer needed to decode the sequence. It is defined as:

$$B = 16 * 1024 * vbv\_buffer\_size$$

where B is the minimum VBV buffer size in bits required to decode the sequence (see Annex C).

load_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if intra_quantiser_matrix follows. If it is set to "0" then the default values defined below are used until the next occurrence of the sequence header.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, stored in the zigzag scanning order shown in Clause 6.6.3, replace the default values shown above. The value for intra_quant[0][0] shall always be 8. For the 8-bit unsigned integers, the value zero is forbidden. The new values shall be in effect until the next occurrence of a sequence header.

load_non_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if non_intra_quantiser_matrix follows. If it is set to "0" then the default values defined below are used until the next occurrence of the sequence header.

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

non_intra_quantiser_matrix -- This is a list of sixty-four 8-bit unsigned integers. The new values, stored in the zigzag scanning order shown in Clause 6.6.3, replace the default values shown above. For

the 8-bit unsigned integers, the value zero is forbidden. The new values shall be in effect until the next occurrence of a sequence header.

extension_start_code -- The extension_start_code is the bit string 000001B5 in hexadecimal. It identifies the beginning of extension data. The extension data continue until receipt of another start code. It is a requirement to parse extension data correctly.

fscalable -- This is a one-bit integer defined in the following table.

0	not fscalable
1	fscalable

Default value: 0

sscalable -- This is a one-bit integer defined in the following table.

0	not sscalable
1	sscalable

Default value: 0

chroma_format - This is a 2 bit integer defined in the following table:

00	reserved
01	4:2:0
10	4:2:2
11	4:4:4

Default value: 01

extent_horizontal_size -- This is a one-bit integer defined in the following table.

0	No horizontal extension
1	Horizontal extension

Default value: 0

horizontal_size_extension -- This word forms the 4 most significant bits from horizontal_size.

extent_vertical_size -- This is a one-bit integer defined in the following table.

0	No vertical extension
1	Vertical extension

Default value: 0

vertical_size_extension -- This word forms the 4 most significant bits from vertical_size.

intra_vlc_format -- This is a one bit integer to indicate if table B.5h is used for intra macroblocks instead of the default tables B.5c through B.5f.

intra_vlc_format	
0	MPEG-1 VLC
1	Alternative Intra VLC

Default value: 0

tcoef_escape_format -- this is a one bit integer in order to increase the maximum range of the transmitted coefficients from +/- 255 to +/- 2047. **tcoef_escape_format** indicates that a single 12-bit level *tcoef* escape is used from table B.5i, rather than table B.5g.

tcoef_escape_format	
0	MPEG-1 Escape
1	12-bit Escape

Default value: 0.

NOTE: If this bit is set coefficient clamping at +/-255 is changed to +/- 2047.

sequence_extension_data -- Reserved.

user_data_start_code -- The user_data_start_code is the bit string 000001B2 in hexadecimal. It identifies the beginning of user data. The user data continues until receipt of another start code.

user_data -- The user_data is defined by the users for their specific applications. The user data shall not contain a string of 23 or more zero bits.

7.3.3 Group of pictures layer

group_start_code -- The group_start_code is the bit string 000001B8 in hexadecimal. It identifies the beginning of a group of pictures.

time_code -- This is a 25-bit field containing the following: drop_frame_flag, time_code_hours, time_code_minutes, marker_bit, time_code_seconds and time_code_pictures. The fields correspond to the fields defined in the IEC standard for "time and control codes for video tape recorders" (see Bibliography, Annex E). The code refers to the first picture in the group of pictures that has a temporal_reference of zero. The drop_frame_flag can be set to either "0" or "1". It may be set to "1" only if the picture rate is 29.97Hz. If it is "0" then pictures are counted assuming rounding to the nearest integral number of pictures per second, for example 29.97Hz would be rounded to and counted as 30Hz. If it is "1" then picture numbers 0 and 1 at the start of each minute, except minutes 0, 10, 20, 30, 40, 50 are omitted from the count.

Table 7-5 --- time_code

time_code	range of value	No. of bits	Mnemonic
drop_frame_flag		1	
time_code_hours	0 - 23	5	uimsbf
time_code_minutes	0 - 59	6	uimsbf
marker_bit	1	1	"1"
time_code_seconds	0 - 59	6	uimsbf
time_code_pictures	0 - 59	6	uimsbf

closed_gop -- This is a one-bit flag which is set to "1" if the group of pictures has been encoded without prediction vectors pointing to the previous group of pictures.

This bit is provided for use during any editing which occurs after encoding. If the previous group of pictures is removed by editing, broken_link may be set to "1" so that a decoder may avoid displaying the B-Pictures immediately following the first I-Picture of the group of pictures. However if the closed_gop bit indicates that there are no prediction references to the previous group of pictures then the editor may choose not to set the broken_link bit as these B-Pictures can be correctly decoded in this case.

broken_link -- This is a one-bit flag which shall be set to "0" during encoding. It is set to "1" to indicate that the B-Pictures immediately following the first I-Picture of a group of pictures cannot be correctly decoded because the other I-Picture or P-Picture which is used for prediction is not available (because of the action of editing).

A decoder may use this flag to avoid displaying pictures that cannot be correctly decoded.

extension_start_code -- See Clause 6.5.2.

group_extension_data -- Reserved. user_data_start_code -- See Clause 6.5.2. user_data -- See Clause 6.5.2.

7.3.4 Picture layer

picture_start_code -- The picture_start_code is a string of 3 bits having the value 00000100 in hexadecimal.

temporal_reference -- The temporal_reference is a 10-bit unsigned integer associated with each input picture. It is incremented by one, modulo 1024, for each input picture. For the earliest picture (in display order) in each group of pictures, the temporal_reference is reset to zero.

The temporal_reference is assigned (in sequence) to the pictures in display order, no temporal_reference shall be omitted from the sequence.

picture_coding_type -- The picture_coding_type identifies whether a picture is an intra-coded picture(I), predictive-coded picture(P), bidirectionally predictive-coded picture(B), or intra-coded with only DC coefficients (D) according to the following Table. D-pictures shall never be included in the same video sequence as the other picture coding types.

Table 7-6 --- picture_coding_type

picture_coding_type	coding method
000	forbidden
001	intra-coded (I)
010	predictive-coded (P)
011	bidirectionally-predictive-coded (B)
100	dc intra-coded (D)
101	reserved
...	...
111	reserved

vbv_delay -- The vbv_delay is a 16-bit unsigned integer. For constant bitrate operation, the vbv_delay is used to set the initial occupancy of the decoder's buffer at the start of play so that the decoder's buffer does not overflow or underflow. The vbv_delay measures the time needed to fill the VBV buffer from an initially empty state at the target bit rate, R, to the correct level immediately before the current picture is removed from the buffer.

The value of vbv_delay is the number of periods of the 90kHz system clock that the VBV should wait after receiving the final byte of the picture start code. It may be calculated from the state of the VBV as follows:

$$vbv\_delay_n = 90000 * B_n^* / R$$

where:

$n > 0$

B_n^* = VBV occupancy immediately before removing picture n from the buffer but after removing any group of picture layer and sequence header data that immediately precedes picture n.

R = bit rate as defined by bit_rate in the sequence header.

For non-constant bitrate operation vbv_delay shall have the value FFFF in hexadecimal.

full_pel_forward_vector -- If set to "1", then the motion vector values decoded represent integer pixel offsets (rather than half-pixel units) as reflected in the equations of Clause 6.6.2.

forward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. forward_r_size and forward_f used in the process of decoding the forward motion vectors are derived from forward_f_code as described in Clause 6.6.2

full_pel_backward_vector -- If set to "1", then the motion vector values decoded represent integer pixel offsets (rather than half pixel units) as reflected in the equations of Clause 6.6.3.

backward_f_code -- An unsigned integer taking values 1 through 7. The value zero is forbidden. backward_r_size and backward_f used in the process of decoding the backward motion vectors are derived from backward_f_code as described in Clause 6.6.3.

extra_bit_picture -- A bit indicates the presence of the following extra information. If extra_bit_picture is set to "1", extra_information_picture will follow it. If it is set to "0", there are no data following it.

extra_information_picture -- Reserved.

extension_start_code -- See Clause 6.5.2.

forward_vertical_f_code ??

backward_vertical_f_code ??

picture_structure - This is a 2-bit integer defined in the table below.

11	Frame-Picture
01	Field 1 of a Field-Picture
10	Field 2 of a Field-Picture
00	reserved

Default value: 11

forward_reference_fields - This is a 2-bit integer defined in the table below, the issue of the default value is still not resolved.

11	Forward prediction from Field 1 and Field 2
01	Forward prediction only from Field 1
10	Forward prediction only from Field 2
00	No forward prediction in this Picture

In I-Pictures this field is always set to "00".

Default value: 11, what is the meaning of this field for non-interlaced pictures?

backward_reference_fields - This is a 2-bit integer defined in the table below.

11	Backward prediction from Field 1 and Field 2
01	Backward prediction only from Field 1
10	Backward prediction only from Field 2
00	No backward prediction in this Picture

In I-Pictures and P-Pictures this field is always set to "00".

Default value: 11, what is the meaning of this field for non-interlaced pictures?

chroma_postprocessing_type - This is a 1-bit integer that indicate how the chroma samples of a 4:2:0 Picture must be postprocessed for display. It is defined in the table below.

0	SIF interlaced (for interlaced Pictures)
1	SIF (for progressive Pictures)

Default value: 1

intra_dc_precision - This is a 2-bit integer defined in the table below.

00	dc_precision 8bit
01	dc_precision 9bit
10	dc_precision 10bit
11	dc_precision 11bitd

Default value: 00

Remark

According to this indicator, the step-size for quantizing and dequantizing the intra DC coefficients is changed and also the initialized or reset value for the intra DC predictor is changed, defined in the following table:

intra_dc_precision	Stepsize for DC	Initial or reset value
00	8	128
01	4	256
10	2	512
11	1	1024

qscale_type - This is a 1-bit integer defined in the table below.

0	MPEG1 compatible:linear
1	non linear law

Default value: 0

picture_extension_data -- Reserved.

user_data_start_code -- See Clause 6.5.2.

user_data -- See Clause 6.5.2.

7.3.5 Slice layer

slice_start_code -- The slice_start_code is a string of 32-bits. The first 24-bits have the value 000001 in hexadecimal and the last 8-bits are the slice_vertical_position having a value in the range 01 through AF hexadecimal inclusive.

slice_vertical_position -- This is given by the last eight bits of the slice_start_code. It is an unsigned integer giving the vertical position in macroblock units of the first macroblock in the slice. The slice_vertical_position of the first row of macroblocks is one. Some slices may have the same slice_vertical_position, since slices may start and finish anywhere. Note that the slice_vertical_position is constrained by Clause 6.3.4 to define non-overlapping slices with no gaps between them. The maximum value of slice_vertical_position is 175.

quantiser_scale -- An unsigned integer in the range 1 to 31 used to scale the reconstruction level of the retrieved DCT coefficient levels. The decoder shall use this value until another quantiser_scale is encountered either at the slice layer or the macroblock layer. The value zero is forbidden.

extra_bit_slice -- A bit indicates the presence of the following extra information. If extra_bit_slice is set to "1", extra_information_slice will follow it. If it is set to "0", there are no data following it.

extra_information_slice -- Reserved.

slice_size: the total number of macroblocks in the slice layer (44).

7.3.6 Macroblock layer

macroblock_stuffing -- This is a fixed bit string "0000 0001 111" which can be inserted by the encoder to increase the bit rate to that required of the storage or transmission medium. It is discarded by the decoder.

macroblock_escape -- The macroblock_escape is a fixed bit-string "0000 0001 000" which is used when the difference between macroblock_address and previous_macroblock_address is greater than 33. It causes the value of macroblock_address_increment to be 33 greater than the value that will be decoded by subsequent macroblock_escapes and the macroblock_address_increment codewords.

For example, if there are two macroblock_escape codewords preceding the macroblock_address_increment, then 66 is added to the value indicated by macroblock_address_increment.

macroblock_address_increment -- This is a variable length coded integer coded as per Annex B Table B1 which indicates the difference between macroblock_address and previous_macroblock_address. -- The maximum value of macroblock_address_increment is 33. Values greater than this can be encoded using the macroblock_escape codeword.

The macroblock_address is a variable defining the absolute position of the current macroblock. The macroblock_address of the top-left macroblock is zero.

The previous_macroblock_address is a variable defining the absolute position of the last non-skipped macroblock (see Clause 6.5.4 for the definition of skipped macroblocks) except at the start of a slice. At the start of a slice previous_macroblock_address is reset as follows:

previous_macroblock_address=(slice_vertical_position-1)*mb_width-1;

The spatial position in macroblock units of a macroblock in the picture (mb_row, mb_column) can be computed from the macroblock_address as follows:

mb_row = macroblock_address / mb_width

mb_column = macroblock_address % mb_width

where mb_width is the number of macroblocks in one row of the picture.

NOTE The slice_vertical_position differs from mb_row by one.

macroblock_type -- Variable length coded indicator which indicates the method of coding and content of the macroblock according to the tables B2a through B2d.

macroblock_quant -- Derived from macroblock_type.

macroblock_motion_forward -- Derived from macroblock_type.

macroblock_motion_backward -- Derived from macroblock_type.

macroblock_pattern -- Derived from macroblock_type.

macroblock_intra -- Derived from macroblock_type.

frame_motion_type - This is a bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
00	Field-based prediction	2	field
01	Frame-based prediction	1	frame
10	CORE EXPERIMENTS	...	...
11	CORE EXPERIMENTS	...	...

field_motion_type - This is a bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
00	Field-based prediction	1	field
01	Dual-field prediction	2	field
10	16x8 MC	2	field
11	CORE EXPERIMENTS	...	...

dct_type - This is a one-bit integer indicating whether the macroblock is frame DCT coded or field DCT coded. If this is set to "1", the macroblock is field DCT coded. This flag is only transmitted in frame-picture. All macroblocks in field-pictures are field DCT coded.

motion_vector_count - This is NOT a syntax element. motion_vector_count is derived from field_motion_type or frame_motion_type as indicated in the tables.

mv_format- This is NOT a syntax element. mv_format is derived from field_motion_type or frame_motion_type as indicated in the tables. mv_format indicates if the motion vector is a field-motion vector or a frame-motion vector. mv_format is used in the syntax of the motion vectors and in the process of motion vector prediction.

motion_vertical_field_select -- This is a 1 bit defined as follows:

0	prediction comes from reference Field1
1	prediction comes from reference Field2

quantiser_scale -- This is a 5-bit code indicating the quantizer-scale. The definition is according to the qscale_type in the Picture layer. If the qscale_type is "0", quantizer_scale is 5bit binary code, i.e., from 1 to 31. If the qscale_type is "1", the quantizer_scale is defined as the non-linear sequence in Table 1. The sequence is expressed by the following formula.

Stepsize = 2 ^ (m-1) + a1 * 2 ^ (m-2) + a2 * 2 ^ (m-3)

where,

m = 0, 1, 2, 3, 4, 5, 6, 7.

a1 = 0, 1.

a2 = 0, 1.

Therefore parameters, m, a1 and a2 are transmitted to specify the stepsize. In this case m is expressed by 3 bits, and each of a1 and a2 is expressed by 1 bit. Namely 5 bits are used to transmit the stepsize.

quantizer_scale	stepsize	Binary expression
00000	0.5	0000.100
00001	0.625	0000.101
00010	0.75	0000.110
00011	0.875	0000.111
00100	1	00001.00
00101	1.25	00001.01
00110	1.5	00001.10
00111	1.75	00001.11
01000	2	000010.0
01001	2.5	000010.1
01010	3	000011.0
01011	3.5	000011.1
01100	4	0000100
01101	5	0000101
01110	6	0000110
01111	7	0000111
10000	8	0001000
10001	10	0001010
10010	12	0001100
10011	14	0001110
10100	16	0010000
10101	20	0010100
10110	24	0011000
10111	28	0011100
11000	32	0100000
11001	40	0101000
11010	48	0110000
11011	56	0111000
11100	64	1000000
11101	80	1010000
11110	96	1100000
11111	112	1110000

motion_horizontal_code -- It is denoted as motion_horizontal_forward_code for forward prediction and motion_horizontal_backward_code for backward prediction, respectively.

motion_vertical_code -- It is denoted as motion_vertical_forward_code for forward prediction and motion_vertical_backward_code for backward prediction, respectively.

motion_horizontal_r -- It is denoted as motion_horizontal_forward_r for forward prediction and motion_horizontal_backward_r for backward prediction, respectively.

motion_vertical_r -- It is denoted as motion_vertical_forward_r for forward prediction and motion_vertical_backward_r for backward prediction, respectively.

motion_horizontal_forward_code -- motion_horizontal_forward_code is decoded according to Table B4 in Annex B. The decoded value is required (along with forward_f - Clause 6.6.2) to decide whether or not motion_horizontal_forward_r appears in the bit stream.

motion_horizontal_forward_r -- An unsigned integer (of forward_r_size bits - Clause 6.6.2) used in the process of decoding forward motion vectors as described in Clause 6.6.2.

motion_vertical_forward_code -- motion_vertical_forward_code is decoded according to Table B4 in Annex B. The decoded value is required (along with forward_f - Clause 6.6.2) to decide whether or not motion_vertical_forward_r appears in the bit stream.

motion_vertical_forward_r -- An unsigned integer (of forward_r_size bits - Clause 6.6.2) used in the process of decoding forward motion vectors as described in Clause 6.6.2.

motion_horizontal_backward_code -- motion_horizontal_backward_code is decoded according to Table B4 in Annex B. The decoded value is required (along with backward_f - Clause 6.6.2) to decide whether or not motion_horizontal_backward_r appears in the bit stream.

motion_horizontal_backward_r -- An unsigned integer (of backward_r_size bits - Clause 6.6.2) used in the process of decoding backward motion vectors as described in Clause 6.6.2.

motion_vertical_backward_code -- motion_vertical_backward_code is decoded according to Table B4 in Annex B. The decoded value is required (along with backward_f) to decide whether or not motion_vertical_backward_r appears in the bit stream.

motion_vertical_backward_r -- An unsigned integer (of backward_r_size bits) used in the process of decoding backward motion vectors as described in Clause 6.6.3.

coded_block_pattern_420 -- For 4:2:0 source format, the coded_block_pattern is a variable length code that is used to derive the variable cbp according to Table B3 in Annex B. Then the pattern_code[i] for i=0 to 5 is derived from cbp using the following:

```
pattern_code[i] = 0;
        if ( cbp & (1<<(5-i)) ) pattern_code[i] = 1;
        if ( macroblock_intra ) pattern_code[i] = 1 ;
```

if pattern_code[i] equals to 1, i=0 to 5, the block number i+1 defined in the block_count is to be received in this macroblock.

coded_block_pattern_422 -- For 4:2:2 source format, an 8 bit FLC is received first and checked. If the ith bit is 1, counting from left to right and from 1 to 8, the pattern_code[i-1] = 1. The block number i defined in the block_count is to be received in this macroblock.

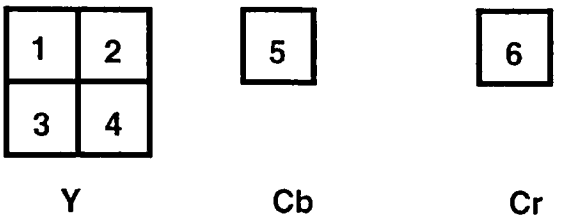
coded_block_pattern_444 -- For 4:4:4 source format, an 12 bit FLC is received first and checked. If the ith bit is 1, counting from left to right and from 1 to 8, the pattern_code[i-1] = 1. The block number i defined in the block_count is to be received in this macroblock.

block_count - This is not a syntax element. block_count is derived from chroma_format as follows :

Table 7-6 Chroma format

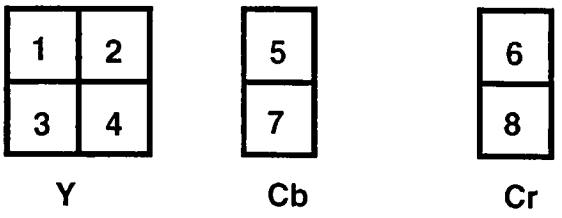
chroma_format	block_count
4:2:0	6
4:2:2	8
4:4:4	12

The 6 blocks in a 4:2:0 MB are numbered in the following order :



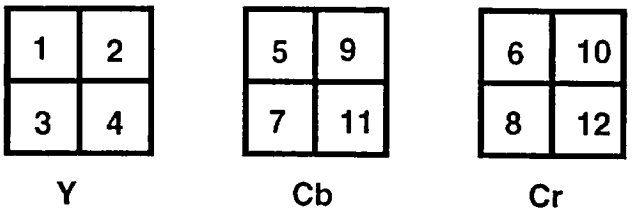
In this case, cbp is a 6-bit integer represented as : <b1><b2><b3><b4><b5><b6>, where <b1> is the most significant bit. <bn> is set to 1 when block n is coded.

The 8 blocks in a 4:2:2 MB are numbered in the following order :



In this case, cbp is a 8-bit integer represented as : <b1><b2><b3><b4><b5><b6><b7><b8>, where <b1> is the most significant bit. <bn> is set to 1 when block n is coded.

The 12 blocks in a 4:4:4 MB are in the following order :



In this case, cbp is a 1bit integer represented as : <b1>...<b12>, where <b1> is the most significant bit. <bn> is set to 1 when block n is coded.

end_of_macroblock -- This is a bit which is set to "1" and exists only in D-Pictures.

7.3.61 Reconstruction of motion vectors

Let recon_right_for and recon_down_for be the reconstructed horizontal and vertical components of the motion vector for the current macroblock, and recon_right_for_prev and recon_down_for_prev be the reconstructed motion vector for the previous predictive-coded macroblock. If the current macroblock is the first macroblock in the slice, or if the last macroblock that was decoded contained no motion vector information (either because it was skipped or macroblock_motion_forward was zero), then recon_right_for_prev and recon_down_for_prev shall be set to zero.

If no forward motion vector data exists for the current macroblock (either because it was skipped or macroblock_motion_forward == 0), the motion vectors shall be set to zero.

If forward motion vector data exists for the current macroblock, then any means equivalent to the following procedure shall be used to reconstruct the motion vector horizontal and vertical components.

forward_r_size and forward_f are derived from forward_f_code as follows:

```
            forward_r_size = forward_f_code - 1
            forward_f = 1 << forward_r_size
        if ( (forward_f == 1) || (motion_horizontal_forward_code == 0) ) {
            complement_horizontal_forward_r = 0;
        } else {
            complement_horizontal_forward_r = forward_f - 1 - motion_horizontal_forward_r;
        }
        if ( (forward_f == 1) || (motion_vertical_forward_code == 0) ) {
            complement_vertical_forward_r = 0;
        }
        else {
            complement_vertical_forward_r = forward_f - 1 - motion_vertical_forward_r;
        }
        right_little = motion_horizontal_forward_code * forward_f;
        if (right_little == 0) {
            right_big = 0;
        }
        else {
            if (right_little > 0) {
                right_little = right_little - complement_horizontal_forward_r ;
                right_big = right_little - (32 * forward_f);
            }
            else {
                right_little = right_little + complement_horizontal_forward_r ;
                right_big = right_little + (32 * forward_f);
            }
        }
        down_little = motion_vertical_forward_code * forward_f;
        if (down_little == 0) {
            down_big = 0;
        }
        else {
            if (down_little > 0) {
                down_little = down_little - complement_vertical_forward_r ;
                down_big = down_little - (32 * forward_f);
            }
            else {
                down_little = down_little + complement_vertical_forward_r ;
                down_big = down_little + (32 * forward_f);
            }
        }
```

Values of forward_f, motion_horizontal_forward_code and if present, motion_horizontal_forward_r shall be such that right_little is not equal to forward_f * 16.

Values of forward_f, motion_vertical_forward_code and if present, motion_vertical_forward_r shall be such that down_little is not equal to forward_f * 16.

```
        max = ( 16 * forward_f ) - 1 ;

        min = ( -16 * forward_f ) ;
```

Frame motion vector

For frame predicted macroblock, there is only one motion vector, whose two components are recon_right_for and recon_down_for respectively. The two components are reconstructed from the above parameters as follows:

```
new_vector = recon_right_for_prev + right_little ;
if ( (new_vector <= max) && (new_vector >= min) )
    recon_right_for = recon_right_for_prev + right_little ;
else
    recon_right_for = recon_right_for_prev + right_big ;
recon_right_for_prev = recon_right_for ;
if ( full_pel_forward_vector ) recon_right_for = recon_right_for << 1 ;
new_vector = recon_down_for_prev + down_little ;
if ( new_vector <= max && new_vector >= min )
    recon_down_for = recon_down_for_prev + down_little ;
else
    recon_down_for = recon_down_for_prev + down_big ;
recon_down_for_prev = recon_down_for ;
if ( full_pel_forward_vector ) recon_down_for = recon_down_for << 1 ;
```

Field motion vector

For field predicted macroblock, there are two motion vectors, whose four components are recon_right_i_for1, recon_right_i_for2, recon_down_i_for1 and recon_down_i_for2 respectively. The four components are reconstructed from the above parameters as follows:

At first the recon_right_i_for1 and recon_down_i_for1 are reconstructed from first set of VLC codes decoded from the bit stream.

```
new_vector = recon_right_for_prev + right_little ;
if ( (new_vector <= max) && (new_vector >= min) )
    recon_right_i_for1 = recon_right_for_prev + right_little ;
else
    recon_right_i_for1 = recon_right_for_prev + right_big ;
recon_right_for_prev = recon_right_i_for1 ;
if ( full_pel_forward_vector ) recon_right_i_for1 = recon_right_i_for1 << 1 ;
new_vector = recon_down_for_prev + down_little ;
if ( new_vector <= max && new_vector >= min )
    recon_down_i_for1 = recon_down_for_prev + down_little ;
else
    recon_down_i_for1 = recon_down_for_prev + down_big ;
recon_down_for_prev = recon_down_i_for1 ;
if ( full_pel_forward_vector ) recon_down_i_for1 = recon_down_i_for1 << 1 ;
```

Second the recon_right_i_for2 and recon_down_i_for2 are reconstructed from second set of VLC codes decoded from the bit stream. The same formulae used to generate recon_right_i_for1 and recon_down_i_for1 are used, except for replacing recon_right_i_for1 with recon_right_i_for2 and recon_down_i_for1 with recon_down_i_for2.

The motion vectors in whole pixel units for the macroblock, right_for and down_for, and the half pixel unit flags, right_half_for and down_half_for, are computed as follows:

for luminance	**for 4:2:0 chrominance**
right_for = recon_right_for >> 1 ;	right_for = (recon_right_for / 2) >> 1 ;
down_for = recon_down_for >> 1 ;	down_for = (recon_down_for / 2) >> 1 ;
right _half_for = recon_right_for - (2*right_for);	right_half_for = recon_right_for/2 - (2*right_for) ;
down_half_for = recon_down_for - (2*down_for) ;	down_half_for = recon_down_for/2 - (2*down_for);

for 4:2:2 chrominance	**for 4:4:4 chrominance**
right_for = (recon_right_for / 2) >> 1 ;	right_for = recon_right_for >> 1 ;
down_for = recon_down_for >> 1 ;	down_for = recon_down_for >> 1 ;
right _half_for = recon_right_for/2 - (2*right_for);	right_half_for = recon_right_for - (2*right_for) ;
down_half_for = recon_down_for - (2*down_for) ;	down_half_for = recon_down_for - (2*down_for);

Note: For field motion vectors, the recon_right_for and recon_down_for should be replaced with recon_right_i_for1 and recon_down_i_for1 respectively for Field1. For Filed2, they should be replaced with recon_right_i_for2 and recon_down_i_for2 respectively.

Motion vectors leading to references outside a reference picture's boundaries are not allowed.

A positive value of the reconstructed horizontal motion vector (right_for) indicates that the referenced area of the past reference picture is to the right of the macroblock in the coded picture.

A positive value of the reconstructed vertical motion vector (down_for) indicates that the referenced area of the past reference picture is below the macroblock in the coded picture.

7.3.7 Block layer

dct_dc_size_luminance -- The number of bits in the following dct_dc_differential code, dc_size_luminance, is derived according to the VLC Table B5a.

dct_dc_size_chrominance -- The number of bits in the following dct_dc_differential code, dc_size_chrominance, is derived according to the VLC Table B5b.

dct_dc_differential -- A variable length unsigned integer. If dc_size_luminance or dc_size_chrominance (as appropriate) is zero, then dct_dc_differential is not present in the bit stream. dct_zz [] is the array of quantised DCT coefficients. The first element of the zigzag-scanned quantised DCT coefficient list, dct_zz[0], is equal to zero. If dc_size_luminance or dc_size_chrominance (as appropriate) is greater than zero, then dct_zz[0] is computed as follows from dct_dc_differential:

For luminance blocks:

```
if ( dct_dc_differential & ( 1 << (dc_size_luminance-1)) ) dct_zz[0] = dct_dc_differential ;
else dct_zz[0] = ( -1 << (dc_size_luminance) ) | (dct_dc_differential+1) ;
```

For chrominance blocks:

```
if ( dct_dc_differential & ( 1 << (dc_size_chrominance-1)) ) dct_zz[0] = dct_dc_differential ;
else dct_zz[0] = ( -1 << (dc_size_chrominance) ) | (dct_dc_differential+1) ;
```

dct_zz[i], i>0 shall be set to zero initially.

example for dc_size_luminance = 3	
dct_dc_differential	dct_zz[0]
000	-7
001	-6
010	-5
011	-4
100	4
101	5
110	6
111	7

dct_coeff_first -- A variable length code according to tables B.5c through B.5g in Annex B for the first coefficient. The variables run and level are derived according to these tables. The zigzag-scanned quantised DCT coefficient list is updated as follows.

```
i = run ;
if ( s == 0 ) dct_zz[i] = level ;
if ( s == 1 ) dct_zz[i] = - level ;
```

The terms dct_coeff_first and dct_coeff_next are run-length encoded and dct_zz[i], i>=0 shall be set to zero initially. A variable length code according to tables B5c through B5g is used to represent the run-length and level of the DCT coefficients.

dct_coeff_next -- A variable length code according to tables B.5c through B.5g in Annex B for coefficients following the first retrieved. The variables run and level are derived according to these tables. The zigzag-scanned quantised DCT coefficient list is updated as follows.

```
i = i + run +1 ;
if ( s == 0 ) dct_zz[i] = level ;
```

```
        if ( s == 1 ) dct_zz[i] = - level ;
```

If macroblock_intra == 1 then the term i shall be set to zero before the first dct_coeff_next of the block. The decoding of dct_coeff_next shall not cause i to exceed 63.

end_of_block -- This symbol is always used to indicate that no additional non-zero coefficients are present. It is used even if dct_zz[63] is non-zero.

pattern_code(i) - For slave_blocks, this code is the same as that of the correlated scaled_block in the slice layer.

more_coefs - more_coefs is true if we have not already decoded the last coefficient in the block of DCT coefficients except that, for the 8x8 slave_slice, more_coefs is always true (this is to retain compatibility with MPEG-1 style of coding 8x8 blocks, which always includes an end_of_block code).

eob_code - An end_of_block Huffman code specified in the appropriate resolution scale VLC table.

next_dct_coef - DCT coefficient coded by run/amplitude or run/size VLCs. The VLC table used depends on "dct_size", as explained in Annex D.

8 The video decoding process

[Start with a diagram of the simple decoder (ie. no scalable extensions) essentially moved from its present position in the introduction.

This section tells us what to do with the numbers recovered from the semantics section (7.3) in order to reconstruct pictures. Essentially all of the arithmetic operations that take place are defined here. Where a technique is used in more than one context then it is introduced in a non-specific way first and then the specific instances are enumerated togther with any special rules or restrictions that affect that case. For instance motion compensation is described in the general case first before being described in the case of Intra macroblocks, Predicted Macroblocks and Bidirectional Macroblocks (note the new MPEG-2 case of Intra MB's with vectors).]

8.1 Dequantisation and inverse transform

8.1.1 Dequantisation

Define dct_recon[m][n] to be the matrix of dequantised DCT coefficients, where the first index identifies the row and the second the column of the matrix.

Intra-coded macroblocks

This clause applies to all macroblocks in Intra-Frames and Intra macroblocks in Predicted and Interpolated Frames.

Define dct_dc_y_past, dct_dc_cb_past and dct_dc_cr_past to be the dct_recon[0][0] of the most recently decoded intra-coded Y, Cb and Cr blocks respectively. The predictors dct_dc_y_past, dct_dc_cb_past and dct_dc_cr_past shall all be reset at the start of a slice and at non-intra-coded macroblocks (including skipped macroblocks) to a value according to the received code of intra_dc_precision.

if (intra_dc_precision == '00') initial or reset value = 128;

if (intra_dc_precision == '01') initial or reset value = 256;

if (intra_dc_precision == '10') initial or reset value = 512;

if (intra_dc_precision == '11') initial or reset value = 1024;

Define past_intra_address as the macroblock_address of the most recently retrieved intra-coded macroblock within the slice. It shall be reset to -2 at the beginning of each slice.

Reconstruction levels, dct_recon(i,j), are derived from the following formulae.

```
dct_recon(i,j) = quantizer_scale * 2 * QAC(i,j) * wI(i,j) / 16
if (dct_recon(i,j)  is an EVEN number && dct_recon(i,j) > 0)
    dct_recon(i,j) = dct_recon(i,j) - 1
if (dct_recon(i,j)  is an EVEN number && dct_recon(i,j) < 0)
    dct_recon(i,j) = dct_recon(i,j) + 1
if (QAC (i,j) == 0)
    dct_recon(i,j) = 0
The DC term is a special case depending on the value of intra_dc_precision.
    if (intra_dc_precision == '00') dct_recon(0,0) = 8*QDC;
    if (intra_dc_precision == '01') dct_recon(0,0) = 4*QDC;
    if (intra_dc_precision == '10') dct_recon(0,0) = 2*QDC;
    if (intra_dc_precision == '11') dct_recon(0,0) = QDC;
```

Where:

quantizer_scale is the quantization parameter stored in the bitstream

wI(i,j) is the (i,j)th element of the Intra quantiser matrix given in Clause 6.5.

The DC term obtained above is only the differential value, the actual value is computed by any means equivalent to the following procedures:

For the first luminance block

```
        if ( ( macroblock_address - past_intra_address > 1) )
            dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
        else
            dct_recon[0][0] = dct_dc_y_past + dct_recon[0][0] ;
        dct_dc_y_past = dct_recon[0][0] ;
```

For the rest of the luminance blocks

```
        dct_recon[0][0] = dct_dc_y_past + dct_recon[0][0] ;
        dct_dc_y_past = dct_recon[0][0] ;
```

For the first chrominance Cb block

```
        if ( ( macroblock_address - past_intra_address > 1) )
            dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
        else
            dct_recon[0][0] = dct_dc_cb_past + dct_recon[0][0] ;
        dct_dc_cb_past = dct_recon[0][0] ;
```

For the rest of the chrominance Cb blocks

```
        dct_recon[0][0] = dct_dc_cb_past + dct_recon[0][0] ;
        dct_dc_cb_past = dct_recon[0][0] ;
```

For the first chrominance Cr block

```
        if ( ( macroblock_address - past_intra_address > 1) )
            dct_recon[0][0] = (128 * 8) + dct_recon[0][0] ;
        else
            dct_recon[0][0] = dct_dc_cr_past + dct_recon[0][0] ;
        dct_dc_cr_past = dct_recon[0][0] ;
```

For the rest of the chrominance Cr blocks

```
        dct_recon[0][0] = dct_dc_cr_past + dct_recon[0][0] ;
        dct_dc_cr_past = dct_recon[0][0] ;
```

The computed dct_recon(i.j) is limited to the range [-2048..2047] by cropping operation.

After all the blocks in the macroblock are processed:

```
        past_intra_address = macroblock_address ;
```

Non-Intra-coded macroblocks

This clause applies to all non-Intra macroblocks in Predicted and Interpolated Pictures. Reconstruction levels, dct_recon(i,j), are derived from the following formulae.

```
        if (QAC(i,j) > 0)
            dct_recon(i,j) =  (2 * QAC(i,j) + 1) * quantizer_scale * wN(i,j) / 16
        if (QAC(i,j) < 0)
            dct_recon(i,j) =  (2 * QAC(i,j) - 1) * quantizer_scale * wN(i,j) / 16
        if (dct_recon(i,j)  is an EVEN number && dct_recon(i,) > 0)
            dct_recon(i,j) = dct_recon(i,j) - 1
        if (dct_recon(i,j)  is an EVEN number && dct_recon(i,j) <0)
            dct_recon(i,j) = dct_recon(i,j) + 1
        if (QAC (i,j) == 0)
            dct_recon(i,j) = 0
```

where $w_N(i,j)$ is the (i,j)th element of the non-intra quantiser matrix, given in Clause 6.5.

The above computed dct_recon(i,j) is limited to the range [-2048..2047] by cropping operation.

8.1.2 Bitstream to 2-D data conversion

The picture data in the bitstream is an one dimensional stream that is converted from two dimensional matrix data by scanning. A similar inverse procedure is needed for decoder to convert the one dimensional bitstream into two dimensional matrix data. This conversion is described in this clause.

The discussion of semantics has defined mb_row and mb_column, which locate the macroblock in the picture. The definitions of dct_dc_differential, and dct_coeff_next also have defined the zigzag-scanned quantised DCT coefficient list, dct_zz[]. Each dct_zz[] is located in the macroblock as defined by pattern_code[].

Define QAC[i][j] to be the AC component of DCT coefficients in matrix format, where the first index identifies the row and the second the column of the matrix. Define QDC is the DC component of DCT coefficients.

Define scan[i][j] to be the matrix defining the zigzag scanning sequence as follows:

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

Where j is the horizontal index and i is the vertical index.

Then the QAC[i][j] and QDC can be converted from dct_zz as:

QAC[i][j] = dct_zz[scan[i][j]]

QDC = dct_zz[0]

The converted QAC[i][j] and QDC are the quantised DCT coefficients. They are further processed by the following dequantisation procedure.

8.1.3 Inverse DCT transform

Once the DCT coefficients are reconstructed, the inverse DCT transform defined in Annex A shall be applied to obtain the inverse transformed pixel values. If the tcoef_escape_format flag is set to 0, the pixel values are in the range of [-256,255]. If the tcoef_escape_format flag is set to 1, the pixel values are in the range of [-2048,2047]. These pixel values shall be limited to the range [0, 255] by cropping operation and placed in the luminance and chrominance matrices in the positions defined by mb_row , mb_column, and the list defined by the array pattern_code[]. The macroblock shall be reconstructed according to the structure of Figure 6-2 or Figure 6-3 depends on whether the DCT is frame type or field type.

8.1.4 Forced updating

This function is achieved by forcing the use of an intra-coded macroblock. The update pattern is not defined. For control of accumulation of IDCT mismatch error, each macroblock shall be intra-coded at least once per every 132 times it is coded in a P-picture.

8.2 Motion compensation

8.2.1 Frame motion compensation

In frame motion compensation there is one vector per macroblock. Vectors measure displacements on a frame sampling grid. Therefore an odd-valued vertical displacement causes a prediction from the fields of opposite parity. Vertical half pixel values are interpolated between samples from fields of opposite parity. Chrominance vectors are obtained directly by using the formulae above. The vertical motion compensation for interlaced picture is illustrated in Figure 6-5.

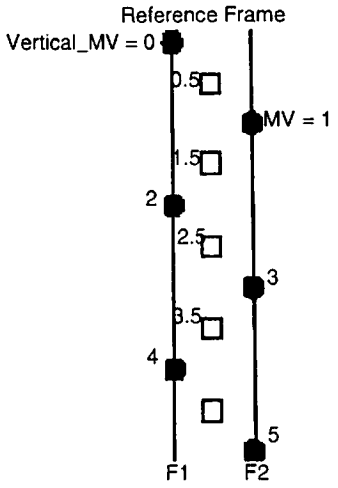


Figure 8-1 Frame Motion Compensation

8.2.2 Field motion compensation

Field motion vectors expressed in the very same way as frame vectors would be if the reference field and the predicted field were considered as "frames" (see Figure 6-6).

Considering that in each field picture, lines are numbers 1.0, 2.0, 3.0, ... (1 is the top line of the field), if the pixel located at line "n" of the predicted field is predicted from line "m" of the reference field, the vertical coordinate of the field vector is "n-m".

Note: "m" and "n" are expressed in units of one vertical half-pixel in the field.

For P-field pictures the two last decoded P-field pictures are used as prediction references. For B-field pictures, the reference field shall be from the previous decoded I- or P-frame. The reference field is identified by the **forward_reference_fields** or **backward_reference_fileds**. If the aforementioned two codes are "11", the **motion_vertical_field_select** (one bit) is used to identify the selected field as reference.

The vertical motion compensation for interlaced picture is illustrated in Figure 6-6.

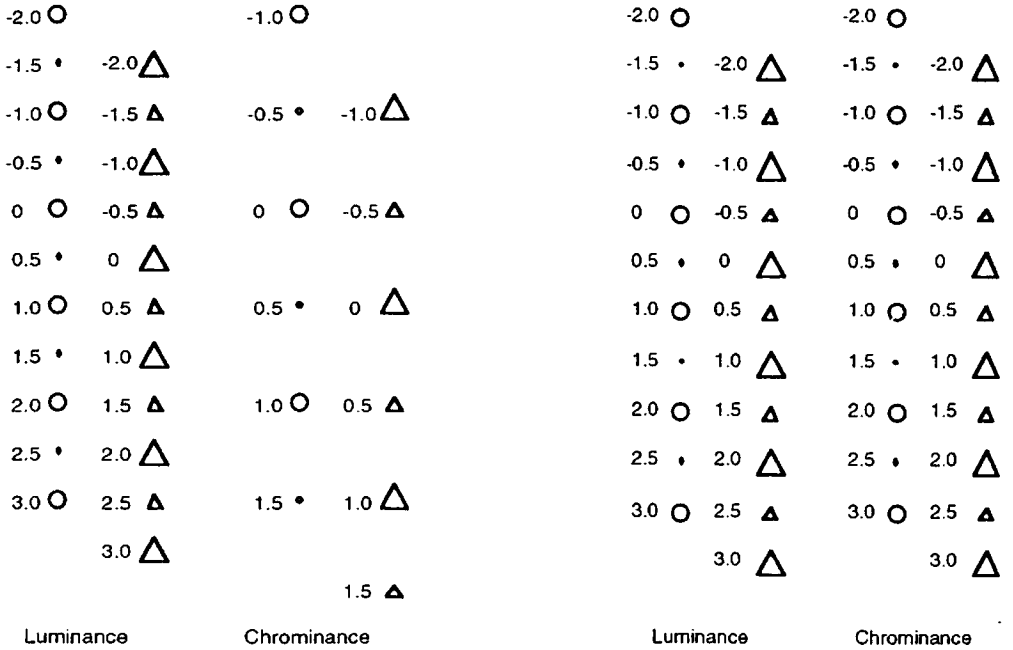


Figure 8-2 Field motion compensation

8.2.3 Motion compensation formulae

Both frame and field motion compensations are calculated by the following formulae. Defining pel_past[][] as the pixels of the past picture referenced by the forward motion vector, and pel[][] as the pixels of the picture being decoded, then:

```
if ( (! right_half_for )&& (! down_half_for) )
    pel[i][j] = pel_past[i+down_for][j+right_for] ;
if ( (! right_half_for) && down_half_for )
    pel[i][j] = ( pel_past[i+down_for][j+right_for] +
        pel_past[i+down_for+1][j+right_for] ) // 2 ;
if ( right_half_for && (! down_half_for) )
    pel[i][j] = ( pel_past[i+down_for][j+right_for] +
        pel_past[i+down_for][j+right_for+1] ) // 2 ;
if ( right_half_for && down_half_for )
    pel[i][j] = ( pel_past[i+down_for][j+right_for] +
        pel_past[i+down_for+1][j+right_for] +
        pel_past[i+down_for][j+right_for+1] +
        pel_past[i+down_for+1][j+right_for+1] ) // 4 ;
```

For field motion vectors in frame pictures, there are two field motion vectors for prediction. The odd lines of the frame should be predicted from the first field motion vector while the even lines of the frame should be predicted from the second field motion vector.

The pel[i][j] which were computed above using the motion vectors shall be added to the inverse DCT transformed pixel values as described in 6.6.2.2. The result of the addition shall be limited to the interval [0,255] by cropping operation and placed in the luminance and chrominance matrices in the positions defined by mb_row , mb_column, and the list defined by the array pattern_code[].

8.3 Reconstruction of intra-coded macroblocks

8.4 Reconstruction of predictive-coded macroblocks

Predictive-coded macroblocks are decoded in three steps. First, the DCT coefficient information stored for some or all of the blocks is decoded, dequantised, inverse DCT transformed as described in Section 6.6.2. Second, the value of the forward motion vector for the macroblock is reconstructed and a prediction macroblock is formed, as detailed below. Third, the DCT coded macroblock is add to the prediction macroblock to form the full reconstruction of a predictive-coded macroblock.

8.5 Reconstruction of bidirectional predictive-coded macroblocks

Predictive-coded macroblocks in B-Pictures are decoded in four steps.

First, the value of the forward motion vector for the macroblock is reconstructed from the retrieved forward motion vector information, and the forward motion vector reconstructed for the previous macroblock. The same algorithm used in 6.6.3 is applied to generate following vector components, which define the integral and half pixel value of the rightward and downward components of the forward motion vector (which references the past picture in display order)

right_for right_half_for down_for down_half_for

However, for B-coded pictures the previous reconstructed motion vectors shall be reset only for the first macroblock in a slice, and when the last macroblock that was decoded was an intra-coded macroblock. If no forward motion vector data exists for the current macroblock, the motion vectors shall be obtained by :

recon_right_for = recon_right_for_prev,

recon_down_for = recon_down_for_prev.

Second, the value of the backward motion vector for the macroblock shall be reconstructed from the retrieved backward motion vector information, and the backward motion vector reconstructed for the previous macroblock using the same procedure as for calculating the forward motion vector in B-pictures. However, all the variables with "forward" should be changed to "backward", "for" to "back" and "prev" to "future". The vector components generated are listed below:

right_back right_half_back down_back down_half_back

which defines the integral and half pixel value of the rightward and downward components of the backward motion vector (which references the future picture in display order).

Third, the pixels of the decoded picture are calculated. If only forward motion vector information was retrieved for the macroblock, then pel[][] of the decoded picture shall be calculated according to the formulae in Clause 6.6.3. If only backward motion vector information was retrieved for the macroblock, then pel[][] of the decoded picture shall be calculated according to the formulae in the predictive-coded macroblock clause, with "back" replacing "for", and pel_future[][] replacing pel_past[][]. If both forward and backward motion vectors information are retrieved, then let pel_for[][] be the value calculated from the past picture by use of the reconstructed forward motion vector, and let pel_back[][] be the value calculated from the future picture by use of the reconstructed backward motion vector. Then the value of pel[][] shall be calculated by:

```
pel[][] = ( pel_for[][] + pel_back[][] ) // 2 ;
```

Fourth, the DCT coefficients for each block present in the macroblock shall be reconstructed by the means introduced in 6.6.2.2. The inverse DCT transformed pixel values shall be added to pel[][], which were computed above from the motion vectors. The result of the addition shall be limited to the interval [0,255] by cropping operation and placed in the luminance and chrominance matrices in the positions defined by mb_row , mb_column, and the list defined by the array pattern_code[].

8.6 Scalable extensions

8.6.1 Spatial scalable extension

{Start with a diagram showing the decoder for this type of decoding.}

8.6.2 Frequency scalable extension

{Start with a diagram showing the decoder for this type of decoding.}

9 Defined profiles and levels

Attention is drawn to clause 5.5 which defines the convention for specifying a range of numbers. This is used throughout to specify the constrained range of values and parameters.

{Editors note: If MPEG defines more profiles (in additon to "Main") these will be described in clauses 9.2, 9.3 etc.}

9.1 Main profile

Bit streams that are compliant with the Main profile shall meet the restrictions set out in clause 9.1.

In addition bit streams that are compliant with the Main level (of the main profile) shall meet the restrictions set out in clause 9.2.

{Editors note: If MPEG defines more levels in the Main profile (in additon to the "Main" level) these will be described in clauses 9.1.2, 9.1.3 etc.}

9.1.1 Main profile syntax

9.1.11 Chroma sampling structure

The chroma sampling structure (defined by chroma_format) shall be 4:2:0.

9.1.12 Spatial scalability

The spatial_embedded flag shall be zero.

9.1.13 Frequency scalability

The fscalable flag shall be zero.

9.1.14 Motion compensation

{Editors note: The fields describing this still seem to be in a state of change. The intention is that frame motion compensation with motion vectors based on either 8x8 or 16x8 boundaries is not allowed since this calls to accesses to fields in units of four lines with consequent implementation overheads. I believe that the current TM4 semantics do not include such modes and so no constraint is required. However the current semantics in this document (subject to imminent review!) do include such modes.

Can anyone help me out here?}

9.1.2 Main level

9.1.21 Picture dimensions

The horizontal size of the picture (defined by horizontal_size_value and horizontal_size_extension) shall be in the range <0:720] pels and shall be a multiple of 16.

The vertical size of the picture (defined by vertical_size_value and vertical_size_extension) shall be in the range <0:576] pels and shall be a multiple of 16. In the case of interlaced pictures this constraint refers to the number of lines in the frame, there will be half this number in each field.

The picture rate (defined by picture_rate) shall be in the range <0:30] pictures per second (and is constrained to the discrete values that picture_rate can represent). In the case of interlaced pictures this constraint refers to the frame rate and the field rate will be double this value.

In addition the product of the horizontal size, the vertical size and the picture rate shall be limited to lie in the range <0:10 368 000] pels per second.

9.1.22 Coded data rate and VBV buffer size

The coded data rate (for fixed bit rate operation specified in bit_rate) shall be limited to the range <0:15 000 000] bits per second.

The VBV buffer size (vbv_buffer_size) shall be limited to the range <0:2 097 152] bits. *{The exact value requires further study}*

9.1.23 Vector range

The reconstructured vertical motion vectors shall be limited to the range [-128:+127.5]. (This range is [-256:+255] half-pel units.)

[Editors Note: Resolution 3.4.5 from Rome calls for parameters at the sequence layer to specify the maximum horizontal and vertical vector range. These have yet to be added. When they are this clause can be written more precisely]

Annex A

Discrete cosine transform

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

The NxN two dimensional DCT is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

with u, v, x, y = 0, 1, 2, ... N-1

where x, y are spatial coordinates in the pixel domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

The input to the forward transform and output from the inverse transform is represented with 9 bits. The coefficients are represented in 12 bits. The dynamic range of the DCT coefficients is [-2048:+2047].

The N by N inverse discrete transform shall conform to IEEE Draft Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, P1180/D2, July 18, 1990. Note that Clause 2.3 P1180/D2 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. The maximum refresh period requirement for this standard shall be 132 pictures, the same as indicated in P1180/D2 for visual telephony according to CCITT Recommendation H.261

(see Bibliography).

Annex B

Variable length code tables

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

B.1 Macroblock addressing

Table B-1 --- Variable length codes for macroblock_address_increment

macroblock_address_ increment VLC code	increment value	macroblock_address_ increment VLC code	increment value
1	1	0000 0101 10	17
011	2	0000 0101 01	18
010	3	0000 0101 00	19
0011	4	0000 0100 11	20
0010	5	0000 0100 10	21
0001 1	6	0000 0100 011	22
0001 0	7	0000 0100 010	23
0000 111	8	0000 0100 001	24
0000 110	9	0000 0100 000	25
0000 1011	10	0000 0011 111	26
0000 1010	11	0000 0011 110	27
0000 1001	12	0000 0011 101	28
0000 1000	13	0000 0011 100	29
0000 0111	14	0000 0011 011	30
0000 0110	15	0000 0011 010	31
0000 0101 11	16	0000 0011 001	32
		0000 0011 000	33
		0000 0001 000	macroblock_escape

B.2 Macroblock type

The properties of the macroblock are determined by the macroblock type VLC according to these tables.

Table B-3 --- Variable length codes for macroblock_type in I-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	macroblock_compatible
1	0	0	0	0	1	0
01	1	0	0	0	1	0

Table B-4 --- Variable length codes for macroblock_type in P-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	macroblock_compatible
1	0	1	0	1	0	0
01	0	0	0	1	0	0
001	0	1	0	0	0	0
0001 1	0	0	0	0	1	0
0001 0	1	1	0	1	0	0
0000 1	1	0	0	1	0	0
0000 01	1	0	0	0	1	0

Table B-5 --- Variable length codes for macroblock_type in B-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	macroblock_compatible
10	0	1	1	0	0	0
11	0	1	1	1	0	0
010	0	0	1	0	0	0
011	0	0	1	1	0	0
0010	0	1	0	0	0	0
0011	0	1	0	1	0	0
0001 1	0	0	0	0	1	0
0001 0	1	1	1	1	0	0
0000 11	1	1	0	1	0	0
0000 10	1	0	1	1	0	0
0000 01	1	0	0	0	1	0

Table B-6 --- Variable length codes for macroblock_type in D-pictures

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	macroblock_compatible
1	0	0	0	0	1	0

B.3 Macroblock pattern

Table B-7 --- Variable length codes for coded_block_pattern.

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51
0110 1	52	0001 0001	23
0110 0	56	0001 0000	43
0101 1	1	0000 1111	25
0101 0	61	0000 1110	37
0100 1	2	0000 1101	26
0100 0	62	0000 1100	38
0011 11	24	0000 1011	29
0011 10	36	0000 1010	45
0011 01	3	0000 1001	53
0011 00	63	0000 1000	57
0010 111	5	0000 0111	30
0010 110	9	0000 0110	46
0010 101	17	0000 0101	54
0010 100	33	0000 0100	58
0010 011	6	0000 0011 1	31
0010 010	10	0000 0011 0	47
0010 001	18	0000 0010 1	55
0010 000	34	0000 0010 0	59
0001 1111	7	0000 0001 1	27
0001 1110	11	0000 0001 0	39
0001 1101	19	0000 0000 1	0 (NOTE)
NOTE — This entry shall not be used with 4:2:0 chrominance structure			

B.4 Motion vectors

Note. Table B-8 is representative of proposals that use a method for vector coding that is similar to that used for DC Intra coefficients. There are various tables proposed and indeed this table cannot be used since it causes start code emulation. The method of coding motion vectors has not yet been finalised.}

Table B-8 --- Variable length codes for motion_size

Variable length code	motion_size
0	0
10	1
110	2
1110	3
1111 0	4
1111 10	5
1111 110	6
1111 1110	7
1111 1111 0	8
1111 1111 10	9
1111 1111 110	10
1111 1111 1110	11
1111 1111 1111 0	12
1111 1111 1111 10	13

B.5 DCT coefficients

Table B-9 --- Variable length codes for dct_dc_size_luminance

Variable length code	dct_dc_size_luminance
100	0
00	1
01	2
101	3
110	4
1110	5
1111 0	6
1111 10	7
1111 110	8
1111 1110	9
1111 1111 0	10
1111 1111 10	11

Table B-10 --- Variable length codes for dct_dc_size_chrominance

Variable length code	dct_dc_size_chrominance
00	0
01	1
10	2
110	3
1110	4
1111 0	5
1111 10	6
1111 110	7
1111 1110	8
1111 1111 0	9
1111 1111 10	10
1111 1111 110	11

{NOTE Tables B-11 and B-12 show two tables that have been proposed for use with the adaptive VLC technique. Alternative tables have also been proposed for both of these tables. No decision has been made about which tables to use.}

Table B-11 --- DCT coefficients table zero

Variable length code (NOTE1)	run	level
10	End of Block	
1 s (NOTE2)	0	1
11 s (NOTE3)	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1 s	0	3
0011 1 s	3	1
0011 0 s	4	1
0001 10 s	1	2
0001 11 s	5	1
0001 01 s	6	1
0001 00 s	7	1
0000 110 s	0	4
0000 100 s	2	2
0000 111 s	8	1
0000 101 s	9	1
0000 01	Escape	
0010 0110 s	0	5
0010 0001 s	0	6
0010 0101 s	1	3
0010 0100 s	3	2
0010 0111 s	10	1
0010 0011 s	11	1
0010 0010 s	12	1
0010 0000 s	13	1
0000 0010 10 s	0	7
0000 0011 00 s	1	4
0000 0010 11 s	2	3
0000 0011 11 s	4	2
0000 0010 01 s	5	2
0000 0011 10 s	14	1
0000 0011 01 s	15	1
0000 0010 00 s	16	1
NOTE1 - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative. NOTE2 - This code shall be used for dct_coeff_first. NOTE3 - This code shall be used for dct_coeff_next.		

Table B-11 --- DCT coefficients table zero (continued)

Variable length code (NOTE)	run	|level|
0000 0001 1101 s	0	8
0000 0001 1000 s	0	9
0000 0001 0011 s	0	10
0000 0001 0000 s	0	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

2 **Table B-11 --- DCT coefficients table zero (continued)**

Variable length code (NOTE)	run	level
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

Table B-11 --- DCT coefficients table zero (concluded)

Variable length code (NOTE)	run	llevell
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

1

Table B-12 —DCT coefficients table one

VLC code	run	level
11 s	0	1
101 s	0	2
100 s	1	1
0111 s	End of Block	
0110 s	0	3
0101 1 s	0	4
0101 0 s	0	5
0100 1 s	1	2
0100 0 s	2	1
0011 1 s	3	1
0011 01 s	0	6
0011 00 s	0	7
0010 11 s	4	1
0010 10 s	5	1
0010 011 s	0	8
0010 010 s	0	9
0010 001 s	1	3
0010 000 s	2	2
0001 111 s	6	1
0001 110 s	7	1
0001 101 s	8	1
0001 100 s	9	1
0001 011 s	10	1
0001 0101 s	Escape	
0001 0100 s	0	10
0001 0011 s	0	11
0001 0010 s	0	12
0001 0001 s	0	13
0001 0000 s	1	4
0000 1111 s	3	2
0000 1110 s	11	1
0000 1101 s	12	1
0000 1100 s	13	1
0000 1011 1 s	0	14
0000 1011 0 s	0	15
0000 1010 1 s	0	16
0000 1010 0 s	0	17
0000 1001 1 s	1	5
0000 1001 0 s	2	3
0000 1000 1 s	4	2
0000 1000 0 s	14	1
0000 0111 11 s	0	18
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

2

1

Table B-12 —DCT coefficients table one (continued)

VLC code	run	|level|
0000 0111 10 s	0	19
0000 0111 01 s	0	20
0000 0111 00 s	0	21
0000 0110 11 s	0	22
0000 0110 10 s	1	6
0000 0110 01 s	1	7
0000 0110 00 s	3	3
0000 0101 11 s	5	2
0000 0101 10 s	6	2
0000 0101 01 s	7	2
0000 0101 00 s	15	1
0000 0100 111 s	0	23
0000 0100 110 s	0	24
0000 0100 101 s	0	25
0000 0100 100 s	0	26
0000 0100 011 s	0	27
0000 0100 010 s	0	28
0000 0100 001 s	1	8
0000 0100 000 s	2	4
0000 0011 111 s	4	3
0000 0011 110 s	8	2
0000 0011 101 s	9	2
0000 0011 100 s	10	2
0000 0011 011 s	16	1
0000 0011 010 s	17	1
0000 0011 001 s	24	1
0000 0011 000 s	25	1
0000 0010 1111 s	0	29
0000 0010 1110 s	0	30
0000 0010 1101 s	0	31
0000 0010 1100 s	0	32
0000 0010 1011 s	0	33
0000 0010 1010 s	1	9
0000 0010 1001 s	1	10
0000 0010 1000 s	1	11
0000 0010 0111 s	2	5
0000 0010 0110 s	3	4
0000 0010 0101 s	5	3
0000 0010 0100 s	11	2
0000 0010 0011 s	12	2
0000 0010 0010 s	13	2
0000 0010 0001 s	18	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

2

1

Table B-12 —DCT coefficients table one (continued)

VLC code	run	|level|
0000 0010 0000	19	1
0000 0001 1111	20	1
0000 0001 1110	21	1
0000 0001 1101	22	1
0000 0001 1100	23	1
0000 0001 1011	26	1
0000 0001 1010	29	1
0000 0001 1001	30	1
0000 0001 1000	32	1
0000 0001 0111 1	0	34
0000 0001 0111 0	0	35
0000 0001 0110 1	0	36
0000 0001 0110 0	0	37
0000 0001 0101 1	0	38
0000 0001 0101 0	0	39
0000 0001 0100 1	0	40
0000 0001 0100 0	1	12
0000 0001 0011 1	1	13
0000 0001 0011 0	1	14
0000 0001 0010 1	2	6
0000 0001 0010 0	3	5
0000 0001 0001 1	4	4
0000 0001 0001 0	6	3
0000 0001 0000 1	7	3
0000 0001 0000 0	8	3
0000 0000 1111 1	9	3
0000 0000 1111 0	10	3
0000 0000 1110 1	14	2
0000 0000 1110 0	27	1
0000 0000 1101 1	28	1
0000 0000 1101 0	31	1
0000 0000 1100 1	33	1
0000 0000 1100 01	0	41
0000 0000 1100 00	0	42
0000 0000 1011 11	0	43
0000 0000 1011 10	0	44
0000 0000 1011 01	0	45
0000 0000 1011 00	0	46
0000 0000 1010 11	1	15
0000 0000 1010 10	1	16
0000 0000 1010 01	1	17
0000 0000 1010 00	2	7
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

2

1

Table B-12 —DCT coefficients table one (continued)

VLC code	run	level
0000 0000 1001 11 s	2	8
0000 0000 1001 10 s	3	6
0000 0000 1001 01 s	4	5
0000 0000 1001 00 s	5	4
0000 0000 1000 11 s	6	4
0000 0000 1000 10 s	7	4
0000 0000 1000 01 s	8	4
0000 0000 1000 00 s	11	3
0000 0000 0111 11 s	12	3
0000 0000 0111 10 s	13	3
0000 0000 0111 01 s	34	1
0000 0000 0111 001 s	0	47
0000 0000 0111 000 s	0	78
0000 0000 0110 111 s	0	49
0000 0000 0110 110 s	0	50
0000 0000 0110 101 s	0	51
0000 0000 0110 100 s	1	18
0000 0000 0110 011 s	1	19
0000 0000 0110 010 s	1	20
0000 0000 0110 001 s	1	21
0000 0000 0110 000 s	1	22
0000 0000 0101 111 s	2	9
0000 0000 0101 110 s	3	7
0000 0000 0101 101 s	4	6
0000 0000 0101 100 s	5	5
0000 0000 0101 011 s	9	4
0000 0000 0101 010 s	10	4
0000 0000 0101 001 s	13	4
0000 0000 0101 000 s	14	3
0000 0000 0100 111 s	35	1
0000 0000 0100 110 s	36	1
0000 0000 0100 101 s	40	1
0000 0000 0100 100 s	41	1
0000 0000 0100 0111 s	0	52
0000 0000 0100 0110 s	0	53
0000 0000 0100 0101 s	0	54
0000 0000 0100 0100 s	0	55
0000 0000 0100 0011 s	0	56
0000 0000 0100 0010 s	1	23
0000 0000 0100 0001 s	1	24
0000 0000 0100 0000 s	1	25
0000 0000 0011 1111 s	1	26
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

2

Table B-12 —DCT coefficients table one (concluded)

VLC code	run	level
0000 0000 0011 1110 s	1	27
0000 0000 0011 1101 s	1	28
0000 0000 0011 1100 s	1	29
0000 0000 0011 1011 s	2	10
0000 0000 0011 1010 s	2	11
0000 0000 0011 1001 s	3	8
0000 0000 0011 1000 s	4	7
0000 0000 0011 0111 s	5	6
0000 0000 0011 0110 s	6	5
0000 0000 0011 0101 s	7	5
0000 0000 0011 0100 s	8	5
0000 0000 0011 0011 s	9	5
0000 0000 0011 0010 s	11	4
0000 0000 0011 0001 s	12	4
0000 0000 0011 0000 s	12	5
0000 0000 0010 1111 s	13	5
0000 0000 0010 1110 s	14	4
0000 0000 0010 1101 s	15	2
0000 0000 0010 1100 s	16	2
0000 0000 0010 1011 s	24	2
0000 0000 0010 1010 s	25	2
0000 0000 0010 1001 s	37	1
0000 0000 0010 1000 s	38	1
0000 0000 0010 0111 s	39	1
0000 0000 0010 0110 s	42	1
0000 0000 0010 0101 s	43	1
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

Table B-13 --- Encoding of run and level following an ESCAPE code

fixed length code	run
0000 00	0
0000 01	1
0000 10	2
...	...
...	...
...	...
...	...
1111 11	63

fixed length code	level
100000000001	-2047
100000000010	-2046
...	...
111111111111	-1
000000000000	forbidden
000000000001	+1
...	...
011111111111	+2047

Annex C

Video buffering verifier

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

Constant rate coded video bit streams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in Clause C.1.

The VBV is a hypothetical decoder which is conceptually connected to the output of an encoder. Coded data is placed in the buffer at the constant bit rate that is being used. Coded data is removed from the buffer as defined in Clause C.1.4, below. It is a requirement of the encoder (or editor) that the bit stream it produces will not cause the VBV to either overflow or underflow.

1 The VBV and the video encoder have the same clock frequency as well as the same picture rate, and are operated synchronously.

2 The VBV has a receiving buffer of size B, where B is given in the vbv_buffer_size field in the sequence header.

3 The VBV is initially empty. It is filled from the bit stream for the time specified by the vbv_delay field in the video bit stream.

4 *This item applies to cases that all pictures are coded and transmitted.* All of the data for the picture which has been in the buffer longest is instantaneously removed. Then after each subsequent picture interval all of the data for the picture which (at that time) has been in the buffer longest is instantaneously removed. Sequence header and group of picture layer data elements which immediately precede a picture are removed at the same time as that picture. The VBV is examined immediately before removing any data (sequence header data, group of picture layer or picture) and immediately after each picture is removed. Each time the VBV is examined its occupancy shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by vbv_buffer_size in the sequence header.

5 *This item applies to cases that some pictures are not coded nor transmitted.*

Encoder may wish to realize low buffering delay with allowing occasional picture droppings. It will regulate its information generation by setting a virtual buffer size B_I smaller than B for stationary pictures. The VBV operates as follows.

All of the data for the non-dropped picture which has been in the buffer longest is instantaneously removed. Then after each interval all data of the non-dropped picture which (at that time) has been in the buffer longest is instantaneously removed. Sequence header, group of picture layer data elements and headers of dropped pictures which immediately precede a picture are removed at the same time as that picture. At some decoding timing where picture dropping takes place in the coder, there will be no sufficient data to remove. In that case, no data removing takes place.

During the stationary state with low buffering delay, the VBV occupancy immediately after each picture is removed shall lie between zero and (B_I - R/P).

To meet this requirement the number of bits dc for the coded picture just before the steady state (including any preceding header and group of picture layer data elements) must satisfy;

no dropped picture	$B_p - B_I + R/P < dc < B_p$
one dropped picture	$B_p - B_I + 2R/P < dc < B_p + R/P$
two dropped pictures	$B_p - B_I + 3R/P < dc < B_p + 2R/P$
n dropped pictures	$B_p - B_I + (n+1)R/P < dc < B_p + nR/P$

where:

B_p: VBV buffer occupancy just before removing the data
B_I: VBV buffer corresponding to low delay operation
R : bit rate
P : picture rate

This is a requirement on the video bit stream including coded picture data, user data and all stuffing.

To meet these requirements the number of bits for the (n+1)'th coded picture d_{n+1} (including any preceding sequence header and group of picture layer data elements) must satisfy:

$d_{n+1} > B_n + (2R/P) - B$

$d_{n+1} <= B_n + (R/P)$

where:

$n >= 0$

B_n is the buffer occupancy just after time t_n

R = bit_rate

P = number of pictures per second

t_n is the time when the n'th coded picture is removed from the VBV buffer

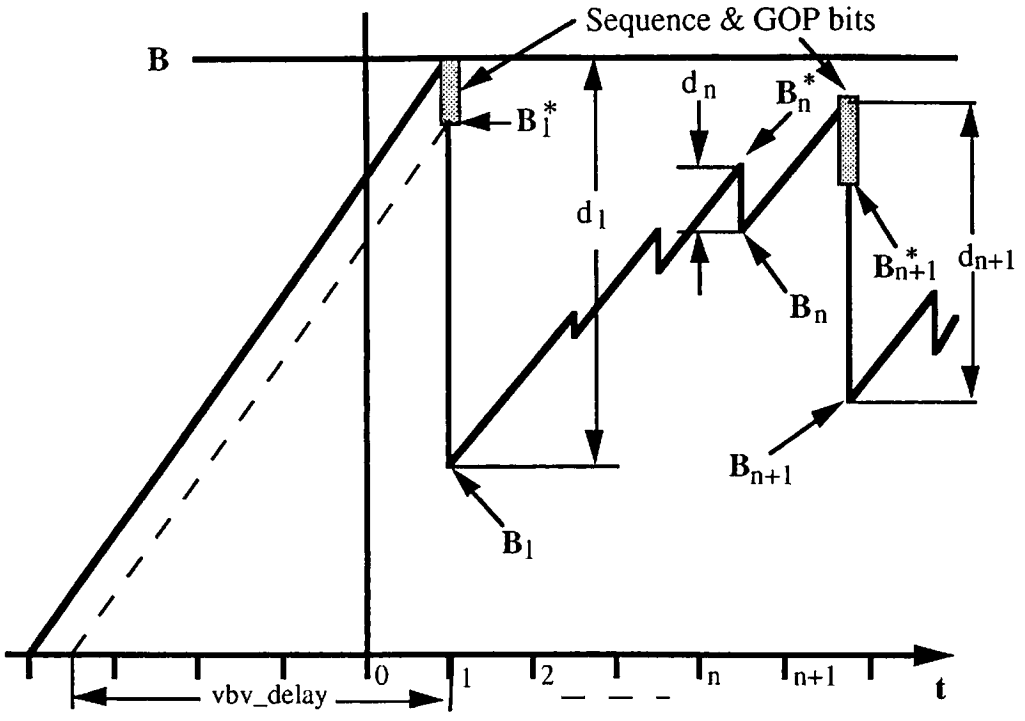


Figure C-1 VBV Buffer Occupancy

Annex D

Features supported by the algorithm

(This annex does not form an integral part of this Recommendation | International Standard)

{This whole section needs to be thoroughly reviewed. In particular comments on error resilience need rewriting since MPEG-2 has many new ways of tackling these issues. Topics such as Low delay, Compatibility and bit streams with multiple picture resolutions need to be written.}

D.1 General comments

Applications using compressed video on digital storage media need to be able to perform a number of operations in addition to normal forward play of the sequence. The coded bit stream has been designed to support a number of these operations.

D.1.1 Flexibility in bitrate

The number of transmitted bits per unit time may be controlled in two ways. For the constant bit rate (CBR) coding, the number of transmitted bits per unit time is constant on the channel. Since the encoder output rate generally varies depending on the picture content, it shall regulate the rate constant by buffering etc. In CBR, picture quality may vary depending on its content. The other way is the variable bit rate (VBR) coding, in which case the number of transmitted bits per unit time may very on the channel under some constriction. VBR is expected to provide constant quality coding.

D.1.2 Random access

Random access is an essential feature for video on a storage medium. Random access requires that any picture can be decoded in a limited amount of time. It implies the existence of access points in the bit stream -- that is segments of information that are identifiable and can be decoded without reference to other segments of data. A spacing of two random access points (Intra-Pictures) per second can be achieved without significant loss of picture quality.

D.1.3 Editing

There is a conflict between the requirement for high compression and easy editing. The coding structure and syntax have not been designed with the primary aim of simplifying editing at any picture. Nevertheless a number of features have been included that enable editing of coded data.

D.1.4 Trick mode

D.1.41 Fast playback (forward, backward)

Depending on the storage medium, it is possible to scan the access points in a compressed bit stream (with the help of an application specific directory or other knowledge beyond the scope of this Specification) to obtain a fast-forward and backward effect.

D.1.42 Reverse playback

Some applications may require the video signal to be played in reverse order. This can be achieved in a decoder by using memory to store entire groups of pictures after they have been decoded before being displayed in reverse order. An encoder can make this feature easier by reducing the length of groups of pictures.

D.1.5 Error resilience

Most digital storage media and communication channels are not error-free. Appropriate channel coding schemes should be used and are beyond the scope of this Specification. Nevertheless the compression scheme defined in this Specification is robust to residual errors. The slice structure allows a decoder to recover after a data error and to resynchronize its decoding. Therefore, bit errors in the compressed data will cause errors in the decoded pictures to be limited in area. Decoders may be able to use concealment strategies to disguise these errors.

D.1.6 Real time aspect ratio changes

D.1.61 Pan/scan

Indicating sender's desire concerning what part of the 16:9 picture be displayed on 4:3 monitors. Desired portion may be panned or scanned.

D.1.62 Letter box

16:9 signal is reduced in vertical resolution and displayed on a 4:3 monitor maintaining original aspect ratio.

D.2 Low delay

(Text to be written)

D.3 Error resilience

Most digital storage media and communication channels are not error-free. Appropriate channel coding schemes should be used and are beyond the scope of this Specification. Nevertheless the compression scheme defined in this Specification is robust to residual errors. The slice structure allows a decoder to recover after a data error and to resynchronize its decoding. Therefore, bit errors in the compressed data will cause errors in the decoded pictures to be limited in area. Decoders may be able to use concealment strategies to disguise these errors.

D.4 Mutliple resolution bitstreams

(Text to be written)

D.5 Compatibility

(Text to be written)

Annex E

Encoding

(This annex does not form an integral part of this Recommendation | International Standard)

{This section is NOT intended to be a thorough treatment of encoding. It is intended as a brief introduction to the encoding process similar to that which used to appear in the introduction (see MPEG-1). In MPEG-2 there are occasions when different approaches to encoding are possible. In particular the various possibilities in the frequency scalable world. I gather that various altenative arrangements are possible with consequent cost/quality trade-offs. These issues would be dealt with here.}

Annex F

Bibliography

(This annex does not form an integral part of this Recommendation | International Standard)

1 Arun N. Netravali & Barry G. Haskell "Digital Pictures, representation and compression" Plenum Press, 1988

2 Didier Le Gall "MPEG: A Video Compression Standard for Multimedia Applications" Trans. ACM, April 1991

3 C Loeffler, A Ligtenberg, G S Moschytz "Practical fast 1-D DCT algorithms with 11 multiplications" Proceedings IEEE ICASSP-89, Vol. 2, pp 988-991, Feb. 1989

4 See the Normative Reference for CCIR Rec 601

5 IEC Standard
Publication 461
Second edition 1986
"Time and control code for video tape recorders"

6 CCITT Recommendation H.261
Codec for audiovisual services at px64 kbit/s"
Geneva, 1990

7 IEEE Standard Specification for the Implementation of 8 by 8 Inverse Discrete Cosine Transform" P1180/D2 July 18 1990

8 ISO 10918-1 (JPEG) "Digital compression and coding of continuous-tone still images"

9 E Viscito and C Gonzales "A Video Compression Algorithm with Adaptive Bit Allocation and Quantization", Proc SPIE Visual Communications and Image Proc '91 Boston MA November 10-15 Vol. 1605 205, 1991

10 A Puri and R Aravind "Motion Compensated Video Coding with Adaptive Perceptual Quantization", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 1 pp 351 Dec. 1991.